PLANETARY MICRO-ROVERS WITH BAYESIAN AUTONOMY

MARK A. POST

A Dissertation submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Graduate Program in Earth and Space Science
York University
Toronto, Ontario

April 2014

Abstract

In this dissertation, we present the Beaver μrover, a 6kg solar-powered planetary rover that is designed to perform exploration missions such as the Northern Light Mars mission, as well as to extend the capabilities of modern robotics here on Earth. By developing systems from the ground up using a pragmatic design approach for modularity and expandability, commercial hardware, open-source software, and novel implementations of probabilistic algorithms, we have obtained a comprehensive set of hardware and software designs that can form the basis for many kinds of intelligent but low-cost robots. A lightweight tubular chassis that can be simply deployed protects sensors, actuators, and wiring, and a novel four-wheel independently driven and passively actuated suspension with enclosed brushless gear motors can stably handle steep slopes and low obstacles. A nanosatellite-sized electronics stack incorporates Linux or dedicated RTOS computing on the ARM architecture, highly-efficient battery charging and power conversion, MEMS and external sun sensors, a powerful hybrid motor controller, and a vision system. Separate rovers and programmable components communicate using a novel network communications architecture over synchronous serial buses and mesh network radio communications. Intelligent autonomy is made possible using probabilistic methods programmed with fixed-point arithmetic for efficiency, incorporating Kalman filters and a Bayesian network constructed both from prior knowledge and from the implicit structure of the hardware and software present and used for inference and decision-making. Navigation makes use of both external sensors and visual SLAM by using optical flow and structure-from-motion methods. Detailed descriptions and comparisons of all systems are given, and it is shown that using a basic set of sensors and the vision system, basic navigational and problem-solving tasks can be performed. Thermal vacuum testing of components is also done to validate their operation under space conditions.

Dedication

"Your paradigm is just the door, your attitude is the key."

"It is the view of the problem that defines the solution."

"Within these unlimited degrees of freedom, nothing is impossible."

I Dedicate this Dissertation

to my Wife and our Families.

Thank You, and Love Always.

Mark Andrew Post, 2013

ACKNOWLEDGEMENTS

The analogy of "standing on the shoulders of giants" has been used by Salisbury, Hooke, and many others. Truly, only by building on each other's work and sharing each other's vision can we see further. Of course, it is important to recognize all those who have directly or indirectly contributed to this work.

First of all, thanks must go to my supervisors for taking me on as a graduate student, for giving me the freedom to explore my own ideas and directions, and for having the patience to allow me to see everything through. Prof. Brendan Quine provided the Northern Light mission, as well as many valuable ideas and interesting anecdotes for space hardware development, and Prof. Regina Lee has provided many opportunities for nanosatellite hardware collaboration and industrial partnerships that have helped my research career immeasurably. Thanks also go to Hugh Chesser for further opportunities to develop on nanosatellite hardware. I would also like to express gratitude to my graduate student colleagues in the York University Space Hardware Laboratory. Nimal Navarathinam, Kartheephan Sathiyanathan, Ian Proper, Tyler Ustrzycki, Guy Benari, Sriyan Wisnarama, Thomas Wright, Thong Thai, and Hugh Podmore have all been kind, helpful, and in general great to work with. Konstantin Borschiov has been extremely helpful in OBC development on several occasions, and Gowry Sinnathamby helped to design the prototype for the vision board. Recognition goes to Abhay Rai for designing the ground-penetrating radar system for the μrover, for maintaining the lab, and for patience as we both learn RF electronics.

My thanks and greetings are extended to all the members of the York University Rover Team who I have worked with over the years who have contributed both directly and indirectly to my research. Recognized in particular are the contributions of Chanpreet Amole who helped develop the design of the μrover's suspension system, Stanley Lio who has had endless good ideas for electronics and communications system, and created the original datalink layer concept that I have used, Dennis Liu for showing all of us just how a rover's GUI should be done, and Tom Young who contributed much brainstorming, design, and programming work to motor drive and actuator control systems. Mike Liscombe contributed greatly to my understanding of robot electronics, Bart Verzijlenberg developed an excellent vision and remote surveying system that served as inspiration, and Houman Hakima and Isaac De Souza contributed vast amounts of mechanical development, and suspension design. Kudos also goes out to Saurabh Bhardwaj and Vaibhav Kapoor for building brilliant communications, electronics, and vision systems and teaching me at least as much as I taught them. Special recognition also goes to Jesse Tebbs and Vincent Huynh both for forming YURT and for managing to get me involved.

On a deeper level, thank you to both of my parents for helping me to develop my passion for science and technology early in life, for teaching me that the most important things in life are to be happy and have fun, and for allowing me the freedom to fumble my way along to reach where I am now. We really do owe who we are to those who raised us. Most of all, I want to thank my partner and wife Lili for her love and dedication to me even at the most difficult of times, for her diligent supervision and guidance throughout my research, and for teaching me how to be a successful academic as well as a successful human being.

Table of Contents

List of Tables

List of Figures

NOMENCLATURE

List of Symbols

$\wp$	The set of all random variables in a Bayesian network
β	Coefficient for likelihood of a correct sensor reading
λ_r	Radio wavelength used for communications
ζ	Fuzzy basis function for fuzzy controller
θ	Heading angle about Z axis, in radians
ϑ	Angle between drive wheel's plane of rotation and the the ground, in radians
ϑ	Adaptive parameter for fuzzy controller
μ_W	Fuzzy membership function for fuzzy controller
ν	Set of membership rules for fuzzy function
χ	Sampled test point for sigma-point Kalman filter representing covariance
σ	Standard deviation for a random variable or set, for which σ^2 is the variance
Σ	Normalization sum for Bayesian inference calculation
ς	Sign of a parameterized control component given a certain movement operation
τ	Torque about a central axis, in newton-metres
τ_m	Output torque of a drive motor, in newton-metres
τ_{rated}	Rated torque of a drive motor, in newton-metres
τ_{stall}	Stall torque of a drive motor, in newton-metres
τ_{max}	Absolute maximum torque of a drivetrain, in newton-metres
τ_c	Control input for mobility controller
τ_d	Disturbance for mobility controller
ϕ	Lift angle about Y axis, in radians
χ	Sigma point values used by a sigma point Kalman filter
ψ	Tilt angle about X axis, in radians
ω	Angular velocity of heading about Z axis, in radians or deciradians per second
ω_m	Angular velocity of motor drive shaft, in radians per second
Ω	Set of all possible outcomes within an event space

a, b, c, d, e, f	Generic scalar coefficient values used for the solution of equations
l, m, n, i, j	Generic indices for iteration over discrete elements
$\mathbf{d}$	Map vector from current location to target location in map coordinates, in metres
d_{max}	Maximum mapping target search radius, in metres
e	Error term, generally the difference between a desired value and a measured value
f_g	Linear force due to gravitational acceleration on a body with mass, in Newtons
f_m	Linear force generated by a motor and wheel in solid ground contact, in Newtons
f_r	Radio frequency used for communications
g	Gravitational acceleration constant, $9.81 m/s^2$ for earth, $3.711 m/s^2$ for Mars
k	Constant coefficient scalar, generally a pre-set parameter
l	Length of a mechanical part or length of wire
m	Mass of a body, in kilograms
p	Scalar, discrete probability value
q	Instantaneous value of system white (Gaussian) noise assumed by a Kalman filter
r	Instantaneous value of measurement white (Gaussian) noise assumed by a Kalman filter
r	Radius of a circle, coil, or wheel, in metres
r_{max}	Maximum range of a range sensor, in metres
t	Time scalar, represented as an integer time step for discrete time systems
u	Image plane coordinate in the horizontal direction, planar analogue to x, in pixels
u	Scalar speed of vehicle body in the vertical direction, in meters per second
v	Image plane coordinate in the vertical direction, planar analogue to y, in pixels
v	Scalar speed of vehicle body in the forward direction, in meters per second
w_r	Angular field of view of a range sensor, in radians
x	Scalar coordinate in first Cartesian axis direction, vehicle forward, in metres
y	Scalar coordinate in second Cartesian axis direction, vehicle rightward, in metres
z	Scalar coordinate in third Cartesian axis direction, vehicle downward, in metres
$\mathbf{e}$	Base polynomial vector for egomotion optimization
$\mathbf{k}$	Auxiliary variable vector for egomotion optimization
$\mathbf{l}$	Location Vector in map coordinates, size $3x1$ for three dimensions, in metres
$\mathbf{o}$	Relative Origin location for start of mapping in map coordinates, size $3x1$, in metres

$\mathbf{q}$	Quaternion Orientation, size $4x1$
$\mathbf{t}$	Translation Vector, size $3x1$ for three dimensions
$\mathbf{u}$	Control input velocity vector with components u_x and u_y, in metres per second
$\mathbf{v}$	Measured velocity vector with components v_x and v_y, in metres per second
$\mathbf{x}$	Current location state of the vehicle in map coordinates, in metres
$\mathbf{x}$	Estimated state vector of vehicle including position and orientation, in SI units
$\mathbf{y}$	Measured state vector of vehicle from sensors and associated models, in SI units
A, B, C, D	Generic random variables used for probabilistic inference
X, Y, Z	Generic random variables used for probabilistic inference
B	Random variable for behaviour selection
C	Electrical capacitance of a capacitor or plate
F	State function for chassis control
G	Actuator function for chassis control
I	Electrical current, $I = V/R$, measured in Amps
K_v	DC motor speed constant, measured in radians per second per Volt
K_t	DC motor torque constant, measured in newton-metres per Amp
L	Electrical Inductance of an inductor or coil of wire
L, M, N	Maximum values for indices l, m and n
O	Set of measurable events or actions associated with a probability distribution, subset of Ω
P_e	Electrical power, $P_e = VI$, measured in Watts
P_m	Mechanical power, $P_m = \omega\tau$, measured in Watts
R	Electrical Resistance, $R = V/I$, measured in Ohms
S	Switching function (or sliding surface) for sliding-mode control
T	Total time in a repeated period or total time for an experiment or test
V	Electrical Voltage, $V = IR$, measured in Volts
V	Lyapunov function
Wm	Sigma point weighting for obtaining mean in sigma-point Kalman filter
Wc	Sigma point weighting for obtaining covariance in sigma-point Kalman filter
$\mathbf{0}$	Zero matrix of size mxn with all elements equal to 0
$\mathbf{B}$	Electromagnetic field created by current moving through a length of wire

K	Kalman Gain matrix for Kalman filter update operation
H	Rotational bilinear optimization matrix for egomotion optimization
M	Known set of mapped landmarks and probabilistic regions
M	Translational bilinear optimization matrix for egomotion optimization
N	Noise matrix for egomotion optimization
E	Essential matrix for camera characterization
F	Fundamental matrix for camera triangulation
F	Intensity matrix for optical flow
I	The *mxm* identity matrix, in which only the values of 1 along the diagonal are nonzero
P	State Vector Covariance for Kalman filter
Q	State Noise Covariance for Kalman filter
R	Measurement Noise Covariance for Kalman filter
R	Rotation Matrix, size $3x3$ for three dimensions
S	Sensor Innovation Covariance for Adaptive Kalman filter
T	Orientation Tensor, size $3x3$ for three dimensions
U	Matrix of left singular value decomposition vectors, size $3x3$ for three dimensions
V	Matrix of right singular value decomposition vectors, size $3x3$ for three dimensions
W	Orthogonal matrix used for two-view orientation calculation
X	Control state matrix for chassis control
An(X)	The set of all reachable ancestors of node X in a Bayesian network
Ch(X)	The set of all direct children of node X in a Bayesian network
De(X)	The set of all reachable descendants of node X in a Bayesian network
E(X)	Expected Value of random variable X
H(X)	Informational entropy present in the distribution of a random variable X
P(X)	Probability of random variable X, represented in normalized form as a fraction of 1.0
Pa(X)	The set of all direct parents of node X in a Bayesian network
SVD(**X**)	The singular value decomposition of matrix **X**
V(X)	The set of possible mapped values that random variable X can take

List of Conventions

x	Individual scalar values or specific values of a random variable are shown as lowercase
X	Random variables with many values or a continuous distribution are denoted by uppercase
$\dot{x}$	The first derivative of a time-varying variable with respect to time is denoted by a dot
$\ddot{x}$	The second derivative of a time-varying variable with respect to time is denoted by two dots
$\mathbf{X}$	Matrices of m rows by n columns are indicated by **bold** uppercase
$\mathbf{x}$	Vectors of m rows by 1 column are indicated by **bold** lowercase
$\mathbf{X}_{mxn}$	Subscripting indicates the size of a matrix is m rows and n columns
x_{ij}	Scalar elements x of a matrix X are referenced by row i and column j
$\hat{\mathbf{x}}$	The caret hat indicates a vector is of unit length. $\lvert\hat{\mathbf{x}}\rvert == 1$
$\bar{\mathbf{x}}$	The bar above a variable indicates the mean or expected value of that variable. $\bar{\mathbf{x}} == \mathrm{E}(X)$
$\tilde{\mathbf{x}}$	The tilde of a variable or matrix indicates that it is an estimated or approximate value.
$0b11111111$	The prefix $0b$ is used to denote a value given in the binary number system (base 2).
$0xFF$	The prefix $0x$ is used to denote a value given in the hexadecimal number system (base 16).

List of Abbreviations

AC	Alternating Current
ACS	Attitude Control System
AFSMC	Adaptive Fuzzy Sliding Mode Control
API	Application Program Interface
ASIC	Application Specific Integrated Circuit
AUKF	Adaptive Unscented Kalman Filter
BIF	Bayesian Interchange Format
BLDC	BrushLess DC
BN	Bayesian Network
BNIF	Bayesian Network Interchange Format
BRIEF	Binary Robust Independent Elementary Features
BRP	Bayesian Robot Programming
CCD	Charge-Coupled Device
CCW	Counter-ClockWise (rotation)
CKF	Cubature Kalman Filter
CLARAty	Coupled Layer Architecture for Robotic Autonomy
CMOS	Complementary Metal-Oxide Semiconductor
COBS	Consistent-Overhead Byte Stuffing
COTS	Commercial Off-The-Shelf
CPD	Conditional Probability Distribution
CW	ClockWise (rotation)
DAG	Directed Acyclic Graph
DBN	Dynamic Bayesian Network
DC	Directed Current
DDIT	Differential Digital Image Tracking
DIC	Digital Image Correlation
DUT	Device Under Test
ECSS	European Cooperation for Space Standardization

EAP	Electroactive Polymer
EKF	Extended Kalman Filter
EMF	Electromotive Force
FAST	Features from Accelerated Segment Test
FDI	Fault Detection and Isolation
FOSS	Free and Open-Source Software
FPU	Floating-point Processing Unit
FSM	Finite State Machine
GCC	GNU Compiler Collection (formerly GNU C Compiler)
GDB	GNU DeBugger
GNU	GNU's Not Unix (recursive)
GPL	GNU Public License
HITL	Hardware-In-The-Loop
HMM	Hidden Markov Model
IC	Integrated Circuit
IDE	Integrated Development Environment
IDL	Interface Definition Language
IMU	Inertial Measurement Unit
IP	Internet Protocol
ISR	Interrupt Service Routine
JPL	Jet Propulsion Laboratory (Caltech)
KF	Kalman Filter
LQE	Linear Quadratic Estimator
LSB	Least Significant Bit
MAP	Maximum A Posteriori
MCU	MicroController Unit
MDP	Markov Decision Process
MIMO	Multiple Input Multiple Output
MPE	Most Probable Explanation
MSB	Most Significant Bit

MUKF	Minimum sigma set Unscented Kalman Filter
NASA	National Aeronautics and Space Administration (USA)
NP	Numerical Processing
OBC	On-Board Computer
ORB	Oriented FAST and Rotated BRIEF
OS	Operating System
OSI	Open Systems Interconnection
PCB	Printed Circuit Board
PDF	Probability Distribution Function
PID	Proportional-Integral-Derivative
POMDP	Partially Observable Markov Decision Process
POSIX	Portable Operating System Interface
PPI	Parallel Peripheral Interface
PWM	Pulse Width Modulation
RANSAC	RANdom SAmple Consensus
RCET	Rover Chassis Evaluation Tools
RPC	Remote Procedure Call
RPET	Rover Performance Evaluation Tools
RT	Real-Time
RTEMS	Real-Time Executive for Multiprocessor Systems
RTOS	Real Time Operating System
RX	Receive
PDF	Probability Distribution Function
SEPIC	Single-Ended Primary-Inductance Converter
SEU	Single Event Upset
SfM	Structure from Motion
SISO	Single Input Single Output
SLAM	Simultaneous Location And Mapping
SMP	Symmetric Multi-Processing
SMT	Surface MounT

SPI	Serial Peripheral Interface
SPKF	Sigma-Point Kalman Filter
SSH	Secure SHell
SVD	Singular Value Decomposition
TCP	Transmission Control Protocol
TRIUMF	Tri-University Meson Facility
TS	Task Switch
TX	Transmit
UKF	Unscented Kalman Filter
UT	Unscented Transform
UTP	Unshielded Twisted Pair
UUB	Uniformly Ultimately Bounded
VM	Virtual Machine
XML	Extensible Markup Language
XBN	XML Bayesian Network
YURT	York University Rover Team
YUSEND	York University Space Engineering Nanosatellite Demonstration

Chapter 1

Introduction

This dissertation describes the design and programming of an autonomous micro-rover for use in exploration of planetary or terrestrial environments, using modern embedded hardware and software, probabilistic reasoning, and monocular vision. In this first chapter, we introduce our μrover and the various systems and concepts necessary for its operation that will be covered. The key concepts for this research include the innovative application of cutting-edge commercial hardware to the challenges of autonomous space systems, and the use of probabilistic structures for intelligent reasoning and control.

1.1 Overview

The increasing number of successful robotic systems in place on earth and in space has provided a good precedent for using robots in place of, or in concert with, humans to perform difficult or dangerous tasks in remote locations. Most of these systems are very complex and specialized, and generally require constant human surveillance to compensate for the limited problem-solving abilities of most robots, requiring large and costly engineering and mission planning teams. While many methods exist for making mobile robots more self-sufficient, the constraints of operating a mechatronic system with limited mobility and computing resources in a hostile, inherently uncertain environment generally impose severe limits on what planetary rovers can safely and reliably do, and the cost per kilogram of launches into space and to other planets makes risk avoidance the highest priority in large-scale exploration missions. To maintain the possibility of exploring other worlds, better methods must be found for implementing more intelligence into less hardware, using more modern components with higher fault tolerance but lower cost, and gaining the ability to distribute missions among a number of agents in case of loss. Much like other technology-based fields, the trend in the space industry is leaning toward smaller, faster, and cheaper. But unlike most other fields, the freedom to compromise is very limited, and new technologies are generally distrusted until proven in space.

In recent years, the development of inexpensive commercial off-the-shelf (COTS) hardware for mobile and embedded systems has expanded exponentially to support the relatively new global market for mobile data and communications systems such as cellular phones, personal information devices, and other information technology systems. Chiefly, the availability of fast but very efficient microprocessors, high-bandwidth digital radios, high-density lithium-polymer batteries, and micro-electro-mechanical sensors and drive systems has now made it possible to inexpensively construct intelligent, mobile systems with remarkable flexibility and a range of capabilities. Among the many fields that have appeared to take advantage of these new capabilities is the field of small-scale space hardware. Micro-, nano-, and even pico-satellites capable of returning useful data from orbit have been proposed and developed, and now offer a feasible alter-

native to the multi-million-dollar giants launched in the last half-century. A well known example is the CubeSat, a common form factor for nanosatellites that is based on a $10cmx10cmx10cm$ cube of under $1.33kg$ mass (considered to be 1 unit in size, or 1U) and can be extended up to two units (2U, $10cmx10cmx20cm$) or three units (3U, $10cmx10cmx30cm$) with the corresponding linear increases in mass. The CubeSat specification was developed in 1999 by California Polytechnic State University and Stanford University for use in research and educational nanosatellite programs, and CubeSats have been launched for purposes ranging from technology demonstration and amateur radio relay to scientific earth observation missions, and by organizations ranging from small student groups to large aerospace companies.

Using similar technologies and principles, the use of small, inexpensive and efficient unmanned ground and air vehicles for planetary exploration is now not only feasible, but an active area of interest for many research groups. While the Mars Science Laboratory "Curiosity" has recently started its journey of exploration across the Martian surface and represents the largest and most complex planetary rover yet developed, there are many scientific tasks that could also be completed by (or even, could be better suited for) groups of small, inexpensive rovers. This includes environmental monitoring, large-area ground surveys, communications relay purposes, material transportation, and short-range subsurface mapping. The use of many small, inexpensive robots that work in concert, rather than one large robot, has great potential in tasks such as these. It would be beneficial for simple mission tasks to have a low-cost, self-sufficient robotic system that is capable of overcoming most of the problems encountered in day-to-day operation.

With the popularization of open-source code distribution made possible by the Internet and the availability of Linux and the GNU project, a vast amount of open-source code has become available and is now used for shared development of robots and embedded systems of all kinds. While many proprietary systems are still used on critical projects, a growing number of private, research, and commercial robotic systems now run on Linux and open-source based software frameworks. The advantages of using and providing open-source code are many, including that bugs and security holes can be found and fixed quickly by any of numerous users, that modifications and improvements can be made and incorporated into a common tree, and that the

combined knowledge of the entire user base can be leveraged to improve and expand capabilities. It is practical, particularly in a research environment, to focus on leveraging technologies from, and contributing to, the open-source community.

An immediate opportunity for development of such robots is the Northern Light mission, a Canadian initiative to send one or more landers to the surface of Mars to study the Martian environment [Quine *et al.* 2008]. A model of the planned lander module for this mission is shown in Figure 1.1. York University is the official Research Host of the Northern Light mission, which is led by Prof. Brendan Quine and Thoth Technology, Inc. It is planned to include a micro-robot, known as the Beaver rover, which will leave the lander and perform geological surveying and imaging of the Martian surface [Post *et al.* 2012c]. The primary science payloads for this mission are an Argus infrared spectrometer for spectral analysis of surface rocks, the same type as is currently used on the CANX-2 Nanosatellite for atmospheric imaging, and a ground-penetrating radar system, which is currently under parallel development. For this mission, the Beaver will have to traverse a distance of under a kilometre while avoiding obstacles and taking sensor measurements. Naturally, extended and reliable operation on the surface of Mars would be preferred.

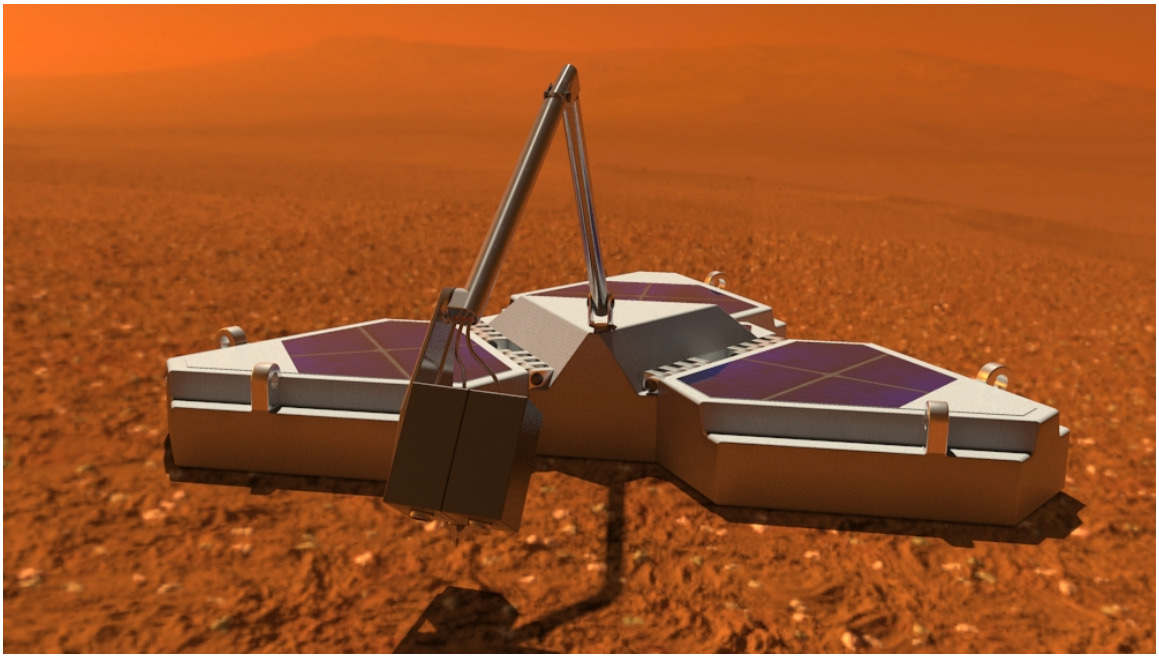


Figure 1.1: Lander module for Northern Light mission (credit: Air Whistle Media/Thoth Technology Inc.)

1.2 Research Summary

The Beaver micro-rover (μrover) designed in this work is a stand-alone, self-powered, autonomous ground roving vehicle of 6kg mass designed to gather data and perform simple tasks in distant or hostile environments such as Mars, the Moon, or here on Earth. The μrover is powered from solar panels and can recharge outdoors while operating, or by powering down onboard systems for extended periods. It has a power-efficient ARM-based onboard computer, a colour CMOS camera, magnetometer, accelerometer, GPS receiver, and IR sensor system for navigation and sensing. It communicates via ZigBee mesh networking, and can operate alone or as part of a group of μrovers. A low-cost, modular design using commercial off-the-shelf (COTS) components makes it very flexible and useable for both planetary and terrestrial research purposes [Post *et al.* 2012c]. The autonomy algorithms are probabilistic in nature, using adaptive Kalman filtering [Li *et al.* 2013d] and Bayesian networks [Post *et al.* 2012b] to handle uncertainty in the environment with minimal computation requirements.

Through work with the York University Rover Team (YURT) [Post & Lee 2011] as both a student [Post & Lee 2009] and a technical adviser [Bailey *et al.* 2012], the York University Space Engineering Laboratory [Li *et al.* 2013a] [Li *et al.* 2012c], and other students and faculty at York University, opportunities to share knowledge and test out new ideas and functioning hardware in the university environment [Post *et al.* 2012a] have helped considerably to move this research forward. Concurrent work on onboard computing, attitude control systems [Li *et al.* 2012b] [Li *et al.* 2013b] and nonlinear control for actuators [Li *et al.* 2013a] [Li *et al.* 2012a] [Li *et al.* 2013c], sun sensor systems [M.A.Post *et al.* 2013] for CubeSat-class nanosatellites by using similar or identical technology has provided many hardware and software components essential to μrover operation, and has inspired the creation of a modular architecture for research-oriented space hardware on rovers [Post *et al.* 2011] and satellites [Lee *et al.* 2012].

1.2.1 Motivations of Research

While a large number of micro-robotic systems and micro-rover prototypes have been developed for a variety of purposes, very few of them have been expressly designed with the intent of space qualification and launch. Similarly, there are numerous programming frameworks and APIs available for mobile robots, but most are either too complex and focused on general-purpose computing for use on resource-constrained embedded platforms, or inadequate for performing critical mission functions in harsh environments. In this research, we address these problems with the following goals:

1. A custom-designed electronics stack including embedded processor, radio communications, actuator drives, and sensor interfaces that can be hand-built and space-qualified

2. A reliable, scalable communications and control method useable on simple serial and SPI interfaces as well as network systems

3. An API for fixed-point math, matrix operations, and statistical calculation for use on memory-limited, embedded systems without a floating-point math unit

4. A compact, reliable and efficient framework for building Bayesian networks in a minimum of space and without dependence on external libraries not available for embedded compilation

5. A method of environmental mapping and hazard avoidance based on low-cost, commercial embedded sensors such as MEMS inertial measurement units and CMOS-based cameras.

1.2.2 Objectives of Research

The proposed research program is twofold, encompassing both the development of hardware and software for the robotic system, and the research and implementation of the intelligence required for autonomous operation. The research objectives can be briefly summarized as follows:

1. Design and construction of a complete autonomous robotic system capable of continuous independent or group operation in complex and extreme environments

2. Research and development of a distributed machine learning and knowledge system that can perform basic navigation and data gathering tasks autonomously

3. Evaluation and analysis of the combined systems' performance given a set of mission goals in a realistic outdoor environment

Successful completion of these goals will provide the Northern Light mission with a functional prototype micro-rover, and form a basis for future autonomous space micro-robotics research. The requirements for the Northern Light mission are as follows: The rover's purpose is geological surface exploration and subsurface imaging, and it must operate under its own power with a nominal range of approximately $1km$. The rover's mass should be approximately $6kg$ including payloads, and will be equipped with a visible-light camera for navigation and exploration, a point spectrometer for geological analysis, and a microscope camera for geological survey. A bottom-mounted ground-penetrating radar will explore the Martian subsurface to a depth of $20m$ to look for signs of water, and an acoustic vibrator and receiver will use sub-millisecond pulses of sound to conduct a study of the subsurface. Additionally, the rover can be equipped for immediate subsurface exploration with a rotary grinding and digging tool [Quine 2013]. The conditions on Mars are also a consideration, with an atmospheric pressure less than 1% of that on Earth and temperatures ranging from $20°C$ at the equator in summer to $-153°C$ at the poles [Quest 2013].

As simultaneous development of all of the equipment requirements for the Northern Light mission is infeasible for a single dissertation and extends outside the scope of research for the planetary rover itself, the research documented here will extend to the most critical requirements of the autonomous roving platform itself. The micro-rover's physical structure, electronic and power systems, mobility systems, navigational vision and sensing, software architecture and programming, and autonomous decision-making will be the focus of this work. The point spectrometer, optical microscope, ground-penetrating radar, acoustic sensors, and digging and abrasion tools will be considered payloads for separate development and later modular integration into this

micro-rover. The complete development of the Northern Light mission will in this way involve several research programs linked by common hardware interfaces and software architectures, and the development of the ground-penetrating radar has already been initiated in a related research program. As the mission hardware must be based on commercial components for reasons of cost and availability, the assumption will be made of a mission time during the Martian summer and a landing site near the equator with most mission operations occurring during the day so that temperatures will remain within the operating range of the rover hardware.

1.2.3 Contributions of Research

The main novel contributions of this research are as follows:

1. A modular hardware design architecture and software framework for facilitating development of open, compatible, and networkable space-qualified systems

2. A thermal vacuum tested, flexible and accessible on-board computer system with inertial sensors built from COTS hardware for micro-rover and nanosatellite applications

3. A compact, hybrid DC motor electronic drive suitable for high currents and space use

4. An embedded DSP board for monocular and stereo computer vision on mobile robots

5. Compact and efficient sun sensing methodologies for use on micro-rovers and nanosatellites

6. A stateful, robust communications protocol and datalink layer for small robotic systems that makes use of byte stuffing and static routes

7. Fixed-point mathematical matrix calculation and Bayesian network implementations in C that are compact enough for microcontroller use

8. Sigma-Point Kalman filter implementations in C that incorporate adaptive statistics and a reduced sigma point set

9. An efficient framework for programming and traversing dynamic Bayesian networks on resource-constrained systems

10. An implementation of Bayesian inference over behavioural structures for use in problem-solving on mobile robots

11. An automated methodology for generating and updating Bayesian networks for reasoning based on expert knowledge and implicit structure from a mobile robot

12. The application of generalized Bayesian networks to the Bayesian Robot Programming methodology

13. A navigational monocular vision system that combines aspects of traditional SLAM with structure-from-motion techniques and Bayesian navigation methods

1.2.4 Content Summary

This dissertation contains five chapters. Chapter 1 introduces the work and provides a background for the main topics covered. Chapter 2 details the mechanical, electronic, and software design of the Beaver μrover, as well as justification for selection of the various components. Chapter 3 provides the framework for autonomous operation, mapping, and vision processes that the μrover uses. Chapter 4 shows some results from the application and testing of the μrover hardware and software, as well as environmental testing, and provides discussion of their validity and significance. Chapter 5 concludes the dissertation. A diagram showing the chapters of this dissertation and their key contents and contributions is shown in Figure 1.2.

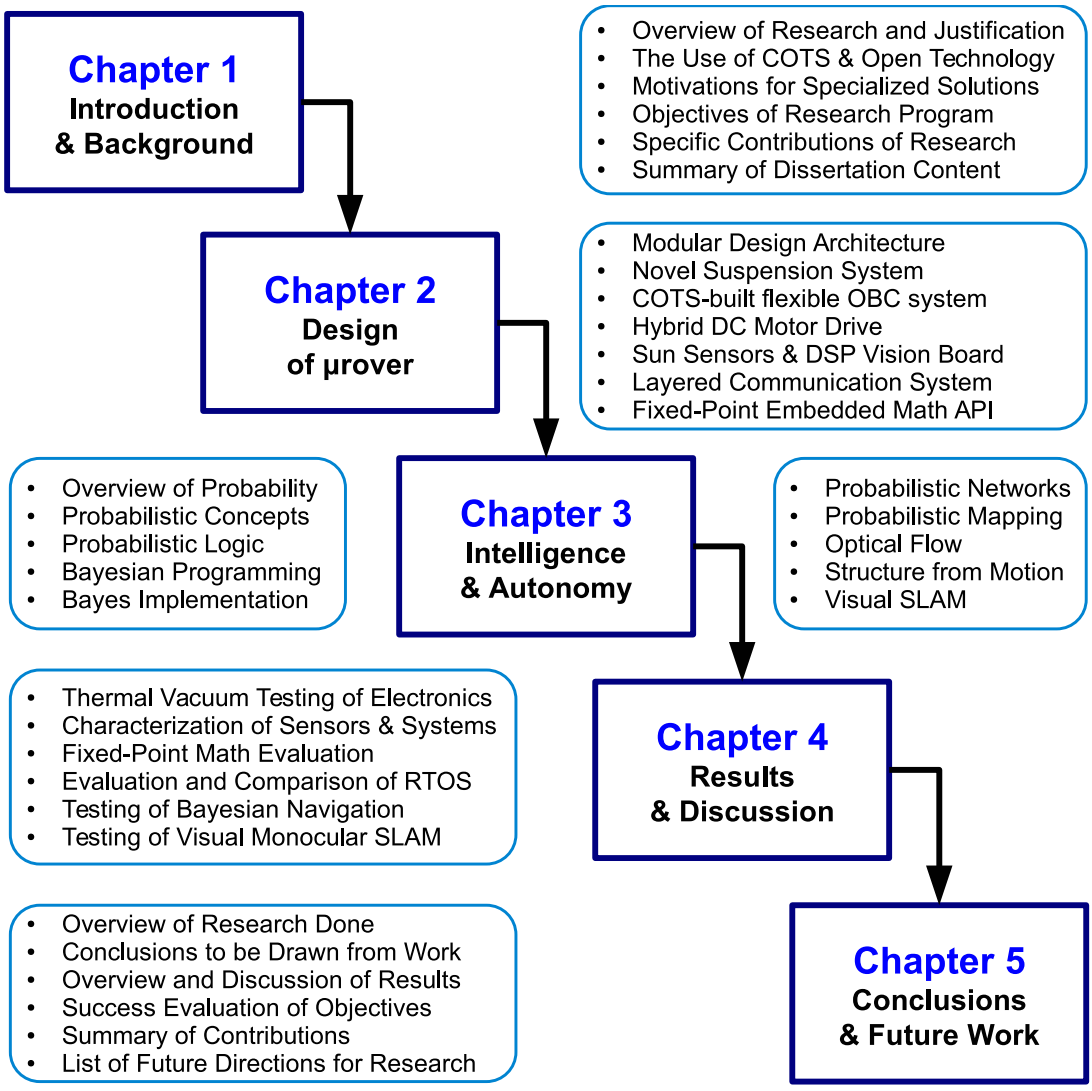


Figure 1.2: Organization of Dissertation with Contents and Key Contributions

1.3 Autonomous Planetary Rovers

There have been many different types of autonomous and semi-autonomous planetary rovers developed for a variety of purposes. Among the most successful of these were the Mars Exploration Rovers developed by NASA and JPL, Spirit and Opportunity, that made many ground-breaking discoveries and have operated far beyond their original mission requirements [Squyres 2008]. The MER were developed based on the success of the Sojourner rover on the Mars Pathfinder mission [Bajracharya *et al.* 2008] and paved the way for the larger Mars Science Laboratory, Curiosity, which is currently operating very successfully on Mars [Grotzinger *et al.* 2013] and is the most scientifically-capable robotic lander ever built, and the only current Martian rover not powered by solar energy [Grotzinger *et al.* 2012]. These types of rovers represent the current extreme of size and complexity, and require considerable time and expense to build, transport, and operate.

Since the cost of interplanetary exploration is largely dependent on the amount of mass that must be moved from one gravity field to another, many recent projects have focused on Micro-rovers ($\lesssim 10kg$) and Nano-rovers ($\lesssim 10g$) [Wilcox 1996]. In the interest of creating lighter, simpler, more efficient, and more robust planetary robots, a variety of innovative concepts have been developed [Barlas 2004]. The Rocky 7 rover was very similar to the Sojourner rover, but used Ackerman-type steering like that on an automobile [Laubach *et al.* 1998], and the FIDO rover also followed the 6-wheeled rover model [Huntsberger *et al.* 2002]. The Micro5 rover series used a 5 wheel "Pegasus" suspension system instead, with one central wheel that supported the rover while climbing obstacles [Kubota *et al.* 2003]. Another innovative suspension design was the Shrimp, a six-wheeled rover designed by the Swiss Federal Institute of Technology which used a single cantilevered front wheel and central rocker system to climb objects up to twice its wheel diameter [Estier *et al.* 2000]. To improve stability, the Scarab rover was designed to rotate its body with active suspension for level drilling into the sides of slopes [Bartlett *et al.* 2008], and the NASA Sample Return Rover prototype had active four-wheel suspension that allowed the rover to shift its center of gravity and vary its ride height dynamically depending on the

terrain. One of the smallest prototypes was NASA's $1.3kg$ Nanorover initially designed for the Japanese MUSES-CN mission (later known as Hayabusa), a four-wheeled solar powered robot with treads angled to aid in steering and active suspension that can invert itself in case of being overturned, designed for exploration on comets or asteroids [Wilcox & Jones 2000]. What is interesting about these latter two prototypes is that a similar design for the suspension system and wheel treads was arrived at independently for the micro-rovers discussed here, except that the suspension is passively actuated and cannot invert like that of the Nanorover in the interest of minimizing complexity and power use.

The MUSES-CN Nanorover is probably the most similar prototype to the μrover in terms of design, despite the significant differences in specifications due to the extreme requirements of asteroid operations. Despite its small $14x14x6cm$ size and $2.5W$ maximum power generation, it was designed with a variety of innovative features such as electroactive polymer (EAP) wipers to clean dust off instruments and a $3000:1$ range of motor drive speeds to allow fine control in microgravity environments. While the Nanorover is more compact, efficient, and robust than the μrover due to the use of customized radiation hardened parts and a temperature qualification of $-180°C$ to $110°C$ for asteroid operations, it is also significantly less powerful both in terms of mass actuation and onboard computing power, using ten $10g$ brushless cryovac motors with $0.007Nm$ of torque and a $256:1$ gearhead for movement and a $10MHz$ Synova R3000 32-bit MIPS flight processor with $2MB$ of RAM for processing. A major driver for this design is that onboard systems must be powered directly from sunlight with no battery due to the extremely cold environment, with the system state stored in EEPROM while sleeping. Similar features to the μrover include a single visual-light camera, infrared spectrometer, optical sun sensors on each face of the Nanorover body and a laser rangefinder for navigation in place of stereo vision, as well as the similarity of the swing-arm suspension.

Many other non-wheeled rover concepts have been developed such as the legged Ambler [Bares *et al.* 1989] or with exotic mobility systems such as the Lockheed Elastic Loop Mobility System [Melzer & Swanson 1974]. For reasons of control and stability though, wheeled rovers are by far the most popular and the majority of mobile robots for planetary exploration

are wheeled, generally with between four to eight wheels attached to one to three suspension supports [Schilling & Jungius 1996]. Suspension designs modelled for rover control include the popular rocker-bogie system, the Crab8 (two-rocker) and double 4-bar linkage [Kim *et al.* 2012], and the independent rocker design which is popular for its simplicity and use of only four wheels while maintaining good stability [Reina & Foglia 2013]. The rocker-bogie design has been popularized by JPL for its stability over a variety of terrains and its ability to turn and maneuver accurately [Lindemann *et al.* 2006]. However, four wheels is the minimum generally required for stability and many designs focused on simplicity use multi-link suspension designs to support four wheels that can move up and down for stability, which can still provide good performance [Robson *et al.* 2012]. Designs can also use rocker differencing to maintain the body angle, similarly to the original rocker-bogie design, but for four wheels, such as in the ORYX prototype [Amato *et al.* 2012]. Unless all four wheels can rotate parallel to the ground for steering, some degree of wheel slip is required and included in traction modelling [Yoshida & Hamano 2002], and this is the basis for most skid-steered vehicles that lack wheel direction control. Due to the requirement of wheelslip, skid steering is not very efficient for manoeuvring, but skid steered vehicles have been comprehensively modelled [Chen *et al.* 2011] and can achieve good performance if appropriate control is used [Chen & Genta 2012].

For the Beaver μrover, we will focus on the simplest practical configuration: a four-wheeled skid-steered vehicle with motors on all wheels and fully independent sprung suspension that can raise and lower to follow terrain. To conserve power and minimize chances of failure, the suspension can be passively controlled rather than directly actuated, and can fold for storage and transit in the lander. Payload space and an enclosed chassis must also be included to support the internal systems that are needed, and the frame must be as light as possible for launch and landing. Finally, as the prototype is to be constructed with limited resources, the design should be built from commonly-available materials and form factors such as aluminum sheet and tube and be simple enough that the entire rover can be constructed without specialized tools, using only hand tools, a band saw, drill press, and grinding wheel.

1.4 Mechanical Design

1.4.1 Prototypes

Before the design of the Beaver μrover was begun, a hand-built prototype rover with simple brush DC drives, and no suspension was built to test the use of the ARM9 microcontroller, ZigBee communications, inertial sensors, solar charging methods, power conversion, and motor drive and control methodologies. Figure 1.3 shows this prototype, which was not very mobile or resilient, but was instrumental in the further development of reliable and efficient rover systems. It used a Linuxstamp OBC based on the AT91RM9200 ARM9 microcontroller and an ATMega644P AVR microcontroller for managing four Allegro A3953 H-bridges that ran four Faulhaber DC micro-motors with integrated encoders and right-angle shaft drives. The electronics included resistive current sensors with ZXCT1009 sense amplifiers on battery, 3.3V, and 5V internal rails, an HMC6352 magnetometer and ADXL330 Accelerometer, and a C328R VGA serial-interface CCD camera. Figure 1.4 shows the internal electronics of this prototype.

To develop the suspension system for use in the μrover, some initial suspension designs were considered, and the best design was chosen of a lightweight frame with hinged suspension arms supported by spring dampers that allowed four-wheel independent suspension with a minimum of moving parts and without the use of drive shafts by placing the motors adjacent to the wheels. This suspension frame was used in conjunction with the electronics of the initial prototype to confirm that the four suspension arm design did, in fact, allow stable travel on uneven surfaces and was resistant to rollover in most cases. An additional discovery was that this suspension could be passively controlled to some extent by varying the relative speeds of the wheels. Figure 1.5 shows this prototype, with the suspension frame geometry shown in Figure 1.6. Two significant drawbacks of this suspension were the weakness of the hinges for the suspension arms, and the low ground clearance of the mounting points for the spring dampers. These were addressed in the current μrover design.

Based on testing of this initial prototype, the first full μrover prototype was built from scratch still using through-hole COTS electronics technology, but with components that had been tested

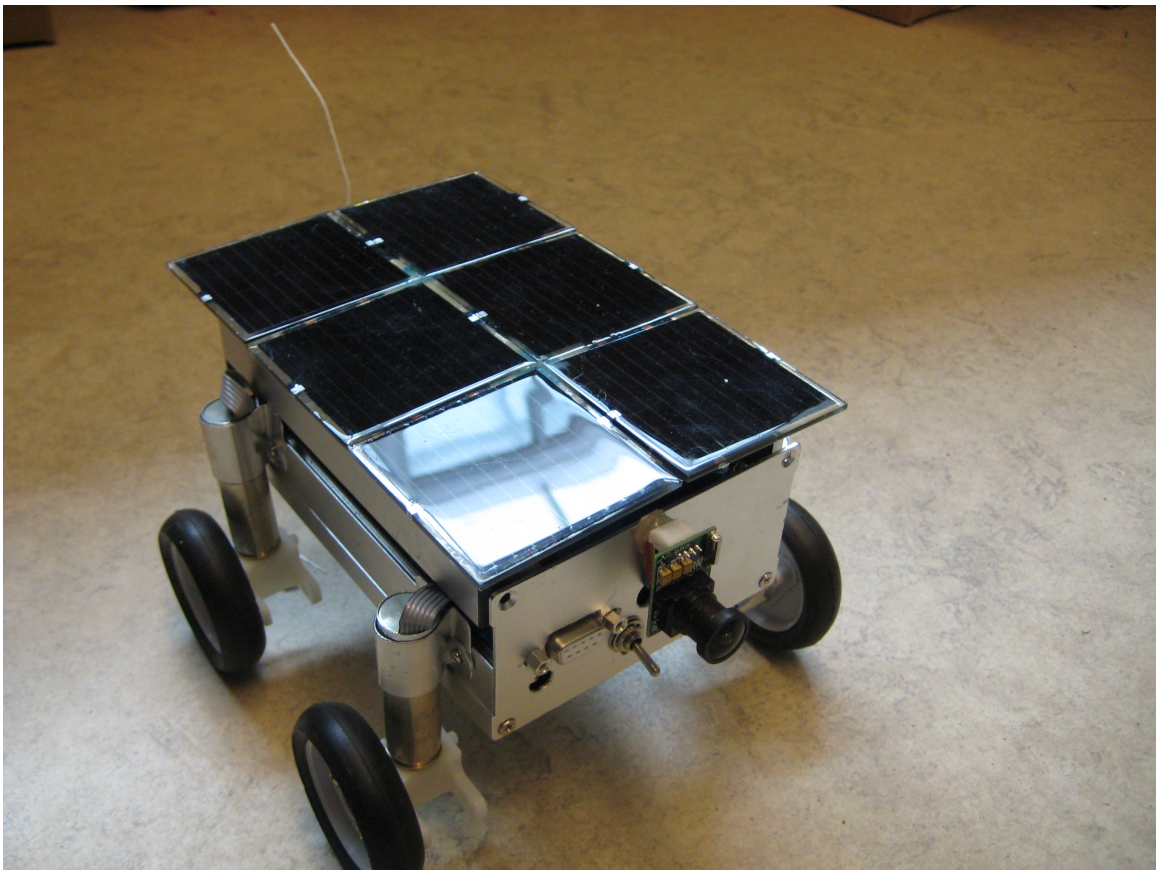


Figure 1.3: Initial μrover Prototype

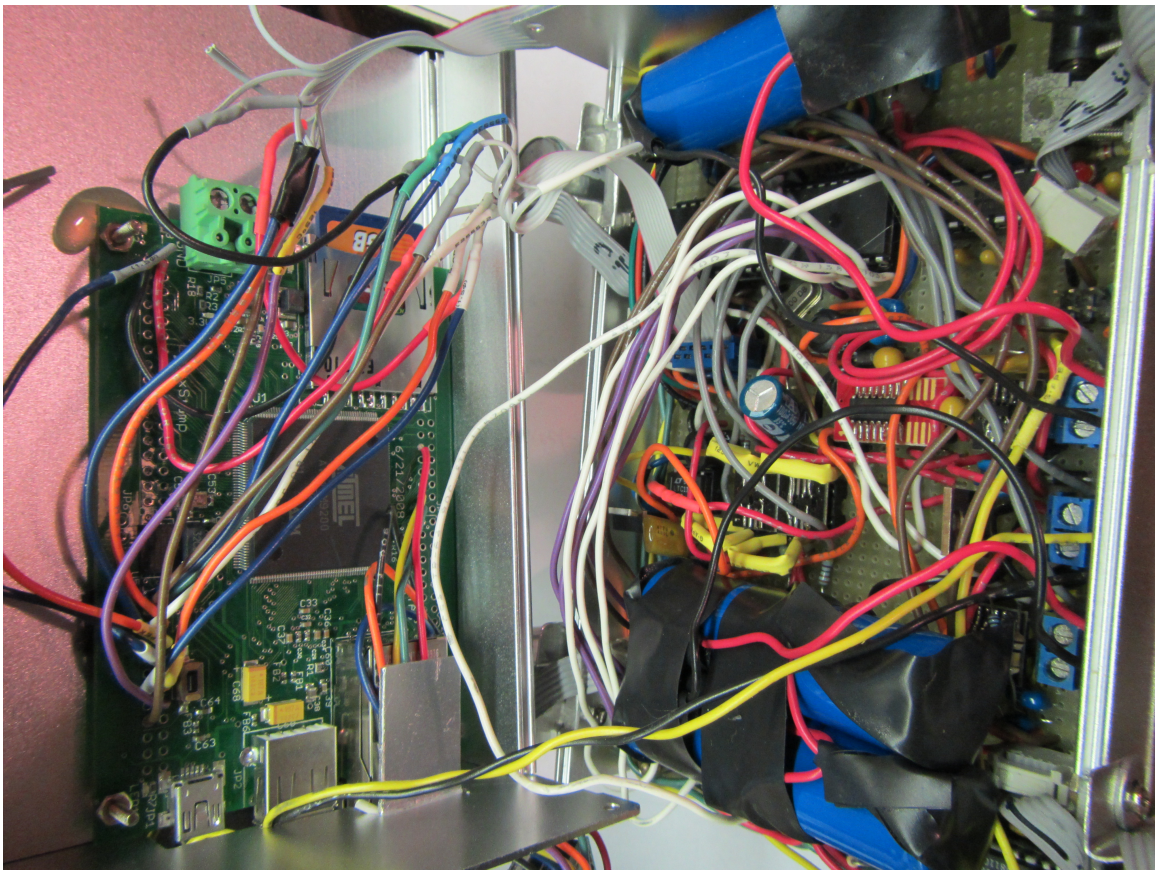


Figure 1.4: Electronics of initial μrover Prototype

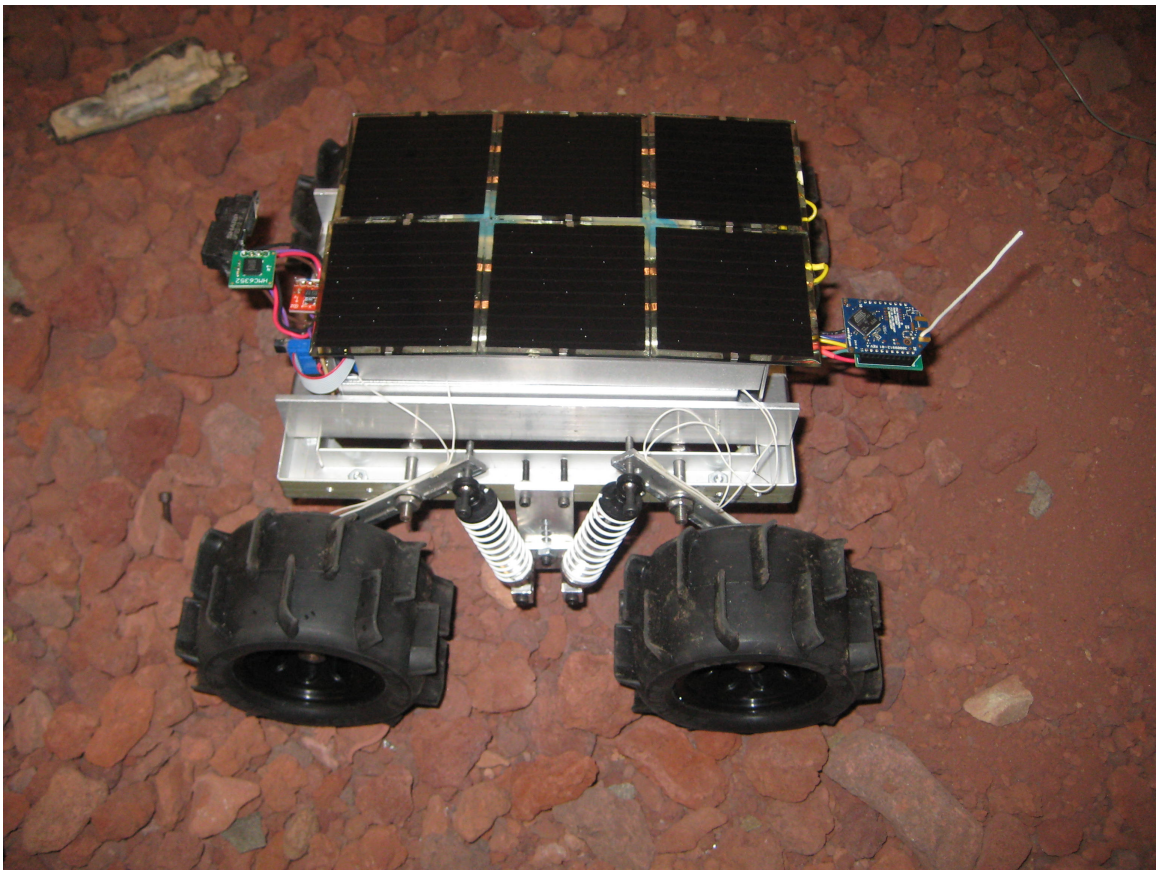


Figure 1.5: Initial μrover Prototype with test suspension

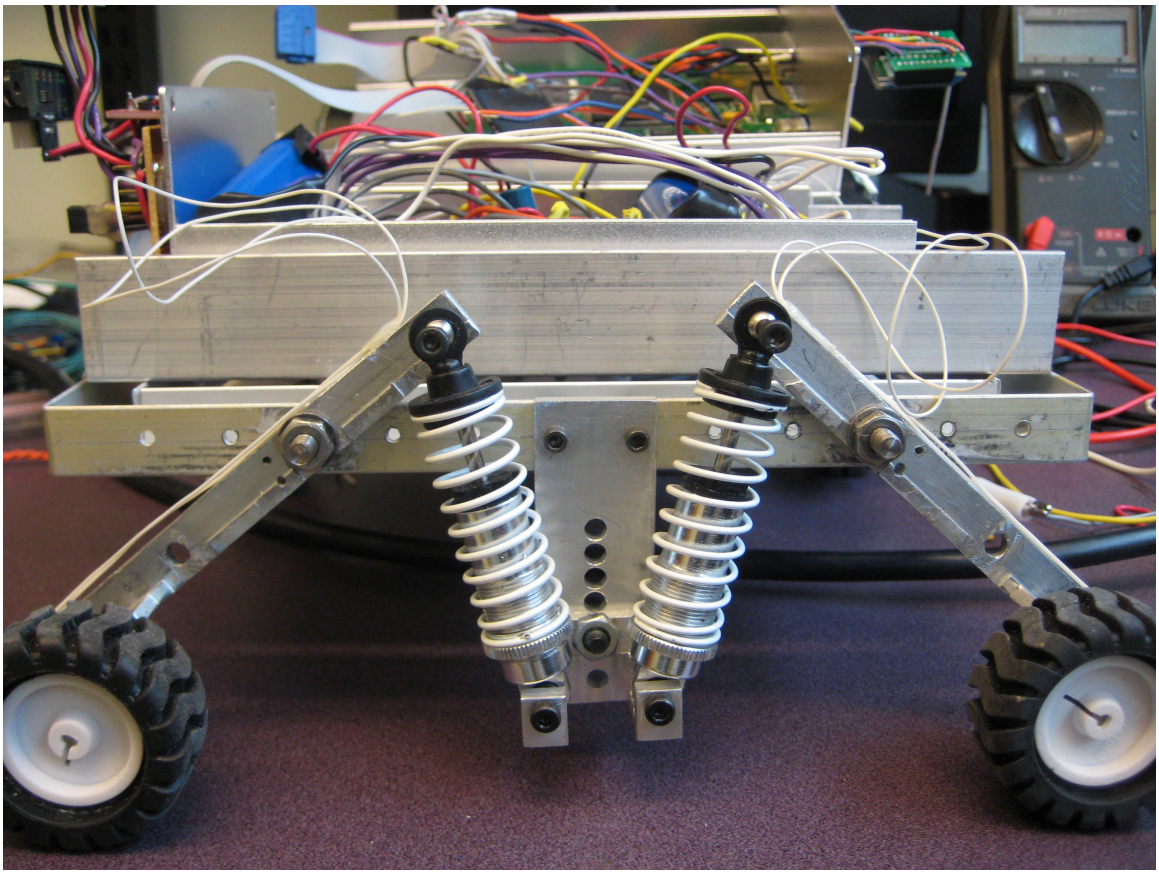


Figure 1.6: μrover Prototype Suspension Design

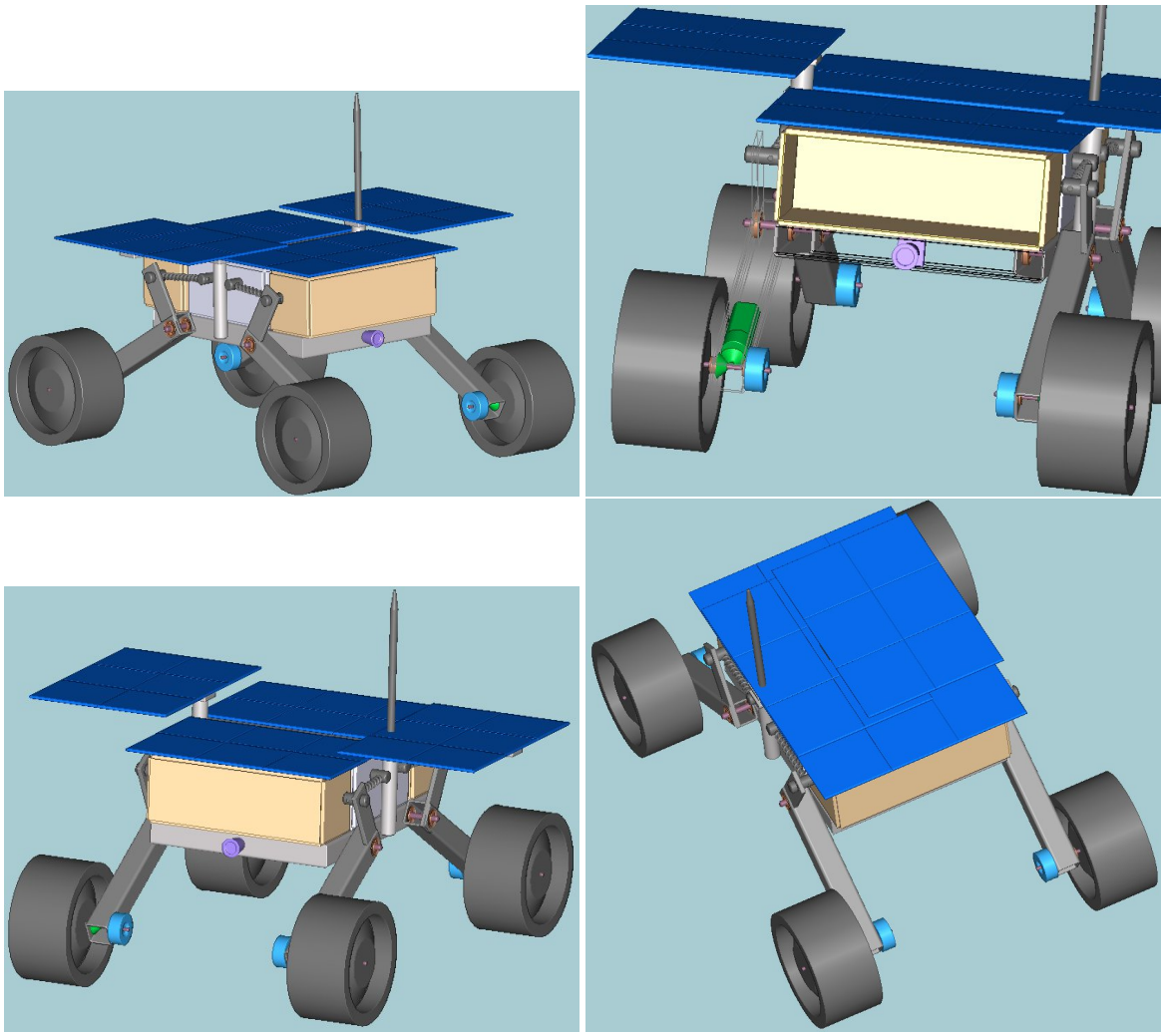


Figure 1.7: μrover Design Sketches

and proven, the capacity for operating outdoors, a full IMU and collision sensor suite, and three-phase sensored brushless motors driven by an adaptable motor drive system. A set of chassis sketches from the development phase are shown in Figure 1.7. All 3-D modelling for the μrover was done in VariCAD, which is one of a select few full-scale 3-D CAD packages that runs natively in Linux and is affordable enough for student use.

The prototype has a hand-machined chassis and suspension, has a larger solar array, and has space front and back for payloads, which was used to house the vision and IR sensor boards.

Figure 1.8: μrover Prototype Suspension Design

The motors are customized brushless DC with planetary gear-heads and integrated Hall sensors. The electronics are assembled following the PCB design to evaluate the system before custom fabrication. Figure 1.8 shows the μrover being tested at the Algonquin Radio Observatory, with the results filmed for a Daily Planet news segment on the Discovery Channel. All control testing, software development, and hardware testing detailed in this research was performed on this prototype, with improvements in hardware and functionality along the way.

Results from field-testing were generally good. Using infrared distance sensors, an IMU, GPS, and simple algorithms for intelligence and mobility control, the μrover has successfully negotiated several types of terrain, and shows promise of being a capable autonomous roving platform.

1.4.2 Chassis

The μrover chassis is designed to minimize potential failures of moving parts and mass by using a minimum of moving parts, while maximizing suspension travel and stability. The electronics enclosure and payloads are supported by a frame of $1 inch$ square 6061-T6 aluminum tube with $1/16 inch$ wall thickness, which also serves as a housing for the drive motors and a conduit for wiring. All chassis and drive components are machinable by standard shop tools, and all electronic components are both hand-solderable and proven environmentally tolerant through tests in a thermal vacuum. Four wheels are used, as more would increase mass and control complexity. Solar panels on the electronics enclosures and on folding supports at the sides are used for recharging the onboard batteries. The outer solar panels are spring-loaded and rotatable at their mount point, and the swing arms are sized so that the chassis can be stored as a flat box, and can spring up to operating position when released. It has been proven through testing that it is possible to vary the ride height and suspension geometry by varying the difference in speed between the front and rear motors, and this underactuated approach is used as part of the drive controller design.

1.4.3 Payloads

Capacity for two enclosed payloads is available in the current chassis design. Payload mass should be on the order of 1-2kg, balanced front and back, and ideally should fit in a 100mmx50mmx50mm space, although the enclosure for the payload may be adapted and enlarged as needed. Payloads planned for future development and for the Northern Light mission include an infrared spectrometer, ground-penetrating radar, and possibly a rock drill. The preferred method of payload interfacing is to use synchronous or asynchronous serial communications with flow control. A DE9 connector is used as the standard payload connector using a modified but electrically-compatible EIA-232/RS-232 pinout. The μrover will provide a female DE9 connector, deliver power through the DCD pin, provide a serial clock signal through the RI pin, and reset the payload via the DSR pin. Payloads will use a male DE9 connector, and provide

a status indicator via the DTR pin. For a more complete interface with SPI and GPIO signals, a DB25 connector can be used that is electrically compatible with the standard Centronics parallel port. In addition, a variety of system busses including RS-422, RS-485, SPI, I^2C, Ethernet and USB can be used for onboard interfacing by adding a different expansion board to the electronics stack within the main enclosure.

1.5 Electronic Design

The rapid development of low-cost microcontrollers and highly-integrated logic devices for the mobile device market has made it possible to build modular, general purpose robotic hardware for a fraction of the cost and complexity needed even two decades ago. Designing and constructing these systems in-house has the benefits of easy modification, low cost, and a better overall understanding of the system's dynamics, and students can work in depth with the system as part of their education. To leverage this capability, we propose a modular electronic design philosophy for electronic systems that can be used in a variety of research projects. The hardware should be easily adaptable to different uses. System modules should be easily field-replaceable and serviceable. Electronic parts used in this system need to be readily available for ease of development, tolerant of noise and voltage in field conditions, and inexpensive. Components in ball-grid array and other no-lead packages should be avoided to facilitate soldering and repair, and to improve vibration resistance. Programming should make use of open and freely-available tools to ensure continuing availability.

1. Modularity: Parts of the system can be added and removed as needed for the purpose at hand, but the system uses common modules and interfaces for multiple purposes.

2. Efficiency: Most space hardware is solar-powered, and most mobile robotic hardware is battery-powered, so minimizing power use and weight is essential.

3. Robustness: Components and bus systems have to tolerate environmental variations and extremes that arise from the variety of conditions that may be encountered.

4. Simplicity: For a research institution, hardware must be simple to understand and flexible in operation so that it can be applied to many different levels of projects.

1.5.1 On-Board Computing

The electronics and batteries for the μrover are entirely housed in the main enclosure, with cables run to other components on the chassis. A stack of printed circuit boards is used to interconnect the onboard electronics via pass-through connectors, as wiring and ribbon cables can prove unreliable due to vibrational fatigue. If more complexity is needed, additional stacks can be added in payload modules and linked together. A diagram of component and interface organization in the modular system is shown in Figure 1.9. The electronics stack includes a common on-board computer (OBC) motherboard with a central ARM-based microcontroller running embedded Linux for centralized control, and additional daughterboards that can be added as needed, including the drive motor controller and payload/sensor interface board. The OBC currently uses the Atmel AT91RM9200 ARM9 microcontroller for centralized processing, and was based on the open-hardware Linuxstamp 1.2 board, which was used for prototype development as shown in Figure 1.10. Daughterboards and payloads typically use small microcontrollers such as the well-supported Atmel AVR 8 and 16-bit microcontrollers, though due to the increasing capability and decreasing cost and power consumption of ARM processors, a small ARM microcontroller will likely take the place of the AVR-based boards.

Programmable logic devices such as FPGAs and CPLDs are now becoming popular for space use as the technology matures and becomes more stable. In μrover development, though, we have avoided the use of FPGA technology, since these devices often require proprietary software and hardware to program, and consume significantly more power than microcontrollers of similar capability. A typical FPGA can use on the order of $4.2mW/MHz$ and require $5V$ in most cases for operation, while integrated circuit (ASIC) microcontrollers use on the order of $5.5\mu W/MHz$ [Li *et al.* 2003]. However, FPGA devices are still considered as an option for tasks where microcontrollers are less well-suited, such as for high-speed data processing and redundant control, and the FPGA will be put into low-power mode when not in active use. FPGAs still can use up to two orders of magnitude more power while in standby than their ASIC counterparts due to the extra logic required to build the interconnecting "fabric" between logic cells, but by applying voltage scaling, power gating, and using low-leakage dynamic memory or Flash memory, it has

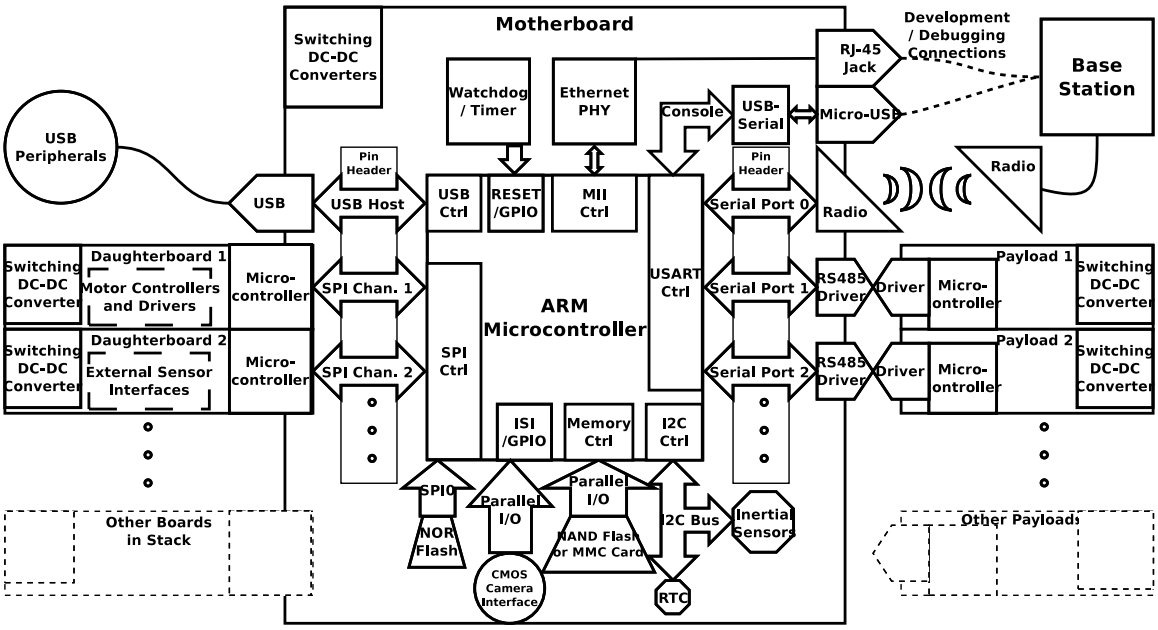


Figure 1.9: Diagram of μrover Electronic Systems

been possible in related work to reduce active power by a factor of 2 and standby power by a factor of 100 [Tuan *et al.* 2006].

Custom PCBs for the μrover were designed using CadSoft Eagle, and are still undergoing some development work. A rendering of the stack is shown in Figure 1.11. All signals are carried between OBC boards via standard 0.1*inch* pin headers. These pin layouts have been developed with the goals of breaking out every common interface available for modern embedded microcontrollers, and combining a standardized set of pinouts for these interfaces into a single header so that individual interfaces can be easily connected to external hardware. Three boards have been designed for use in the μrover so far: the ARM-based motherboard, a quad-motor control daughterboard for the brushless drive motors, and a sensor and payload board that provides extra ADC sensor hardware and mounts and buffers the payload connectors. All use external-lead packages for higher flexing and vibration tolerance, and large surface areas for better thermal coupling to the board substrate.

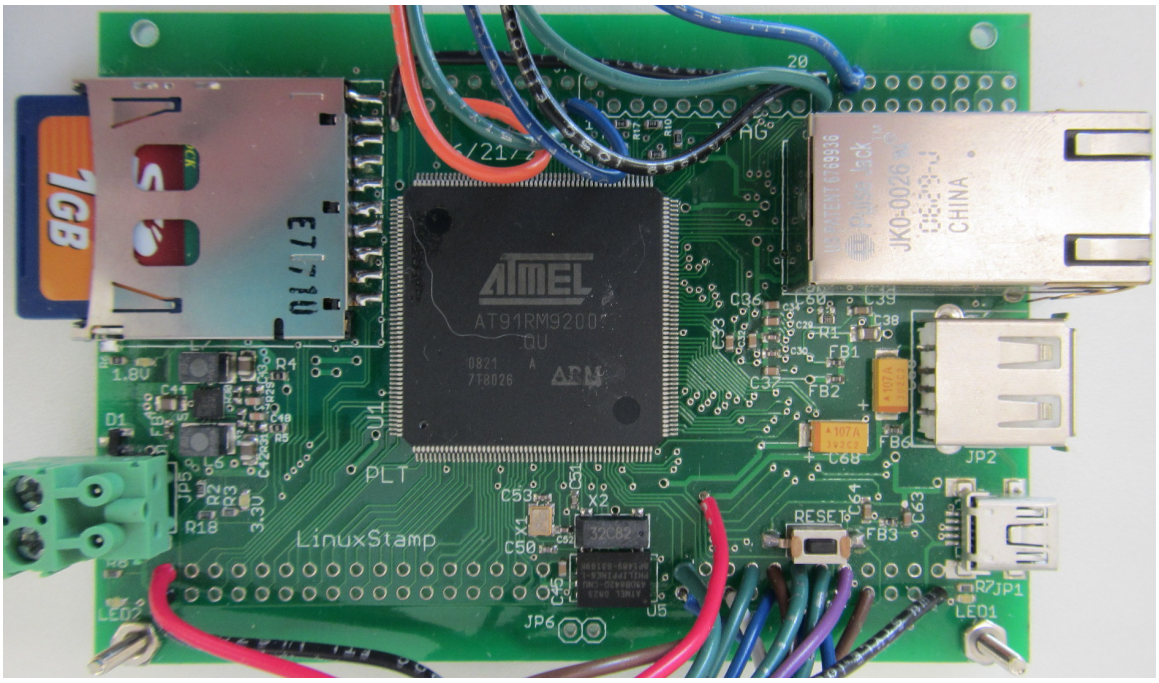


Figure 1.10: μrover prototype Linuxstamp OBC board

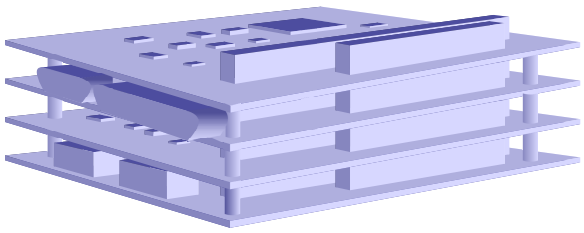


Figure 1.11: Rendering of PC/104 stack as used on μrover

1.5.2 Control Topology

The OBC stack includes a common motherboard for centralized control. Additional daughterboards can contain a set of small component microcontrollers that perform low-level tasks and can be customized to suit different requirements. To make most effective use of off-the-shelf components, the interfaces used by rover systems have to be common enough to be present on most modern embedded hardware, but still well-suited for real-time robotic applications. To maximize reliability, "multiple-hop" communications are avoided. SPI channels are connected directly to the central microcontroller and serial interfaces are connected via a MAX489 or similar serial buffer IC with high voltage and ESD protection. Damage to robotic components, which is often caused by ESD and electrical shorts, is isolated by means of the communication buffers and is less likely to spread to adjacent components.

The standardized ARM architecture is preferred for central microcontrollers as it is possible to use embedded operating systems and program source code that are easily ported between specific microcontrollers, with only the low-level hardware interfaces requiring modification in some cases. In the case of embedded Linux, these low-level interfaces are generally handled in the kernel and accessible by means of user-space device interfaces. Currently, the Atmel AT91RM9200 ARM9 microcontroller is used, though a variety of automation-oriented ARM microcontrollers are available, most notably the Cortex-M3 and M4 series that use the Cortex Microcontroller Software Interface Standard (CMSIS), and will be used in future implementations. Due to the master-slave paradigm used in the modular system, only one central ARM motherboard is usable in each OBC stack, but this limitation has not caused problems as ARM microcontrollers comparable in power to desktop computers are now available, and multiple OBC stacks can be used on a vehicle if needed. The component microcontrollers can be inexpensive 8-bit microcontrollers for motor control, sensor monitoring, and payload management, or more complex controllers if required. The microcontroller most often chosen for this role is the Atmel AVR 8-bit RISC architecture, which is easily in-system programmable using the GNU C/C++ compilers and open-hardware SPI programmers. Modularity is achieved by attaching these to the central ARM microcontroller using simple serial and parallel bus standards and pin headers, or

D-sub connectors for external connections. The use of small microcontrollers in this manner allows hardware customization without having to change the central controller in the system or its motherboard, which is often the most complex and costly component in a small robotic system.

1.5.3 Power Systems

Each module in the μrover is responsible for its own voltage conversion and uses only step-down (buck) converters to achieve system efficiencies close to 90%. Also, each module is responsible for filtering its own load noise via capacitor-inductor networks, and zener diodes should also be used to protect from ESD and over-voltages. Linear voltage regulators are unsuitable for use in vacuum or high-temperature environments as the only mechanism for cooling is direct radiation of heat [K. Sarda 2010]. To provide more power for the drive system and to be compatible with more payloads, including a ground-penetrating radar system being designed for the Northern Light mission, the power system for the μrover has been modified to supply up to $12.6V$ ($11.1V$ nominal) by using a stack of three $3.7V$ Li-Ion cells. The charging system has been redesigned to balance-charge all three cells and use one cell to power the onboard electronics. Since the solar panel area on the μrover is very limited, generating more than $12.6V$ reliably to charge the cell stack in a traditional manner is difficult and leaves little room for contingency power capacity. So it is necessary to step up the voltage provided by the solar panels in as efficient a manner as possible using the cell at the bottom of the stack as a buffer. The "charge ladder" approach, where voltage-multiplying charge pumps are used to charge each successive battery in series from the previous one is applied here as it provides higher efficiency and a simpler implementation than transformer-based solutions. The lowest battery in the stack is charged through conventional pulse charging, which has the advantage of very high efficiency and reliability.

1.5.4 Motor Drivers

An essential element of most robots is electronic motor and actuator control. Traditionally, brush DC motors with mechanical commutation have been used for robotic movement due to their cost-effectiveness and simplicity of implementation, and brush DC motors are still dominant in

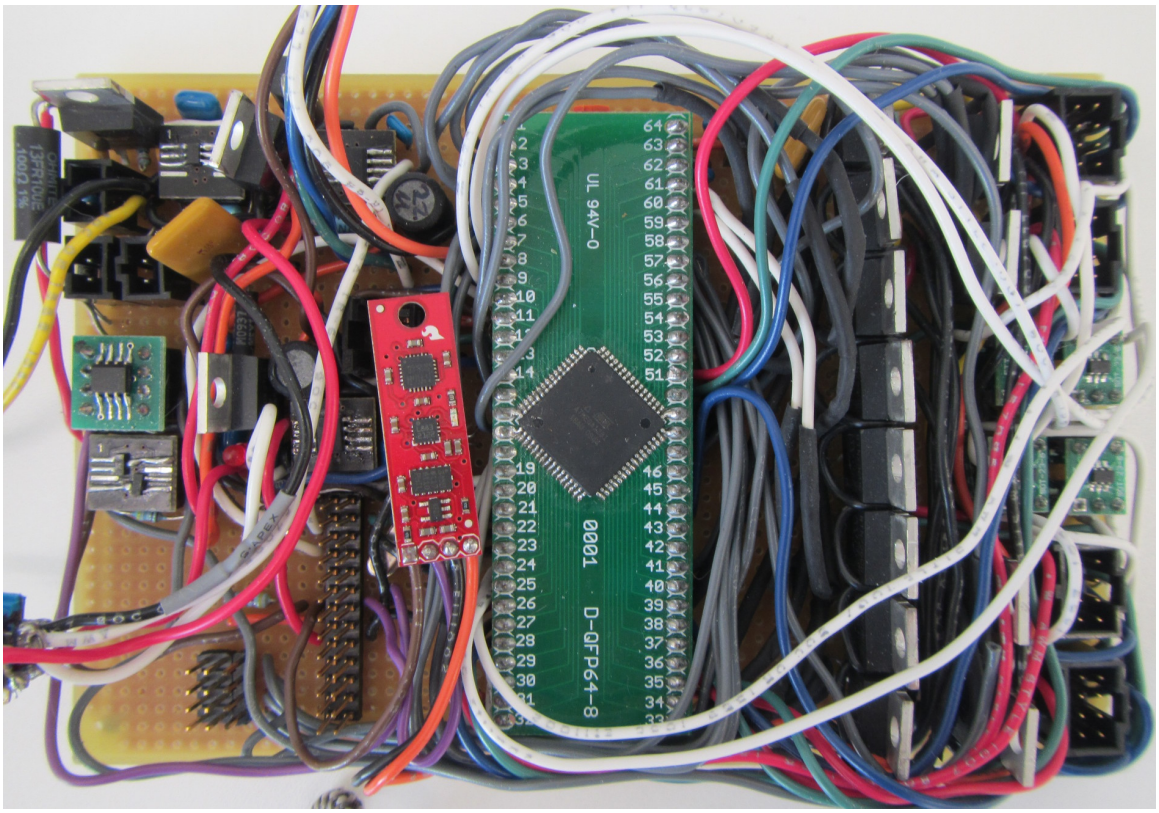


Figure 1.12: μrover prototype power electronics and drive board

the marketplace. However, the recent availability of high-speed microcontrollers and integrated drive bridges have made brushless DC motors, which require external electronic commutation, increasingly popular. Brushless DC motors generally have higher efficiency and longevity due to the lack of mechanical brushes and are preferred for hazardous environments and space applications, but require a different control method usually considered incompatible with brush DC [Lee *et al.* 2003a]. There are many brushless motor controllers on the market, but none have been found with the environmental tolerance, low-speed control, current capacity, and small size required by the μrover. Consequently, a complete motor drive system has been designed as part of the μrover, that is both power efficient and tolerant of high currents and is also capable of driving both DC and brushless DC motors. Figure 1.12 shows the hand-built prototype board for the μrover power, sensors, and hybrid motor drive systems. This board has performed very well in both outdoor testing and operation in thermal vacuum.

1.5.5 Sensors

With the number of other systems that need to be interfaced to the OBC, there is not much I/O available for sensor interfaces. For this reason, I^2C interfaces are used whenever possible for sensors that are not critical for navigation or survival. Critical sensors such as Sharp GP2Y0A02YK IR range sensors, and ZXCT1009 current sense amplifiers, and an 8-bit camera interface to an OV7670 CMOS camera module are connected directly to the OBC and the other component microcontrollers which communicate with the OBC using SPI. The IMU sensors were selected to be currently-available COTS MEMS components that operate at $3.3V$ within the range of normal operation for the μrover. An Analog Devices ADXL345 accelerometer set to the $\pm 2g$ range is used for gravity vector sensing, an InvenSense ITG-3200 three-axis MEMS Gyroscope with a maximum range of $2000^{\circ}/s$ measures angular rates, and a Honeywell HMC5883L three-axis magnetometer with a minimum resolution of $73nT$ is used for magnetic field sensing. As the I^2C bus may not operate reliably in harsh environments with long electrical path lengths, all I^2C sensors are placed within $2.5cm$ of the host microcontroller to which they are attached. Minimizing the electrical trace lengths and making the I^2C data (SDA) and clock (SCL) traces as close as possible to an equal length helps to increase reliability as well.

While on Earth we can obtain a directional bearing simply by measurement of the Earth's $25\mu T$ to $65\mu T$ magnetic field with a magnetometer and basic knowledge of magnetic declination, many other planets do not have this convenience. The planet Mars for example has very little residual magnetic field in the mean range of $24nT$ with the exception of anomalies caused by impact craters and other crust features [Lillis *et al.* 2008] and Earth's moon has virtually none except for a few crustal anomalies [Tsunakawa *et al.* 2010]. Therefore, an alternate method of obtaining headings and verifying localization information is needed. Measuring solar angles with a sun sensor is a good way of estimating absolute orientation [Volpe 1999] [Trebi-Ollennu *et al.* 2001] [Furgale *et al.* 2011]. Typical requirements include an accuracy on the order of 1 degree and a field of view of 30 degrees or 60 degrees [Maqsood & Akram 2010]. Wide-Field-of-View sun sensors [Francisco *et al.* 2012] suitable for use on micro-rover platforms are still an open area of research, and in many cases, simpler systems are desirable. Low cost sensors for research use

are usually constructed by graduate students and researchers, and must be efficient, compact in size, and robust enough to survive the space environment, which focuses on CubeSat technology development as well as a micro-rover under development for the Northern Light Mars Lander Mission. We outline the development of two coarse sun sensor methodologies that are compact and efficient enough for a CubeSat-class nanosatellite and can provide reliable solar angle information for embedded nanosatellite ADCS technology. There are several basic methodologies that are in use for sun sensors, including the use of Position Sensitive Photodiodes (PSD), linear and grid sensor arrays such as CCDs and photodiode arrays, and the measurement of sunlight on solar panels used for powering the spacecraft. We will make use of the latter two.

1.5.6 Bus and Payload Interfaces

The SPI bus is preferred for board-to-board communication at high bit rates, and communication with both boards and payloads is achieved using 8-bit serial communications. Provision is made on the board headers for at least four serial interfaces and four SPI interfaces with chip selects. As I^2C buses and devices have proven to be less reliable under extreme environmental conditions, I^2C interfaces are used only within each board for register-level IC communications between adjacent devices. For interfacing high-bandwidth devices, CMOS and CCD cameras are interfaced directly to the central ARM microcontroller via either a parallel bus of general-purpose input-output (GPIO) pins. Both volatile (RAM) and non-volatile (Flash) memory is interfaced by means of dedicated memory hardware.

For external payload communications, RS-485 has been selected as the standard of choice. The RS-485 interface is an industry standard for differential-pair serial communications to multiple transceivers, and is already used on many robotic systems. Full-duplex with a single master at 115200*baud* is preferred to minimize payload speeds and avoid the need for bus arbitration. RS-422 devices are directly compatible, and RS-232 is compatible either with an adapter or by using TX- to RX and RX- to TX with RX+ and TX+ to GND. The external interfaces are situated on a board in the OBC stack with line drivers and external DE9 connectors. Up to 32 devices on a single connector can be used. To pass GPIO signals and parallel buses, DB25 connectors

are useful with each pin buffered against ESD and high voltage also. Ethernet and USB are also available on the OBC, but have significant programming and hardware overhead, so they are only used for debugging and testing.

1.5.7 Radio Communications

To provide a long-range serial mesh networking system, the Digi XBee PRO Digimesh 900 serial radio module is used. The module has a well-known form factor, so replacing them with other modules is simple. For higher-power radio systems, an OBC daughterboard can be used to incorporate a wide variety of radio systems, including S-band and UHF radio systems for space hardware.

1.6 Software Implementation

Historically, much of the reliability-critical programming for space and aerospace vehicles has been done in bare assembly language or dedicated languages for the target system. Just as circuits were hand-built one at a time, programs were written at one time for one purpose on one vehicle. For the relatively simple systems of the 1960s, 70s, and 80s this was tractable, but as embedded systems rapidly became faster, cheaper, and more ubiquitous and more complex programs for autonomous vehicles became necessary, there was a steady progression of thought away from deterministic structures and fixed scheduling and toward high-level, dynamic, and reusable programming. The level of complexity for programming the Mars Science Laboratory required tolerance of idiosyncratic behaviour at many levels for its four million lines of code [Manning 2012]. At these scales, the policy of low-level programming and absolute determinism for space vehicles becomes completely intractable, which necessitates the adoption of not only general-purpose languages such as C for portability and code re-use, but also multi-tasking operating systems such as Linux and the associated overhead and execution uncertainty. This is offset to some extent by the high speed and storage capacity of modern embedded systems, and the efforts to increase reliability by other more portable and reusable means such as redundancy, run-time error checking, and fault detection. We therefore can afford to approach the development of a planetary rover by way of COTS hardware with embedded operating systems and shared libraries, as most terrestrial robots are built.

In order to benefit from and support the open-source community, and to ensure that the basis for the system remains freely available and up-to-date, open-source OS software and the GNU compiler collection (GCC) is preferred to form the software base for the μrover. It is assumed that most mobile platform software will have to control actuators, read sensors, and communicate with a base station, a set of central control programs for autonomous operation, or both with system coordination occurring on the central ARM microcontroller. The YURT rover [Post & Lee 2011] and micro-rover prototype are good examples of this kind of a system, as is the example is given in [Marosy *et al.* 2009]. As such, a robust multitasking framework and a

routing and command handling system are essential to reliable operation. The original software architecture for the μrover is shown in Figure 1.13, and it has stayed relatively consistent with respect to the programming involved with the exception of the Bayesian system implementation, described in detail in a later chapter.

The software is stored in NAND flash memory on the microcontrollers and on redundant NAND and NOR flash devices on the motherboard. NAND flash is the most common and provides large storage capacity, while NOR flash devices can store only a few megabytes but may be more resistant to radiation and corruption [Farokh Irom 2008]. To mitigate the risks of software corruption, in the event of a boot failure the bootloader for the central ARM microcontroller will switch between two kernel/filesystem images. It is possible for the central ARM microcontroller to boot from the smaller failover device and retain sufficient capacity to reprogram internal component microcontrollers as required using SPI. Implementation of error-correction codes is also proposed for improving storage reliability, though it would have to be done either in software or on an external logic device such as a programmable logic device (PLD). For NAND flash devices, the recent addition of UBIFS to the Linux kernel provides error detection and wear-levelling for NAND devices, and currently appears to be the best option for NAND devices in Linux.

1.6.1 Operating Systems

Direct-to-hardware programming on the central ARM microcontroller is possible, but an embedded operating system (OS) is usually used in research applications to expedite software development and simplify hardware interfacing. The Emdebian Linux OS has been used for most of the existing development work, and other FOSS systems are being investigated for real-time flight hardware implementations such as Linux with Xenomai for real-time support and RTEMS as a standalone real-time framework.

Customized Linux kernel patches and drivers that implement user-space support for the embedded OBC peripherals such as the RS-485 interfaces, SPI controllers, and I^2C sensors have been developed to provide hardware integration, and a set of dedicated libraries for 8-bit AVR microcontrollers has been created to assist in programming component applications for this sys-

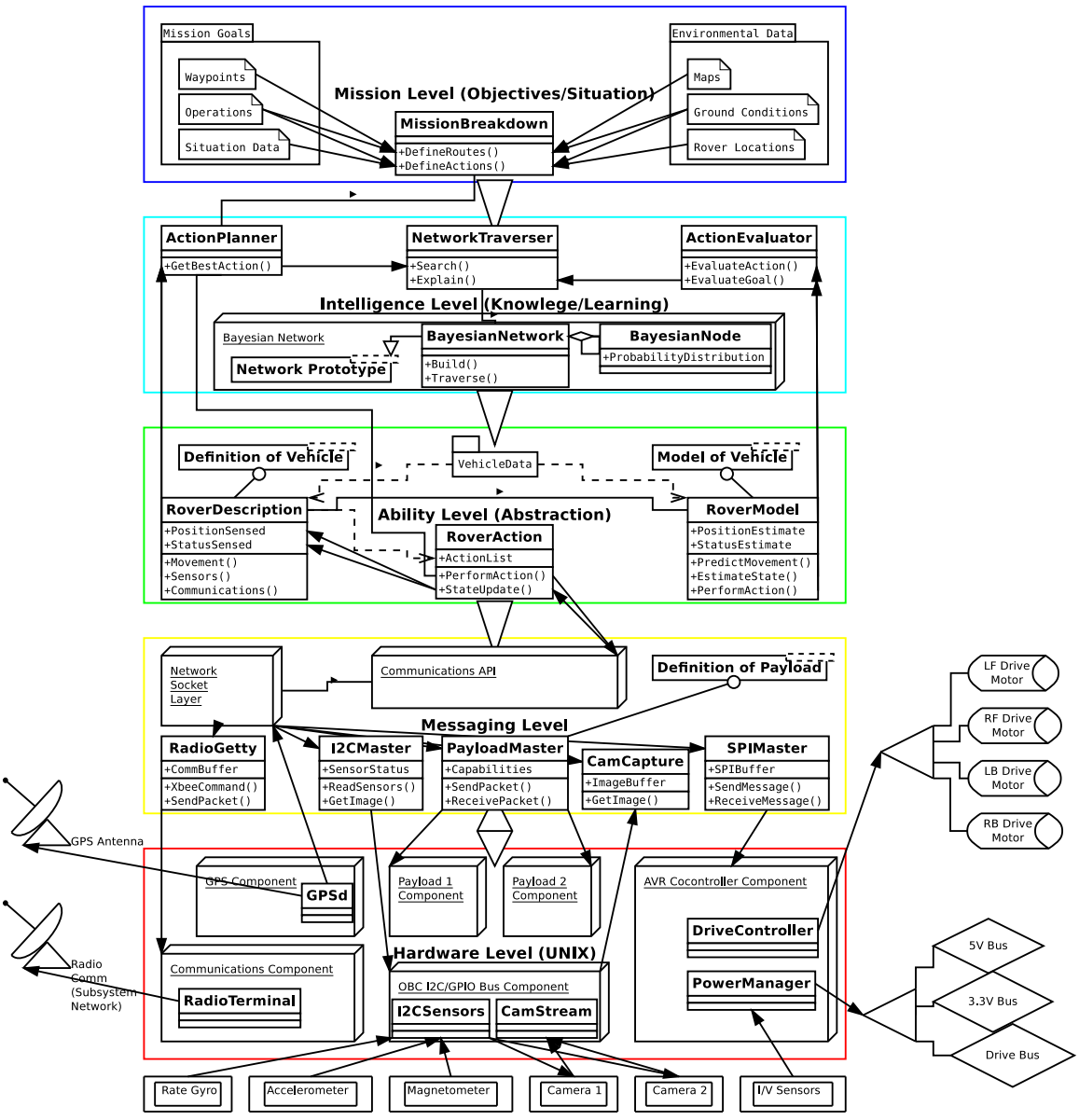


Figure 1.13: Original software diagram for μrover

tem. The software for onboard systems has been developed entirely in C and C++ for efficiency and minimal code size. Other languages can be run on resource-constrained hardware such as real-time Java, and using a Linux environment enables the use of Python as well. These will be retained as options for future research.

For overall control and monitoring of remote systems at the base station, a graphic user interface (GUI) has been developed using Java, with the philosophy that it must be portable to whatever OS is used at the base station, but need not run on resource-constrained embedded hardware. The GUI development is based on the work of the York University Rover Team, who have been steadily improving their base-station GUI concept for each generation of remotely-controlled rover. The GUI integrates system health monitoring, GPS localization, joystick movement control, and camera displays for convenient use of the system operator.

1.6.2 Operating System Programming

The choice of operating system is important to the reliability, flexibility, and ease of development for the μrover. While simple systems can work effectively with single-threaded, bare-hardware computing platforms, it is assumed by design that at least one on-board microcontroller will be capable of running a multi-threaded OS. The ARM port of the Linux OS has been used throughout most of the development process, as Linux has the most community support and cross-compatibility of the free operating systems. However, the use of a hard real-time operating system for flight hardware is preferred for reasons of better reliability and real-time task response in managing hardware. Several candidates have been evaluated and compared in two categories: dedicated real-time POSIX operating systems for the highest reliability and performance, and real-time Linux kernels, which offer better real-time performance with very few changes to the Linux system that is being currently used.

To minimize resource use and ensure that the μrover maintains compatibility between the operating system, the low-level interface components, and the mission-specific software, some measure of standardization for the languages used for programming must be made. In general, it is important to use a low-level, portable, efficient language for embedded systems, but robotic

programming on modern platforms is done using many different languages. Usually a tradeoff is encountered between languages with simplicity and high performance, and languages that are easy to program in and flexible.

Embedded systems are also often limited in processing capability by the lack of a Floating-point Processing Unit (FPU), which increases the complexity and power draw of the processor. For this reason, it is desirable to implement as much functionality as possible in fixed-point arithmetic to prevent the compiler from the slower, less efficient and sometimes unreliable use of software routines for handling floating-point calculations. For this purpose, we have implemented a general-purpose scalar and matrix math library that uses limited-precision fixed-point calculations based on integer calculation hardware. It is used for accelerating calculations for Kalman filtering, Bayesian inference, and to allow complex calculations on floating-point incapable hardware such as the Atmel AVR microcontroller.

1.6.3 Communications and Integration

In a complex, networked system such as a group of micro-rovers, an organized, point-to-point messaging system is necessary so that control, telemetry, and state information can be delivered to where it is needed. Originally, the OpenJAUS 3.3 messaging API was used to send standardized messages between rovers and between components [Group 2007]. However, with the release of 4.0 (which conforms to the SAE JAUS standards), OpenJAUS was released under very restrictive license terms, such as that the API could not be used without an explicit time-limited license, and that all changes to the source code have to be submitted back to OpenJAUS with suitable conformance. The independent JAUS++ API remains open-source, but is not designed for deeply embedded use, and will not even build without desktop-related dependencies such as X11 support. Combined with the closed development of SAE standards and the cost overhead of the standards themselves, this effectively renders the public JAUS APIs unusable for an open, independent development project.

In contrast to the limited number of public JAUS implementations in use, the Robot Operating System ROS has enjoyed great popularity, partly due to its adoption by companies such as

Clearpath Robotics and notable events such as the DARPA robotics challenge. ROS is not an operating system, but rather a suite of interconnected processes running on a host OS such as Linux. Both ROS and JAUS share many common paradigms, such as arbitrary message routing, establishment of service connections and data streaming, and capability/status reporting on each communications endpoint. As an open-source project, it is designed to cater to as many different systems and languages as possible, with bindings for C++, Python, Octave, Lisp, and others. It also uses XML-RPC for connection negotiation and configuration, and messages are written in a dedicated interface definition language for compatibility [Quigley *et al.* 2009]. The main drawback of ROS is that it still requires an OS and filesystem, and development of μrover software is aimed to eventually obviate these on actual space hardware. Some aspects of ROS such as the use of nodes and separate processes are used as inspiration for the μrover messaging and control systems, and some measure of compatibility with ROS may be possible in the future. However, ROS is designed for ease of use rather than efficiency and reliability, and introduces significant complexity and overhead into the component control and messaging process, which is undesirable in embedded space systems.

Other architectures for robot control surveyed include the Player/Stage project [Vaughan & Gerkey 2007], which is a robot control and simulation system similar to, and often used in, ROS systems as well, and shares both its easy-to-use methodology of TCP/IP interprocess communications and much of its extra complexity, but has not been maintained as consistently as a separate project since the popularization of ROS. The Orocos Project is a set of libraries for robotics and automation that includes kinematics and dynamics, Bayesian filtering, and a toolchain for real-time embedded software compilation, and represents the best available alternative to ROS in terms of scope and complexity [Bruyninckx 2001]. The main reasons for not adopting Orocos are again the need for high integration and reliability on space hardware platforms, and the requirement for support of low-level hardware and communications on non-UNIX platforms. However, this does not preclude making use of the extensive Orocos libraries at some future time for higher-level programming if needed, and Orocos remains a useful resource of such code. Similarly, the Mobile Robot Programming Toolkit is a set of C++ libraries for robot

programming and control that can provide useful resources, but is not targeted at space hardware [Claraco 2008]. In contrast, CLARAty (Coupled Layer Architecture for Robotic Autonomy) is a robotic system that is expressly designed for space hardware use and has been applied to the control of the Rocky 7 and 8, FIDO, K9, and MAX rovers from NASA and the Jet Propulsion Laboratory [Volpe *et al.* 2000]. While an exemplar of space robotics software, the control of source code by NASA and limited availability of some restricted libraries make CLARAty more appropriate as a model for design rather than as a complete solution. The FINROC real-time framework was designed to address some of the problems with high complexity, computational requirements, and loose coupling between components in the above approaches [Reichardt *et al.* 2013], but is still in the process of maturing, and some differences in implementation requirements with the μrover exist also. An XML standard for modelling and interoperation of robots known as RobotML has been proposed, and provides a useful common platform for robot-related information [Dhouib *et al.* 2012]. As of 2013, very little support information is available for the language and its state of activity is uncertain, but a RobotML implementation for common communications may be possible in the future. Additional application-specific frameworks are the OpenRAVE virtual environment for motion control [Diankov & Kuffner 2008], the USARSim robot simulation system [Carpin *et al.* 2007], and modelling systems such as the Rover Chassis Evaluation Tools (RCET) and Rover Performance Evaluation Tools (RPET) [Ding *et al.* 2011], which are useful but out of scope for current μrover development.

The μrover messaging and execution system has been developed using a similar component and routing model to JAUS and with the component messaging paradigm of ROS, but with different data link and framing layers designed specifically for small embedded robots. It should be noted that this technically could still render the implementation ”JAUS compliant“ from an architecture standpoint. Each separate hardware device driver, autonomous process, or data node operates as a separate process on a multi-tasking OS, specifically Linux or a dedicated multi-threaded RTOS. Components run by microcontrollers that are too small to support a multi-threaded OS typically operate as only one process and are considered as such by the communications system. In the course of previous work with the York University Rover Team, the problem of reliable

point-to-point inter-process and inter-vehicle messaging has been investigated in detail. Physical layers are mainly Ethernet and RS-485 serial, so historically command packets have been forwarded from the base station ("user") to the rover components ("clients") via several intermediate hops, such as the on-board computer ("host"). To obtain a reliable, low-latency, low-bandwidth, and maintainable link, variable-length binary packets with a common header and set of operational codes are used. To prevent redundant commands from overloading serial communication links, a model of all the pertinent state variables is kept on each link in the chain of communications, updated and checked at regular intervals to detect communications failures, and variables are only forwarded to the relevant components at a speed they can handle.

1.6.4 Embedded Kalman Filtering

All sensor systems on the μrover have some amount of noise or systematic inaccuracy associated with them. Given that most of the processes that need to be estimated (such as inertial measurements and actuator states) are well characterized in terms of their inputs, using a predictive filter such as a Kalman filter is appropriate. To obtain reliable values for state estimation of the μrover in real time, it is desired to use a nonlinear filter that is simple to implement algorithmically (such as being derivative-free), computationally efficient, and provides good performance. Traditional Kalman filters, as first formally described as an algorithm by Rudolf E. Kalman, are also known as Linear Quadratic Estimators (LQE) and are predictive Bayesian filters that use calculated predictions of the system state and the inputs to the system to produce a statistically optimal estimate of the system state in the next time step. This is accomplished by updating the estimated system state by a weighted average derived from the estimated means and covariances of the state variables under a Bayesian assumption [Welch & Bishop 1995]. The Kalman filter can be considered to be a Bayesian filter similar to a Hidden Markov Model, as it is built on the Markov assumption that only the immediately preceding state is necessary to determine the current state of the system. However, the Kalman filter also assumes a continuous state space for all variables, and generally a Gaussian distribution model. Figure 1.14 shows the Kalman filter model visualized as a Bayesian network. In this network, a time series of states $x(t)$ are assumed

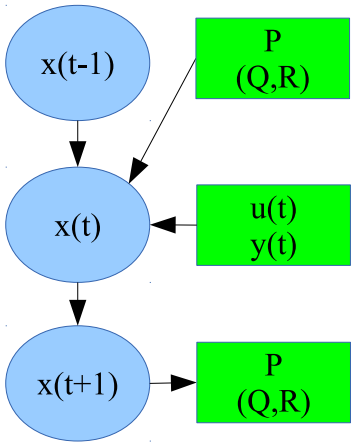


Figure 1.14: Kalman filter represented as a Bayesian network

to be statistically dependant on a state covariance P, are affected by known control inputs $u(t)$ with added state noise covariance Q, and are sensed by sensor outputs $y(t)$ with additive sensor noise covariance R. These statistics are used to predict the next state estimate $x(t+1)$ and correct the sensor measurements of that state, by which process P is updated, and potentially Q or R also.

If the means and covariances of the state variables in the system are known and an accurate linear model is used, the covariance of the output can be minimized without any assumption about the actual statistical form of the noise in the system, but an exact probability estimate will be obtained if the system noise is Gaussian [Kalman 1960]. As the original Kalman filter (and its continuous-time equivalent, the Kalman-Bucy filter) was limited to linear systems (using a linear system of equations as a model), effort was very quickly made to build nonlinear predictive filters with the same qualities, particularly for aerospace use. The Extended Kalman Filter (EKF) addresses the linearity constraint by linearizing the nonlinear model about the state estimate. Jacobians are used to estimate the gradients of the system and measurement models near the state estimate so that the filter can operate in much the same way as a linear Kalman filter [Ribeiro 2004]. However, this approximation makes the performance of the EKF very dependent on the model and linearization that is used, and accurately estimating high-order systems is

difficult, in addition to the complexity of obtaining linearizations of the system model.

A significant amount of work was directed toward overcoming the limitations of the EKF when estimating highly nonlinear systems. The key to progress in this area was the eventual realization that it was not necessary to propagate and estimate directly using a linear model. Rather, *only the statistics* of the process in question had to be propagated for estimation to work. The use of a Monte Carlo method with ensembles of pseudo-random state vectors to represent state uncertainty was suggested by Evensen, while reducing the number of state vectors that had to be calculated [Evensen 1994]. Independently, Quine, Uhlmann, and Durrant-Whyte developed a method to approximate state uncertainty for an n dimensional state vector using $2n$ separate vectors and a mean for propagating covariance through the estimator [Quine *et al.* 1995]. The ensemble approach is effectively equivalent to the EKF, as it propagates the first two moments of a state distribution, but does not require the calculation of Jacobians, and was formalized by Quine [Quine 2006]. A similar parameterization by Julier and Uhlmann used a set of $2n + 1$ discretely sampled points to represent the global mean and covariances for n state variables and a method called the *unscented transform*. Central to this method is the assumption of Gaussian statistics, so that only a mean $\bar{x}$ and two outer points χ known as *sigma points* are necessary to completely describe a Gaussian distribution, obviating the need for randomly-sampled test points as in Monte Carlo methods. These test points are calculated based on the statistics of the prior distribution, propagated through the state and measurement models, and used to reconstruct the posterior distribution directly, without the need for gradient approximations, and accuracy over the EKF increases with the nonlinearity of the model [Orderud 2005]. This filter is known as the Unscented Kalman Filter (UKF), and has formed the basis for most of the Kalman filter development in recent years of the larger class of sigma-point Kalman filters (SPKF), named for the test points that are propagated. A diagram of the calculation flow through the UKF is shown in Figure 1.15.

Many different variants of sigma-point Kalman filters have been developed, but we will summarize here only the variants that have demonstrated specific advantages over the UKF. Although the name “sigma” may cause one to assume that they are located at one standard deviation from

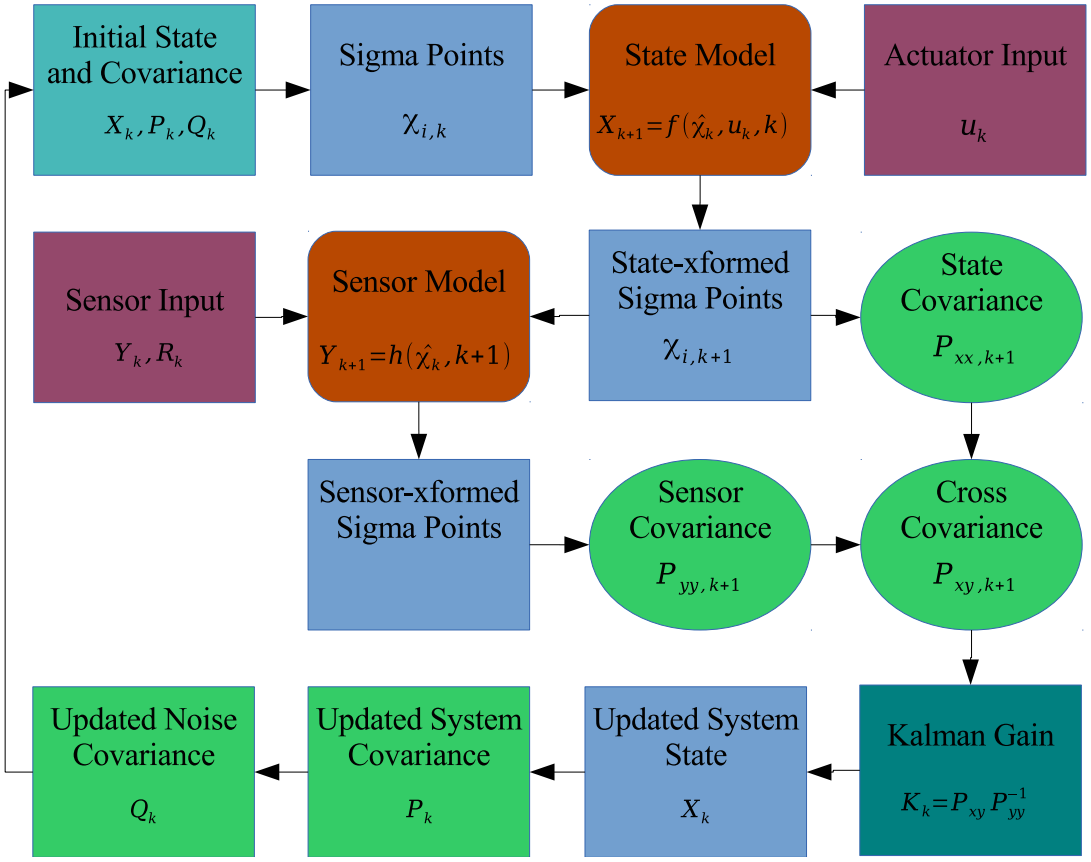


Figure 1.15: Kalman filter operational diagram

the mean, this is generally not the case as the locations and number of sigma points varies depending on the filter design, and optimal choice is sometimes a great source of debate. The Square-Root Unscented Kalman Filter (SRUKF) avoids repeated computation for the Cholesky factorization (as a matrix square-root) of the system covariance matrix by propagating the square root of the matrix only and updating it separately with each iteration using an algorithm commonly known as "cholupdate" to achieve up to 20% performance improvement over the UKF [Van der Merwe & Wan 2001]. The Scaled Unscented Transform Kalman Filter (SUKF) uses a scaled version of the Unscented Transform to allow sigma points to be scaled to an arbitrary dimension for efficiency in conversion of different number systems, but otherwise operates similarly to the UKF [Julier 2002]. As Kalman filters traditionally use fixed estimates of the state noise $\mathbf{Q}$ and the measurement noise $\mathbf{R}$, another improvement over the conventional Kalman filtering method is to adapt these noise covariance matrices based on the residual (error) from the measurement estimate. This is known as an Adaptive UKF (AUKF). To achieve even better results in embedded systems, Grewal's SigmaRho filter (SRF) implements square-root propagation, numerical scaling, and adaptive statistics to improve performance [Grewal & Kain 2010]. As formal implementation is more complex than other filter types and performance is still comparable to a modified UKF, implementation of a SigmaRho filter will be left for future work. To further reduce the computational load of an SPKF, one of the most straightforward methods is to reduce the number of sigma points that must be propagated. The Spherical Simplex Unscented Kalman Filter (SSUKF) was one of the first implementations to estimate posterior probability distributions by using a reduced sigma set. The SSUKF uses for n dimensions only $n + 2$ sigma points weighted proportionally to $1/n$, of which $n + 1$ lie on a hypersphere with radius proportional to $\sqrt{n}$ [Julier 2003].

Finally, two of the most recent major innovations in SPKF research were made by Arasaratnam and Haykin. First, by linearizing the system and measurement models with statistical linear regression though a set of Gauss-Hermite quadrature points, the Quadrature Kalman Filter (QKF) presents an alternative method of parameterizing Gaussian distributions [Arasaratnam *et al.* 2007], and can be extended like the UKF to propagate square-root statis-

tics [Arasaratnam & Haykin 2008] but suffers from exponential complexity increase with the size of the state space [Closas & Fernandez-Prades 2010]. Second and more importantly, by explicitly assuming all statistics to be Gaussian and applying a third-degree spherical-radial cubature rule akin to those used for numerical integration of multi-dimensional integrals, a different set of sigma points can be obtained that offer a better statistical approximation of high-order nonlinear functions. This is effectively an application of linear estimation theory to nonlinear filtering, and is called the Cubature Kalman Filter (CKF) [Arasaratnam & Haykin 2009]. Being Gaussian in nature, the CKF requires $2n$ sigma points that are distributed on a sphere for n states without the use of a mean, and is comparable in computational complexity and performance to the UKF. In fact, the CKF process reduces to that of the UKF for the case when state dimensionality is three [Arasaratnam & Haykin 2011]. However, the CKF uses a more completely determined set of sigma points with less parameters, does not require a mean point, and can approximate nonlinear systems of higher order with better stability [Jia *et al.* 2012]. The Cubature Kalman Filter is also well suited to continuous-discrete state space models, and square root propagation can be used to limit the numerical range required for finite word length machines [Arasaratnam *et al.* 2010] [Sarkka & Solin 2012]. For these reasons, the CKF has attracted great interest for nonlinear filtering and has become a major focus in recent research [Li *et al.* 2009] [Fernandez-Prades & Vila-Valls 2010] [Pesonen & Piche 2010] [Mu & Cai 2011] [Pakki *et al.* 2011]. We will focus on the adaptive and reduced sigma set UKF and CKF filters in this work.

1.6.5 Vision and Navigation

For tracking the motion of a vehicle in an unknown, distant environment, inertial odometry is generally quite inaccurate over long distances and global references such as GPS will not typically be available, so external reference points must be found and used. These reference points also must be identified and tracked while the vehicle moves from location to location. This creates a chicken-and-egg problem, as any inaccuracy from the reference points will propagate to the vehicle localization, which then will propagate back to identification of the reference points [Durrant-Whyte & Bailey 2006]. The solution to this problem comes from first recognizing that the localization of the vehicle and mapping of landmarks must be carried out simultaneously, and second that when this approach is formulated as a single estimation problem, it is convergent [Durrant-Whyte *et al.* 1996]. In the last two decades, research on what has been termed SLAM (Simultaneous Localization and Mapping) has dominated mobile robot research. Much of the groundbreaking work on probabilistic and Kalman-filter based SLAM was done by Durrant-Whyte and others in the late 1990s [Bailey & Durrant-Whyte 2006].

While there are a wide variety of sensory navigation and mapping methods for autonomous vehicles, visual perception has become by far the most common way for a vehicle to obtain information about its environment. This has as much to do with vision being the primary sense of the humans who *design* the robots as it does with the reason why humans rely on vision in the first place: Visual (the sensing of reflected electromagnetic waves with a lens and planar array) sensing is the most efficient method that we have evolved of obtaining a large amount of environmental information in a short time. A wide variety of related non-tactile sensing methods (most also using reflected electromagnetic wave sensing) have been developed such as LIDAR, RADAR, laser and infrared distance measurement, ultrasonics (another form of wave sensing), and more exotic "visual" methods. However, the practicality and intuitiveness of simply using cameras based on the capabilities of the human eye, as well as the recent availability of high-speed image processors and intelligent algorithms, have made machine vision the most popular method by far for wide-range environmental sensing.

Traditional approaches to machine vision have frequently centred on the detection of features.

This includes lines (e.g. Canny edge detection), corners (e.g. Harris corner detection), and other types of features. Structure from Motion (SfM) is a method for obtaining 3-D structures using only feature matches between multiple images. Using either one or two cameras and multiple angles, a 3-D point cloud can be produced, from which structure can be inferred [Shil 2012b]. The technique of bundle adjustment is often used to refine the accuracy of complete point clouds using least-squares estimates [Triggs *et al.* 2000]. Although bundle adjustment is a useful and flexible technique for improving observational estimates, it is not strictly necessary for point cloud reconstruction.

While point-cloud techniques usually involve the matching of every image taken to every other image, the processing requirements can be simplified by making assumptions about which images specifically contain point correspondences. In particular, we can assume that images taken during navigation will have point correspondences with only the most recent set of images, and inertial measurements can help to identify which most recent images will correspond. The mapping methodology follows the probabilistic model that has been used thus far, but implements statistical measures based on visual feature point clouds and applies them to an occupancy map. Stochastic occupancy or "voxel" maps are commonly used for representation of multivalued spatial data.

For implementation on the micro-rover, a very limited platform, it is necessary to use a smaller, more efficient set of methods for SfM. The methods of Hartley & Zisserman, who wrote one of the most influential works on the subject of point cloud triangulation, have been adopted for this [Hartley & Zisserman 2004]. Due to the convenience of OpenCV methods, the need for double-precision arithmetic for accuracy, and the use of a dedicated DSP-based platform for vision, this implementation was done using C++ with OpenCV routines and abstractions.

1.7 Probabilistic Autonomy

Intelligent autonomous operation is easily the most difficult problem in mobile robotics. While known and finite sets of conditions can be planned for and responses pre-programmed using a variety of methods, giving a robot the ability to appropriately handle unexpected and uncertain circumstances remains an open and very challenging problem. Planetary rovers would easily benefit the most from full autonomy, given that they must operate in uncertain conditions while isolated from any direct human assistance. Certainly real-time control of a rover on another planet is infeasible, as the delay in communicating a signal to Mars at the speed of light ranges from 3 to 21 minutes, not even considering temporary communications blackouts and time for retransmitting due to packet errors. However, due to the complexities involved, autonomy on space hardware has been very slow in adoption because of the inherent risks of autonomy failures with the extremely high costs of putting space hardware on other planets. The Mars Exploration Rovers Spirit and Opportunity include facilities for visual odometry (VisOdom) and obstacle avoidance (AutoNav), but due to the limited processing capacity on board (a $20MHz$ RAD6000 CPU), analysis can take 3 minutes for only $50cm$ of movement in both cases, and consequently careful and time-consuming manual supervision of each stage of the rover's movement is performed in advance with fail-safe navigational alarms operating on the rover to halt movement if a planned trajectory is determined to be hazardous [Leger *et al.* 2005]. This both slows down the mission significantly and requires a large ground crew to be constantly available for planning and problem response, though it allows more adaptive and comprehensive mission operations than relying on validated software that may have to be adjusted during the mission [Biesiadecki *et al.* 2007]. The Mars Science Laboratory Curiosity has a more advanced system making use of stereo images to identify obstacles and calculate an optimal path that has been recently tested on Mars [Ackerman 2013], but the reliance on ground crews and manual decision-making continues due to the risks and costs involved. To justify the use of autonomy on other planets, extensive development and testing of robust methods is required, and the cost savings in required personnel for operation will have to become significant with respect to the mission cost. As technology

develops and more missions are fielded, this aspect will become more important with time. Also with the development of more capable and robust autonomy methods, it is also possible to realize significant decreases in mission time required for operations and a related increase in number of mission goals that can be achieved, which could also play a role in decreasing overall cost and actually increasing the likelihood of mission success. Development of intelligent and robust autonomous systems should remain a high priority for future missions.

Many machine intelligence techniques relating to autonomous robotics have been developed. Expert systems for planning and control are commonly designed using Fuzzy Logic (FL), which allows the use of imprecise set memberships and qualitative decisions for programming. Fuzzy Inference Systems (FIS) determine actions based on memberships in sets and operators between these sets, and fuzzy controllers use linguistic fuzzy rules instead of systems of differential equations to describe feedback control [Engelbrecht 2002]. This allows machines to "understand" the world in terms of human qualitative judgements, as humans do. However, the caveat is that such expert systems rely on humans to design the linguistic variables and fuzzy rules, which may be very limited in range of decisions that can be made. In effect, they are really "only as smart as the person who programmed them". For a machine to have the ability to understand and learn, it must be able to properly represent external stimuli. Neural Network (NN) systems, modelled on the basic principles of the human brain structure, attempt to replicate true autonomous learning processes using discrete machine logic. A set of inputs is mapped to a set of hidden neurons by means of activation functions (input mapping) and then weighted sums or products. These then produce a set of outputs by means of more weighted sums or products and can feed the outputs back to the neuron inputs to make the network time-sensitive [Engelbrecht 2002]. Fuzzy logic can be combined with such a five-layer network to build an Adaptive Neuro-Fuzzy Inference System (such as ANFIS or CANFIS) [Jang 1997]. The main problem with neural networks is that of training. The optimization algorithm required to train the system by synthesizing appropriate link weights must be designed appropriately for the problem (as a consequence of the No-Free-Lunch (NFL) theorem) and occupies most of the time and complexity involved in the system's operation. Once trained, neural networks require further training to expand their abilities, which

may degrade the performance of the original system. Another difficulty is that a neuron structure is fundamentally different from a modern computer, meaning that neural networks are typically very resource-intensive to implement.

Another way for machines to make decisions based on intelligence is to use Bayesian logic. Autonomous ground vehicles with a long mission lifetime will need more complex planning systems so that they can adapt to unknowns like mechanical wear, system failures, and changes in environmental conditions. It is desirable for the rover itself to be able to deal with uncertainties probabilistically. Bayesian Networks (BN) are well-suited for handling uncertainty in cause-effect relations, and handle dependence/independence relationships well provided that the network is constructed using valid relational assumptions. Some drawbacks of this method are that the variables, events, and values available must be well-defined from the beginning, and the causal relationships and conditional probabilities must be available initially [Kjærulff 2008]. This makes construction of the Bayesian network a considerable challenge. Machine learning of unknown variables is possible by using Bayesian Field Theory, which provides a method for empirical learning (finding general laws from observations) by combining the probabilistic model for training data with the probabilistic model representing additional a priori information [Lemm 2003]. Additionally, Bayesian networks can be combined with fuzzy logic in what is known as a Fuzzy Bayesian Network (FBN). This combines the capability to use both probabilities and vague quantities in concert. Bayesian networks have been used extensively for image recognition, diagnostic systems, and machine behaviours. However, the potential of these concepts for distributed machine learning and problem-solving is considerable, and warrants further real-world research, so Dynamic Bayesian Networks will form the basis for this research. In the reference [Kozma *et al.* 2008], a self-organized control method for a planetary rover has been studied, but only extends to the navigation problem.

1.7.1 What is a Probability?

The generally-accepted concept of a probability is used frequently in predicting future conditions or events. Examples include “there is a 30% chance of rain”, “the mean time between failures

is 1000 hours", and "only one door out of these three has a prize behind it". All of these examples involve estimation of the presence of some future quantity, and usually are based on either stochastic experience or an estimation using expert knowledge of how a system works. However, the interpretation of what a probability actually *is* or what it represents in real, physical terms has been at the centre of many mathematical and philosophical debates. While a probability p can be seen as simply a "likelihood" of a given event occurring, or alternately a "degree of confidence" that it will occur, there is no existential quantity to give p meaning *right now*. Quite the opposite - p only has meaning when predicting quantities with *uncertainty*. This concept of uncertainty is central to both the definition of probabilistic systems, and their value as a mathematical tool.

Theoretical systems created using the scientific method are by virtue of their repeatability supposed to be clearly-defined and certain, and the science of probability is seemingly a clever paradox - it provides a clear and certain way to model phenomena that are themselves inherently *not* clear and certain. The prediction of the weather, the estimation of a failure in a complex system, and guessing of the location of a hidden prize without having some knowledge of how to reliably predict it are all examples of systems that are, in principle, real and knowable. However, the systems themselves are complex and carry with them additional variable factors that affect the outcome, many of which are either not directly observable or are themselves uncertain from a physical standpoint such as the movements of all the air molecules in the atmosphere, the state of all the parts in a machine, and the thoughts in a game show host's head. Most people know this as the "Butterfly Effect" from chaos theory, a science that grew from tiny anomalies in weather prediction, in which a relatively small event can have far-reaching consequences due to the overlapping of these many additional variables [Gleick 1987].

So how can we deal with all these extra variables? The famous mathematician Pierre-Simon Laplace is recorded to have said in his time that if we knew the behaviours of all the particles in the universe at a given time, then we could calculate their behaviour at any other time, past or future [Hawking 1999]. Aside from the obvious intractability of this solution, does it even seem true? The Heisenberg uncertainty principle states that due to the quantum of energy required, one could not measure the position and momentum of a particle simultaneously without one of them

changing [Heisenberg 1927]. The consequence of this is you can only measure, at best, half of the total information of a given particle at a given time, and if only half the information needed to make an absolutely certain prediction is available, overcoming the "Butterfly Effect" and creating a prediction - any prediction - with absolute certainty suddenly seems to be difficult indeed.

So what is the reason for trying to find certainty in processes that scientifically are supposed to not be certain at all? Albert Einstein is famous for having stated that "God does not play dice with the universe" when discussing with Max Born the then-new science of quantum mechanics, which in itself is the most famous argument against the existence of certainty in predicting the future. This statement carries with it a high risk of misinterpretation, however. Einstein was not questioning the validity of quantum physics itself, but specifically the so-called orthodox Copenhagen interpretation [Clark 1973], for which Erwin Schroedinger's famous Cat-in-a-Box thought experiment was intended as a rebuttal. The Copenhagen interpretation, derived directly from the work of Niels Bohr, Werner Heisenberg, and others deals with the measurement and determination processes of energy quanta, and holds that the process of measurement immediately causes the collapse to a specific value of the probabilistic wave function determining the actual state of a quantum, but the original intent has again been frequently disputed [Faye 2008]. Given the nihilistic implications of such an interpretation, is not surprising that Einstein and others sought "hidden variables" behind the uncertain measurement processes that could maintain the deterministic nature of the universe at the quantum scale, but the existence of such convenient entities has not yet been proven.

Yet there are alternatives to being able to directly predict the future. If precise "strong" measurements destroy state information at the quantum level, what about imprecise measurements, in the grey area between a strong measurement and no measurement at all? A concept was developed in 1988 for using so-called "weak measurements" to retain some information about the prior state of a quantum, and then obtaining many such measurements in succession and averaging so as to obtain an expected value, which unlike a standard expectation value can be a complex number [Aharonov *et al.* 1988]. Recently, groundbreaking work by Lundeeni et al has made use of statistical averaging of weak measurements of position and their complex value in conjunction

with a following strong measurement in an attempt to defeat the Heisenberg uncertainty principle [Lundeen *et al.* 2011]. From this and other similar work, the postulate can be inferred that is not always necessary to know the underlying values of a process if you can predict how it should behave by using indirect methods, specifically statistical and probabilistic methods.

In summary, there is no way to obtain perfect measurements without noise and uncertainty to estimate future quantities in the real world, but what if you don't *have* to know the "real" hidden values behind an uncertain process? Accurate statistical characterization and probabilistic estimation is in many cases a suitable substitute. This is, in fact, the basic principle behind the sigma-point Kalman filter and the Bayesian network. Here, absolute knowledge of the underlying state of a system is not the goal, but rather making a "best estimate" so that appropriate decisions can be made regarding a course of action, is based on the most statistically-important factors while attempting to overcome statistically characterized noise and measurement uncertainty that result from the additional unknowns in the system.

1.7.2 Basic Concepts of Probabilistic Systems

A wide variety of approaches to dealing with probability exist. In general, the concept of uncertainty is encapsulated in a "random variable", which is defined as having a value that is to some degree determined by randomness, in contrast to "definite" variables that have a certain known or unknown value. In a stochastic system, states are determined by the probability distributions in random variables, where the values that a random variable takes on have precise probabilities associated with them, and the sum (or integral) of all probabilities in a random variable is defined to be 1 to reflect that there must be some value associated with the random variable that is present. Probability theory provides a framework for logically handling random variables and the relationships between them so that useful results can be determined. Logically connecting many random variables together based on probabilistic dependencies requires the concept of "evidence", on which a change in a given set of probabilities can be based. The interpretation of Bayesian probability makes use of propositional logic to enable "hypotheses" to be tested and updated based on probabilistic data. This process of "Bayesian inference" is central to our treatment

of probabilistic systems. The term "Bayesian" refers to the 18th century statistical theologian and Presbyterian minister Thomas Bayes, who formulated the basic theorem of statistical inference based on conditional probability.

A vast body of knowledge has been built up around Bayesian theory and methodology. Cox's theorem of probability provides a justification and formalization for inferring the plausibility of postulates [Cox 1961], which has been used to justify and support the use of Bayesian theory for probabilistic inference [Jaynes 2003]. The area of Dempster-Shafer evidence theory, where a "degree of belief" about propositions is represented by numerical intervals between "belief" and "plausibility", is considered to be a generalization of Bayesian subjective probability. Robinson also provided a systematic set of definitions of conditional properties and their consequences [Robinson 1979]. One of the most significant uses of Bayesian theory is the application of Dynamic Bayesian Networks (DBNs) to determine relationships between biological processes purely from statistical data [Vinh *et al.* 2011]. An alternate method for learning a system's relationships based on data is the Grainger causality test, which uses statistical hypothesis testing to determine whether a given data time series is useful in the prediction of another time series. It has been stated that dynamic Bayesian networks perform better for short data sizes, while the Grainger causality approach performs better for larger datasets [Zou & Feng 2009]. Machine learning and making decisions based on series of data is frequently done using Markov decision processes, based on probabilistic transitions between states of a system. Hidden Markov Models (HMMs) function as a simple Bayesian network in terms of time, where only the system states immediately before and after the current time step are considered to determine the current system state, and have found popular application in areas such as bioinformatics and speech recognition. However, since there is generally no consideration for causal knowledge between state variables and the sequence of states is flat, HMMs are very limited compared to general Bayesian networks [Infantes *et al.* 2011]. More recent probabilistic methods use Partially Observable Markov Decision Processes (POMDPs), where a partially-observable Markov model is again used and a probability distribution over all possible states is maintained based on observations and observation probabilities [Koenig *et al.* 1996]. Particle filters are also often used as a method of robust

estimation by directly implementing Bayesian recursion with a grid of state "particles" that represent the posterior density when passed through a system. Though Monte-Carlo methods such as grid filters are inherently more "random" than deterministic methods and are inefficient for dealing with high-dimensional systems, they provide the optimal solution for a discrete state space with countable states and are relatively reliable [Liu 2001].

Probabilistic Bayesian systems are closely related to, and are frequently compared with, fuzzy systems. Fuzzy sets are effectively "classes" of objects with varying degrees of membership, such that a fuzzy set can take on several values but to different degrees [Zadeh 1965]. A fuzzy set "room temperature" could take on the values "hot" and "cold" but to different degrees such that a high room temperature has a high degree of membership to "hot" and a low degree of membership to "cold", but anything we would consider to be moderate has some membership of each. This has obvious parallels with a random variable, which can also take on many values, not in terms of "membership" but in terms of "probability", or to what degree a value *might* be present given its statistical characterization, and only one value is assumed to be an outcome. A random variable for "room temperature" might give a probability of 0.6 for "hot" and 0.4 for "cold" but ultimately could only be either "hot" or "cold". Put another way, a random variable is expected to have a single outcome while fuzzy sets represent multiple outcomes, or alternately imprecise degrees of belief. This imprecision is reflected in the extension to fuzzy sets of possibility theory, in which the "possibility" of an event describes a measure of likelihood and its complement is the "necessity" of an event [Zadeh 1999]. Possibility theory can be seen as a kind of upper probability theory, where the operation of addition corresponds to a maximum, or as a plausibility measure in Dempster-Shafer evidence theory. Several other formalisms for reconciling probability and fuzzy logic have been built upon these concepts, generally focused on the creation of a "fuzzy random variable" that can take on fuzzy values with varying probabilities [Hajek *et al.* 1995] [Liu & Liu 2003] and often can be related to Dempster-Shafer theory. There is also some interest in using fuzzy theory as an approximation to probability theory when priors are not known [Dubois & Prade 2001], and type-2 fuzzy systems also could be seen as adding uncertainty to a fuzzy variable due to the extra rules involved [Lin *et al.* 2010].

With so many different methodologies in use, it is no surprise that it is sometimes hard to clearly interpret them in realistic terms. The literature on probabilistic systems and intelligence contains many (often heated) differences of opinion regarding the validity and interpretation of these systems. It is the author's observation that these are most often caused by differences in interpretation such as "probability refers to my system now" versus "probability is only a future estimate", "probability describes real quantities" versus "probability is purely an abstraction", and "probabilities are precise in their formulation" versus "probabilities are by nature an approximation". This is closely related to the interpretation of a system as being either composed of deterministic physical principles or uncertain stochastic processes, with the only difference often being the degree of prior knowledge about how the system works, and it has been noted that practically-minded roboticists and theoreticians of reasoning under uncertainty tend to have different views on what a probability actually is [Bacchus *et al.* 1998]. It would perhaps be most appropriate to say that probability provides a form of meta-information about the behaviour of a system when accurate statistics regarding a given situation are available, independent of the interpretation.

Each method that has been developed for complex machine reasoning has certain specific and established advantages, but little emphasis has been placed on the advantages that a more general interpretation of such methods might have. It has been observed that purely probabilistic systems bear many mathematical similarities to fuzzy systems and state machines, and consequently that the vast majority of real implementations use the same set of simple mathematical constructs for numerical storage and analysis regardless of the interpretation of how these constructs actually reflect reality. Fundamentally, there are only a limited number of simple numerical representations for the abstraction of real quantities and qualities that are possible within commonly-used mathematical systems, and it is understandable that the methods that operate best on these systems tend to resemble each other. There are several practical situations where implementing either fuzzy or probabilistic models would provide similar results due to these mathematical similarities. It is therefore appropriate to expect some manner of convergence between the many different methodologies, such as the generalization of a variable to a construct with both fuzzy

and random properties that may or may not be significant in a given situation. This is analogous to how matter is modelled as having both particle and wave properties at the subatomic level but exhibits mainly particle properties at the macroscopic level. One form of probabilistic convergence is the generalization of Bayesian systems such as HMMs and Kalman filters to Bayesian networks. These kinds of similarities between systems make them both easier to work with and more widely applicable, and suggest that there are significant advantages in the use of a common framework for probabilistic machine intelligence.

1.7.3 Building Bayesian Systems

There are many existing software packages for building and using Bayesian networks, but to retain the flexibility and freedom of non-proprietary, open-source software that the μrover has been built on, we will restrict the selection of available software to currently-maintained community FOSS packages. For pure research purposes, the Octave and MATLAB environments are frequently used, and have numerous toolkits and user-distributed packages for Bayesian calculation and Bayesian networks. However, these are very high-level languages, and require computation, memory, and user interface resources that are generally not present on embedded systems, as well as being wasteful from a functional standpoint. Java also has several frameworks available, but as stated already, the Java language offers little for embedded systems that is not already provided by C++ and in the interest of minimizing system resources, the μrover does not run a Java VM. The Python language is, in some cases, suitable for embedded use and provides a very comprehensive set of libraries and features that are comparable to higher-level languages like Octave, such as those in the Numpy and SciPy packages. Bayesian network packages based on Numpy include the original Mocapy for dynamic Bayesian network construction, PEBL (Python Environment for Bayesian Learning), and the OpenBayes framework based on the MATLAB Bayes Net Toolbox (BNT). These provide a higher level of abstraction and easier programming than low-level code.

While the use of Python is under consideration for embedded robotics due to the ease of programming and, as stated earlier, may form the basis for future space systems, the dependence

on a UNIX environment, resources required for the VM, and extra complexity in implementation still make native C and C++ code more attractive for space embedded applications without a user interface. Several currently-maintained and FOSS packages were considered for use as the Bayesian system on the μrover. The original framework for Bayesian Robot Programming, known as OPL for Open Probabilistic Language, was originally made available freely [Lebeltel *et al.* 2000] but was soon afterwards unfortunately removed from availability, renamed and redistributed as a proprietary package called ProBT [Lebeltel *et al.* 2004], eliminating it as a candidate for future community development. The Bayes++ library of C++ classes uses an existing Bayesian network to implement discrete filters such as Kalman filters, particle filters, and other probabilistic constructs, and includes several different abstractions of Bayesian systems. However, it focuses on filtering processes and does not abstract a Bayesian inference system. A toolkit specifically for dynamic Bayesian networks is Mocapy++, which is a C++ implementation of the Mocapy DBN inference toolkit that is designed for Bayesian inference and structure learning, primarily for medical use. Mocapy is designed for constructing dynamic Bayesian networks with maximum likelihood parameter learning using Markov Chain Monte Carlo methods, and was originally written for bioinformatics use in molecular biology. However, it is very complex, supporting several different types of nodes, and was determined to be overbuilt for embedded use. Both APIs are now written in C++ and complement each other to some degree, as Mocapy constructs and traverses networks while Bayes++ directly implements them. Both are also dependent on floating-point mathematics and not ideal for embedded use, and of course, they do not communicate. The most promising candidate for embedded Bayesian inference is LibDAI, a FOSS C++ inference engine that includes several inference methods and parameter learning of conditional probability tables [Mooij 2010]. While a robotic inference system could be built on top of LibDAI, it was still under development at the time of this research and was determined to be not ideal for use in BRP methods, requiring significant amounts of extra programming. There is no known general framework that is implemented in bare C for efficiency and portability as is preferred for μrover software, and is focused specifically on robotic use in embedded systems, particularly with consideration to fixed-point math. It was therefore determined that the Bayesian

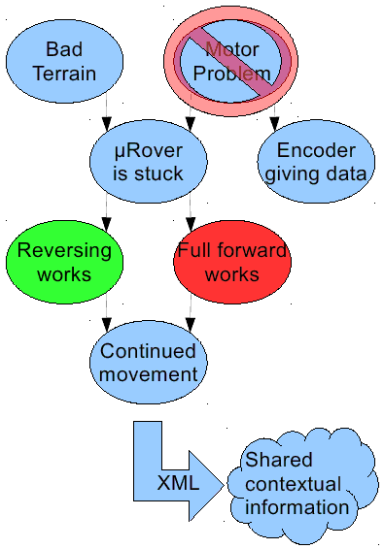


Figure 1.16: Example use of a Bayesian network for decision-making

inference engine for the μrover had to be written from scratch despite the extra time and effort required.

1.7.4 Bayesian Network Robot Programming

A probabilistic mission execution system has been designed for control of the μrovers. The basis of the machine learning and decision architecture for the μrover is the use of Bayesian Networks. Bayesian networks are similar in concept to fuzzy systems, but have one major theoretical difference: *Fuzzy memberships are constant and imprecise, whereas probabilities refer to precise quantities but may change when an event occurs.* Probability in this case implies directed causality $a \rightarrow b \rightarrow c$, and Bayesian networks allow associations to be defined to determine the likelihood of both a and b if c is known. Using associated probabilities allows a machine learning system to calculate not only what specific actions should have a certain effect, but what actions are unlikely to have that effect, and also what actions may have already had that effect or not. This ability to characterize multiple causes and "explain away" unlikely causes makes a Bayesian network a very powerful predictive tool. Figure 1.16 shows an example of how this causal reasoning process may work for the likely situation of a stuck rover.

A Bayesian Network (BN) can be visualized as a directed acyclic graph (DAG) which defines a factorization (links) of a joint probability distribution over several variables (nodes). The probability distribution over discrete variables is the product of conditional probabilities (”rules”). Chain Graphs generalize DAGs, but require more semantic complexity and are less intuitive. For a causal statement $X \rightarrow Y$ often it is needed to find $\mathrm{P}(X|Y = y)$ using the distribution $\mathrm{P}(X)$. Rev. Thomas Bayes' rule states that

$$\mathrm{P}(X|Y = y) = \frac{\mathrm{P}(Y = y|X)\mathrm{P}(X)}{\mathrm{P}(Y = y)}. \tag{1.1}$$

A Bayesian network, when viewed from the point of view of a single joint probability distribution, can be represented by:

$$\mathrm{P}(X_1 \cdots X_n) = \prod_{i=1}^{n} \mathrm{P}(X_i|\mathrm{Pa}(X_i)) \tag{1.2}$$

where the parents of each i'th node X_i are denoted by $\mathrm{Pa}(X_i)$. To construct a BN, you identify the variables and their causal relationships and construct a DAG that specifies the dependence and independence assumptions about the joint probability distributions. Bayesian networks can be constructed using semi-automated means if there is information regarding the probability and independence of network variables. It is proposed that a Bayesian network can be constructed using information about the intrinsic structure of the rover itself and the mission plan, where causality is implicit as the structure is broken down into components and sub-components.

1.7.5 Knowledge-Based Autonomy

The μrovers use probabilistic methods for mission planning, operations, and problem-solving, where each state variable in the system is considered to be a random variable. A Bayesian Network is used to represent conditional relationships between the random variables, which may change as actions are performed and information on success or failure is gathered. A variable with two states such as the sensor model discussed above follows a Bernoulli model. Bayesian networks generalize these concepts to a tree structure, represented by a directed acyclic graph

(DAG) which defines a factorization (links) of a joint probability distribution over several variables (nodes).

Rovers work in a partially unknown environment, with narrow energy/time/movement constraints and, typically, limited computational resources that limit the complexity of on-line planning and scheduling. This is particularly true of micro-rovers and other small robots using low-power embedded systems for control. These constraints represent a significant challenge in the design and programming of rovers, and considerable research work has been done on efficient planning algorithms [Penna *et al.* 2010] and algorithms to autonomously modify planning to handle unexpected problems or opportunities [Estlin *et al.* 2011]. Adaptive learning and statistical control is already available for planetary rover prototypes, and can be significantly improved to decrease the amount of planning needed from humans [Huntsberger 2009], but is still often underutilized. Gallant et al. [Gallant *et al.* 2011] show how a simple Bayesian network can operate to determine the most likely type of rock being sensed given a basic set of sensor data and some probabilistic knowledge of geology. To make it possible for the small, efficient planetary rovers of the future to be decisionally self-sufficient, focus is needed on the design and implementation of efficient but robust embedded decision-making systems.

1.8 Environmental Tests

To verify the operation of the micro-rover hardware and actuators under space conditions, thermal vacuum (T-VAC) testing was conducted for the complete micro-rover electronics stack and one motor in September 2012 at York University. This testing was done simultaneously with testing of the existing electronics stack for the York QB50 nanosatellite. Two tests were run on the μrover hardware: an OBC stress test to estimate heating of the OBC in high-processing periods, and a motor drive test to verify the operation of the brushless motors in vacuum.

Chapter 2

Design

There are a wide variety of different ways to construct a roving ground vehicle, depending on the nature and environment it is intended to operate in. The design and construction of the μrover is detailed in this chapter, including mechanical, electronic, and software components. Key innovations include the passively-actuated swing-arm suspension system with parametric control, the hybrid and flexible motor control architecture, efficient multiple-voltage power distribution bus, embedded sun sensing methods, DSP vision board, robust and flexible communications system, and adaptive sigma-point Kalman filtering implementations with fixed-point arithmetic.

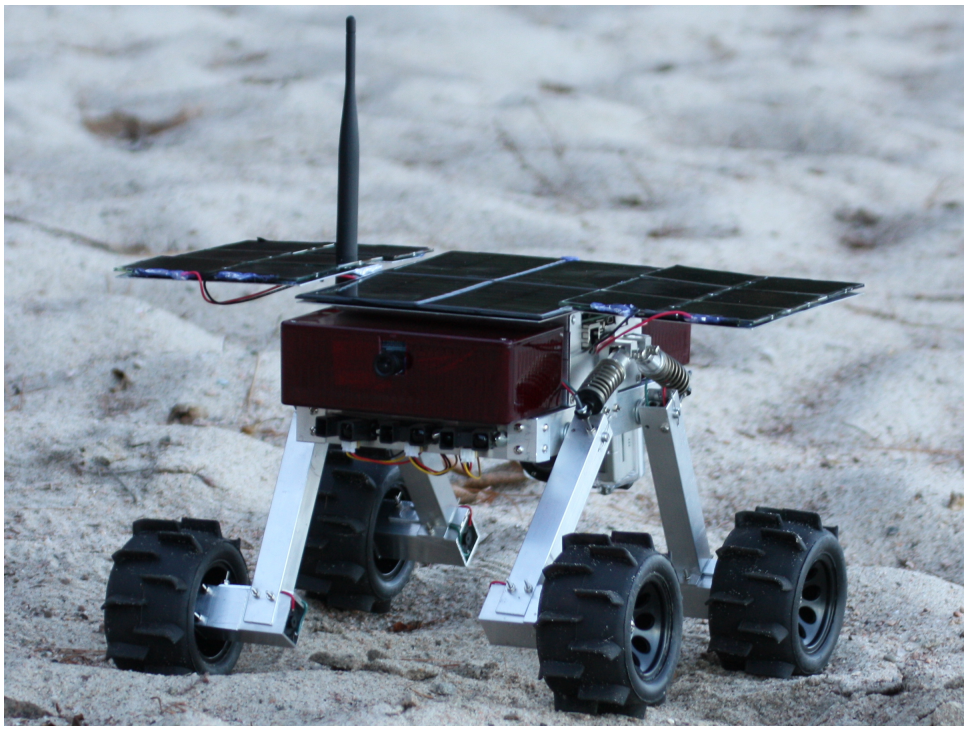


Figure 2.1: μrover Operating in Sand at the ARO

2.1 Mechanics

As the μrover was designed for research purposes and simple missions, with the criteria of simplicity, reliability, and cost-effectiveness, the mechanical designs are minimalist in philosophy and flexible. Only 14 moving parts are used, with the extending suspension arms and rotating solar panel mounts only being used for deployment, and the four motors and four suspension arm joints being used for mobility. Figure 2.1 shows the μrover prototype operating in sand, which due to the wheel slip it allows, provides good steering control for differentially-driven vehicles.

2.1.1 Chassis Design

To simplify construction and maintenance, and minimize the chance of structural failure, the chassis design for the μrover is designed to be very simple and sturdy. Parts needed were standardized as much as possible. $1 inch$ ($25.4 mm$) outer diameter aluminum tube with the minimum

wall thickness available of $1/16inch$ was used for both the frame and suspension arms, and the prototype was machined to sufficient tolerances using only a band saw and drill press. Flat aluminum with $1/8inch$ thickness is used to fabricate the suspension supports and drivetrain mounts. The solar panels are supported by $1/16inch$ 6061-T6 aluminum sheet with a $5mil$ thick Kapton film coating to ensure electrical insulation. Solar panels are sized so that they will not interfere with each other as they deploy. The platform for the μrover components is a square of tubular aluminium of horizontal size $300mmx150mm$, through which the wiring to chassis components is run and sensors are placed.

The electronics are housed in a $100mmx150mmx50mm$ box in the center of the chassis, and payloads are housed in $50mm - 100mmx150mmx50mm$ boxes at the front and back. Solar panels are mounted on the electronics boxes, and raised on cylindrical supports that allow the side solar panels to rotate. A solid model is shown in Figure 2.2. The electronics enclosure and supporting chassis sizes were chosen so that a PC/104 sized electronics stack (a form factor commonly used for CubeSats and other small space hardware) could fit easily inside with clearance for wall thickness, and an additional $100mmx50mmx50mm$ of interior space for a battery stack. The center line of the chassis extends $50mm$ beyond the main electronics box front and back, allowing payloads to be mounted via screws to the platform and the main enclosure, and connect electrically via D-sub connectors. This allows the chassis to be no larger than necessary to support the electronics and payloads, but large enough in section to be very rigid.

The $166mm$ communications antenna that is currently used for $900MHz$ communications with the base station/lander is mounted above the taller of the solar panel supports. Good communications requires the antenna to be as high as possible to achieve line of sight with the base, lander, or nearest relaying entity for as much time as possible. The antenna is sized as a half-wavelength dipole [Johnson & Jasik 1984], but larger antennas for lower or higher frequencies can also be used provided they can fold. The length calculation is

$$\frac{\lambda_r}{2} = \frac{c}{2f_r} = \frac{1}{2}\frac{3\times10^8}{900\times10^6} = 0.166\ (m). \tag{2.1}$$

Numbered screw sizes with only one thread count per diameter were used to minimize the

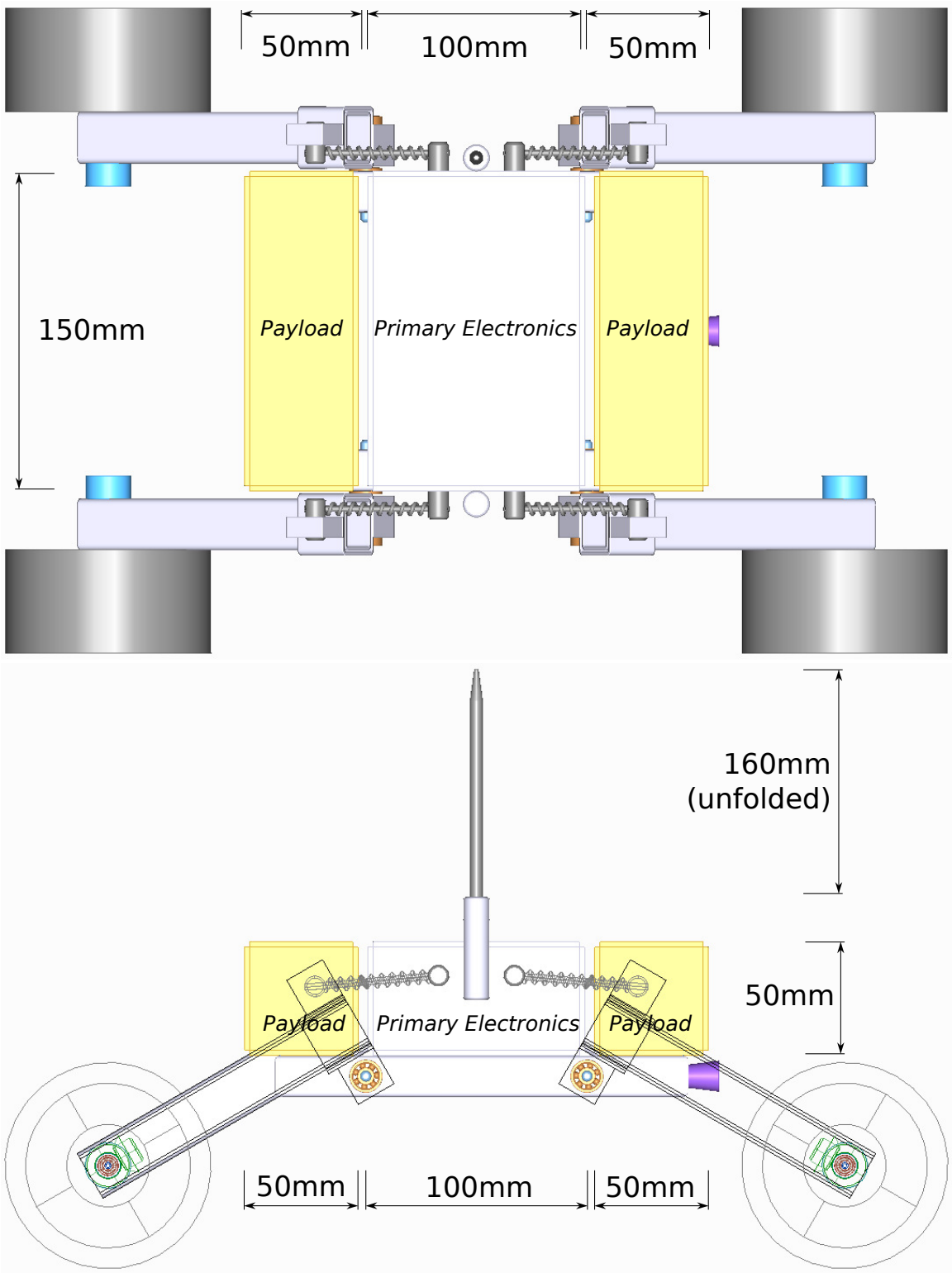


Figure 2.2: Dimensions of Electronics Enclosures for μrover

selection of fasteners needed, and as a halfway point between outdated imperial sizing and metric sizing that is preferred but difficult to obtain easily in North America. The numbered screw sizes used were only #2 – 56, #4 – 40, #6 – 32, #8 – 32, and #10 – 32, which are to some extent similar to metric sizes. However, 2*mm* metric screws were required for mounting the motors. All screws and steel parts are made from austenitic 304 and 316 stainless steel to minimize the magnetic permeability of the μrover so as not to interfere with sensors or become magnetized over time. The motors, held further from the instruments in the electronics enclosures, are the exception as suitable air-core (induction) motors built from nonmagnetic materials were not available, and would also be significantly less powerful.

2.1.2 Suspension Design

Based on the criteria of simplicity, compactness, and reliability, a suspension design based on four independent suspension arms was chosen for the μrover. The wheels are mounted on the end of square aluminium tubes, which can be sealed to protect the drivetrain components. The suspension is sprung through horizontally-mounted spring dampers that are hinged at the ends to allow free rotation as the suspension arms move. The prototype μrover uses adjustable threaded rods to calibrate the suspension, but the calibrated position is fixed in the completed design. The length of the suspension arms must be long enough to house the motors and provide sufficient ground clearance, while not making the rover high enough to tip easily and being able to fold into the chassis for transportation. The chassis with suspension is shown in Figure 2.3.

This suspension design is similar in concept to the Horizontal Volute Spring Suspension (HVSS) used on tanks and other off-road vehicles, which uses pairs of wheels or bogies that are sprung against one another with a freely-moving volute spring used to transfer loading between each wheel [Zaloga 2003]. However, unless used in conjunction with other bogies to prevent body rotation, the spring will not resist body tipping, so the springs in the μrover suspension are fixed to the chassis at the center line by rotating pin joints. The speed of the motors is monitored using rotary Hall sensors, and the angle of the suspension arms can be directly measured by means of potentiometers on the suspension joints that are monitored by ADC channels on the mo-

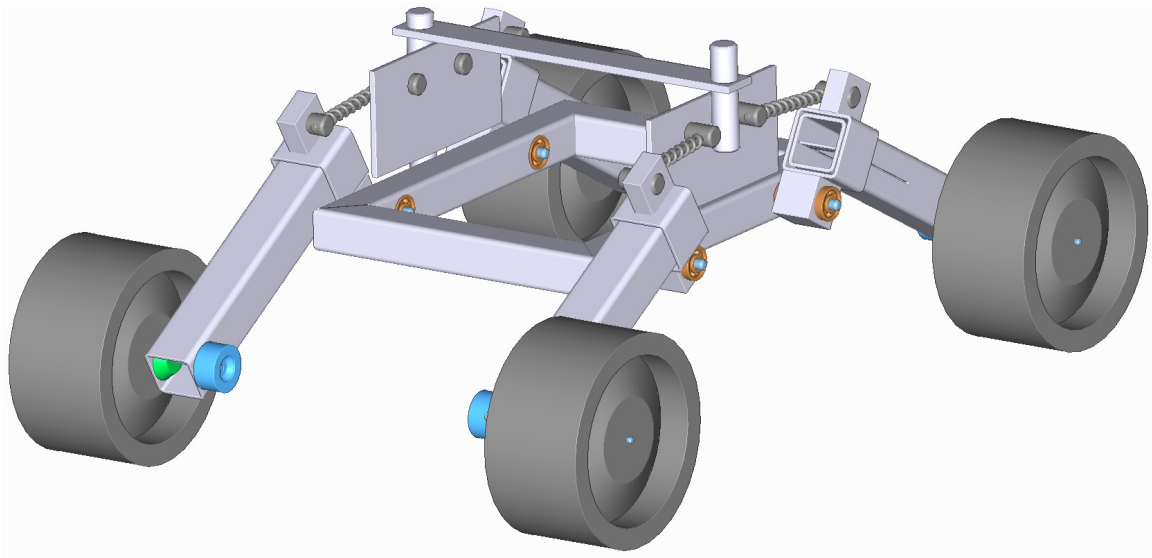


Figure 2.3: μrover Chassis

tor controller. Although the Figures show the tubular aluminum with open ends for illustration, the ends would be securely closed and wires passed into the chassis through sealed grommets in a finished μrover for field use.

2.1.3 Deployment

As the fully-deployed chassis is expected to be too large to fit in the lander module planned for Northern Light, and it must be contained very securely while in transit or for transport for other uses, the μrover must compress and then deploy by itself. It is preferable that the deployment be automatic and spring actuated, rather than adding the complexity of electronic actuation. To accomplish this, solar panels are placed permanently on top of the electronics enclosures and also on two swivelling supports at each side, that rotate to deploy via rotary springs when released from the lander. Thermally resistant and heavily Teflon-lubricated sleeve bearings are used to allow this one-time rotation, motivated by type 302 stainless steel torsion springs within the solar panel mounts. While stowed, the solar panels overlap facing up, so in the event of a deployment failure, some solar power can still be generated, with approximately 50% of the power available with respect to the deployed configuration.

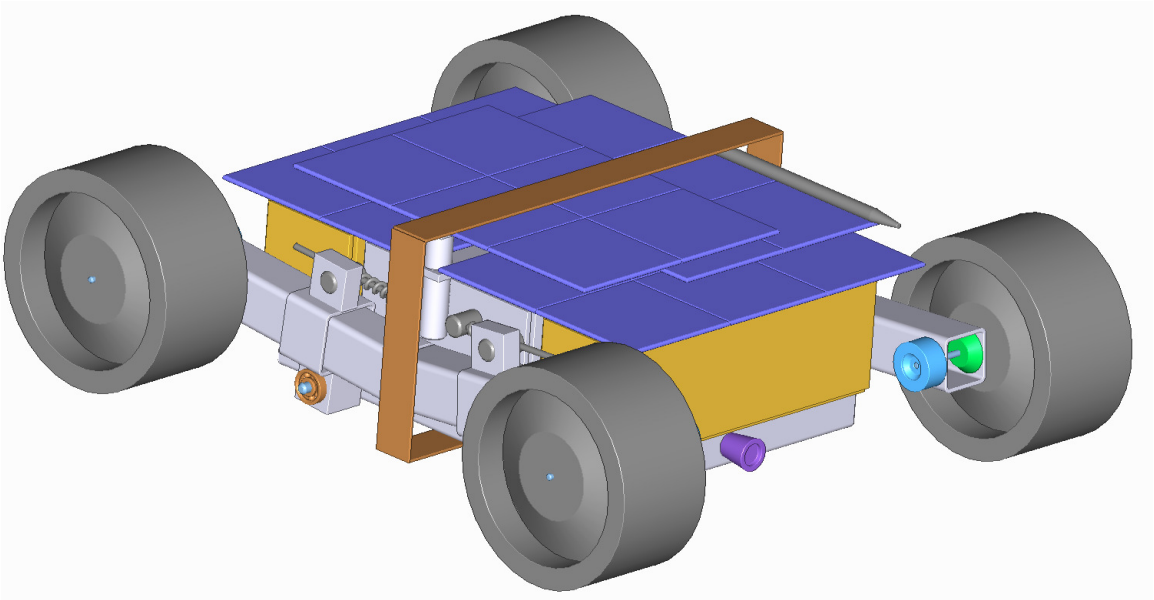


Figure 2.4: μrover in Compact Stowed Configuration

The suspension arms present a more complex problem, as they must only rotate in a single plane for stability, and the motors and gearboxes require the arm for storage. The chassis design overcomes this by housing the suspension arms in a sliding mount adjacent to the chassis connection that is internally loaded with a stainless steel compression spring to extend the suspension arm when released from the lander. The springs are sufficiently strong as to keep the arms extended even in the event of a collision or failed motor, and Teflon dry lubricant is used to ensure that the tubes do not bind. In the event of a failed deployment, the μrover can still move, although with significantly reduced ground clearance, as long as the suspension arms can still rotate downwards by 15° or more. Figure 2.4 shows the stowed configuration, held in place by a single strap around the diameter that is attached to the host module, and outer dimensions are shown in Figure 2.5. For automatic deployment, all that is necessary is to cut the strap, which can be done by a metal blade attached to a heating coil that is heated electronically by the host module, immediately allowing the solar panels to deploy and the suspension arms to extend and rotate.

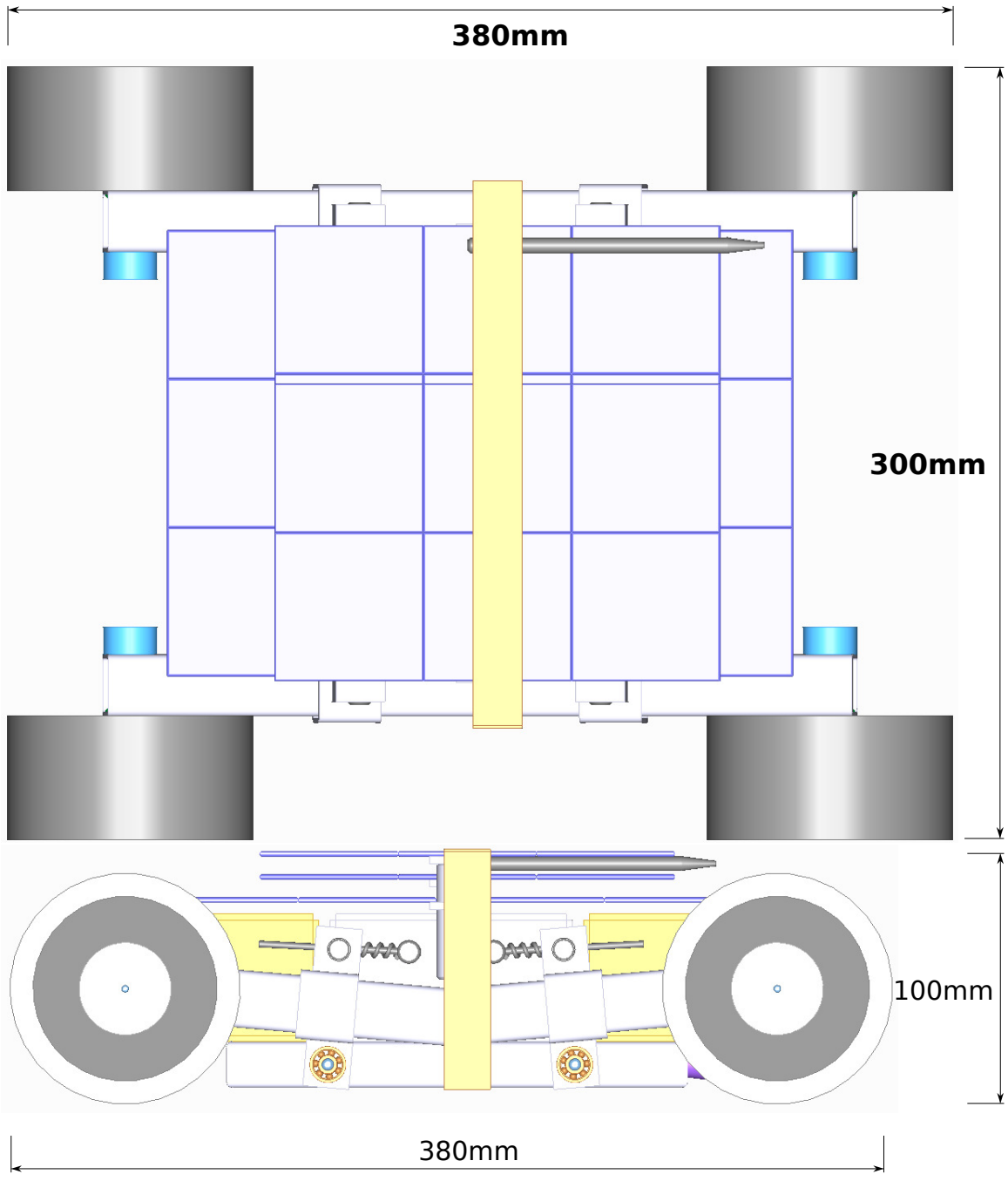


Figure 2.5: Dimensions of μrover When Stowed

2.1.4 Motor and Drivetrain Selection

The drive motors are most critical moving component of a planetary rover, and the selection of an appropriate motor for the μrover involves many considerations. First, there is compactness and efficiency. The motor should fit within the chassis with a diameter $d_m < 2.5cm$ or at least be easy to cover in order to be protected from the environment, and adding extra weight and complexity to accommodate the motor must be avoided. Also, the power draw for four motors must be within the limited power ($10W$) that can be continuously produced by the batteries, and lower power draw allows for longer run times before recharge, assuming the motors, driver, and OBC draw more than the solar panels can provide. Several models of DC drive motor were used for initial testing of the μrovers, and on a larger scale the YURT rovers, and it was determined that high tolerances on both the electronic drive system and the drive gearboxes were essential to avoid failures in rough terrain and adverse conditions. Additionally, the motors used for flight hardware must be autoclavable and tolerant of extreme temperatures for the trip in hard vacuum to Mars, and then operable in an atmospheric pressure of $0.6kPa$ and potentially temperatures of $-55°C$ or lower on the surface of Mars.

The fundamental requirement for the drive motors is that they provide sufficient torque to move the rover on any surface it is expected to have to drive over. High speed is not generally required or even desirable for planetary rovers and autonomous robots in general, which should not move faster than they have to for a mission goal and must not move faster than their navigation and collision-avoidance systems can follow. Using torque as a base requirement, we can estimate the torque τ_m from a motor based on the wheel radius $r_w = 0.1m$ using the basic equation for torque $\tau_m = f_m \times r_w$. The rover's mass m_r will exert a body force $f_g = m_r g$ in the vertical direction. Vertical forces are perpendicular to the applied wheel torque on a perfectly flat surface, with the ground at an angle of $\phi_{GND} = 0°$ with the plane perpendicular to the gravity vector. However, if the rover is on a slope of any kind or if a wheel is climbing a small hill, the wheels will experience a component of the gravity vector in the direction of the applied motor force $f_g \sin(\phi_{GND})$, which the motor must overcome to move up the slope. The weight distribution over the wheels due to the terrain varies considerably, but for motor selection it is sufficient to

just consider the worst case, where a steep slope causes most of the rover's weight to be on two wheels (though in this situation, the rover should stop due to accelerometer feedback well before the tipping point is reached). Distributing the weight of the μrover over two wheels out of four lets us describe the worst-case torque required for each motor as

$$\tau_m = r_w \frac{m_g g}{2} \sin(\phi_{GND}). \tag{2.2}$$

This may at first seem to be a high estimate for required torque, but in outdoor environments terrain features, loose rocks, and high friction in bearings can all increase the torque required for normal operation, and a high estimate is important. The YURT rovers were designed with these metrics, and proved to have just sufficient (but not excessive) torque for off-road operations. To obtain this level of torque, a gearbox is essential, as DC motors by themselves provide torque on the order of millinewton-metres and speed on the order of hundreds of radians per second and a gear ratio of at least 100 is usually necessary for a slow-moving rover. A gearbox has the effect of simultaneously increasing torque depending on the ratio of gear teeth (or more generally, circumference) between two gears, and a set of gears that implements a given ratio is known as a *stage*. Given a gear with n_1 teeth in contact with a gear with n_2 teeth, the speed of n_2 with respect to n_1 will be $\omega_2 = n_1/n_2\omega_1$ and the ratio of torques will be $\tau_2 = n_2/n_1\tau_1$, simply because the rate of circumference travel of the two gears must be the same, but the total *amount* of circumference per revolution is different. It is important to realize that neglecting frictional losses, this means that the total mechanical power in the system stays constant as it should because $\omega_1\tau_1 = n_2/n_1\omega_2 n_1/n_2\tau_2 = \omega_2\tau_2$. Additional considerations for gearing include the maximum rated torque of the gearbox τ_{max}, beyond which damage to the gearbox may result.

The type of gearing used is also important for the efficiency of the motor. Each stage of gears adds frictional resistance to the drivetrain and therefore losses in torque at the output shaft, but the highest ratio possible in a small space is typically needed. Therefore it is important to use gearing stages with the largest possible ratio of sizes between gears to obtain the highest ratio per stage. Gearbox types include common spur gears, planetary gears, and worm gears. Spur gears are most often used because they have high (90%) efficiency, but low gear ratios due to the large gears

required in limited space, meaning that several stages will be needed and efficiency will drop as the product of the individual stage efficiencies. Worm gears offer the highest ratio conversion in a small space, but because of the large surface area of the worm that is used for driving the output gear and the resulting friction, worm gears have the lowest efficiency (70%) and are generally not back-drivable from the wheels (which can cause drivetrain breakage if the wheels are forced or jammed). Planetary gearsets, in which a connected ring of gears transfer motion between a central drive gear and a round outer gear ring, generally provides a good tradeoff of efficiency (80%), compactness, and gear ratios, and are most often used in transmissions and low-loss drives. An additional type of gearbox is called a harmonic drive, which operates in a similar fashion to a planetary gearbox by which a deformable inner gear barely differs in number of teeth from an outer gear ring, and regular deformation of the inner gear ring by a cam from the input shaft causes a very slow progression (one tooth per input revolution, per cam lobe) of the inner gear ring with respect to the outer. Although micro harmonic drives are available, they are extremely expensive due to the precision of their construction and even more so in a form that could be used in space due to the material considerations involved. The best low-cost gearbox for a micro-drive in this case is a planetary gearbox. Given that the greatest reduction possible using spur gears within $2.5cm$ pitch diameter is $8/28 = 1/3.5$, a reduction of over $1/100$ would require four stages, and about $5.5cm$ of linear space, while a planetary gearbox can provide a comparable reduction ratio in three stages and $4cm$ of space, narrowing the efficiency gap as well. Additional benefits include compactness and resiliency, as the gearbox case forms part of the gearing.

Motors are rated with two torque figures. The rated torque τ_{rated} is the safe torque for continuous operation, while the stall (or ultimate) torque τ_{stall} is the torque at which the motor is no longer capable of moving the shaft. The general rule is that τ_{rated} should not be exceeded under normal operations on shallow grades, and τ_{stall} should not be exceeded in the worst-case estimate above. In earth's gravity of $9.81m/s^2$, assuming wheel diameter $0.1m$, mass $6kg$, and a normal grade of $\phi_{GND} = 10^\circ = \pi/18 radians$, and a maximum grade of $\phi_{GND} = 45^\circ = \pi/4 radians$ the motor will need to operate at

$$\tau_{rated} = 0.1\frac{(6)(9.81)}{2}\sin(\frac{\pi}{18}) = 0.511 \tag{2.3}$$

$$\tau_{stall} = 0.1\frac{(6)(9.81)}{2}\sin(\frac{\pi}{4}) = 2.08 \tag{2.4}$$

where τ_{rated} and τ_{stall} are in Nm. The operating environment has a great impact on the requirements for actuator hardware. On Mars, with a gravitational constant of $3.711 m/s^2$, the requirements are $\tau_{rated} = 0.193 Nm$ and $tau_{stall} = 0.787 Nm$, and on the moon with a gravitational constant of $1.622 m/s^2$, the required torques are a minuscule $\tau_{rated} = 0.0845 Nm$ and $\tau_{stall} = 0.344 Nm$. Although this is a simple model and there are other challenges due to planetary environments, it shows that to obtain equivalent operation on Earth, larger motors are needed than the motors for space use.

Only one motor was found that fit the requirements of compactness ($d_m < 2.5 cm$), torque, and minimal power consumption. The BLWRPG093S-12V-3500-R139 from Anaheim Automation is a NEMA-09 form factor BLDC motor custom wound for $12V$ with a Y winding topology, 8 poles, Hall sensors for rotor position placed 120° apart, a rated speed of $366.5 rad/s$, and maximum rated torque of $30 mNm$. The motor is mated to a $139:1$ planetary gearbox, which allows the motor to provide up to $4.17 Nm$ of torque. It can also be assembled to be autoclavable (vacuum rated), though cost increases significantly.

In the complete μrover, the wheels must be driven at right angles to the motor, which is contained within the suspension arm to keep the chassis compact. Mitre gears are used to drive the wheels at right angles to the motors, and to fit the limited space within the $1 inch$ chassis tubes, the largest mitre gears readily available are 32 diametral pitch from Boston Gear, with a hub diameter of $10.4 mm$ and tooth face $3 mm$ in size (cat. no. GSS462Y). Two of these gears are used in each suspension arm, one bored out to $6 mm$ shaft diameter for mounting on the BLDC motor, the other bored to $1/4 inch$ for mounting on a flatted driveshaft for the wheels. The gears are held on with $\#4-40$ set screws in tapped holes. Being constantly in motion, the driveshafts and suspension arm joints are supported by rotary bearings. The bearings used are ABEC-5

flanged double-shielded bearings, with $3/8inch$ outer diameter press-fitted into the aluminium frame, and $1/4inch$ inner diameter to accommodate the drive shafts and suspension arm pivots. Figure 2.6 shows the outside and the inside in wireframe view of one of the suspension arms with deployment spring, brushless motor, mitre gears, and axle.

2.1.5 Physical Conventions

For control purposes, the wheel rotation speeds $\omega_{m1}, \omega_{m2}, \omega_{m3}, \omega_{m4}$ are defined with respect to the motor shafts. Since with no wheel slip a round wheel traverses $2\pi r_w$ of distance for each rotation, the wheel rotation speed ω_m in radians per second is simply related to the speed of forward movement of a wheel as $v = r_w\omega_m$ and the factor of 2π is not needed since radians per second are used. We also define the wheel speed vector $\omega_{\mathbf{m}} = \{\omega_{m1}, \omega_{m2}, \omega_{m3}, \omega_{m4}\}$. Similarly, the torque τ that a motor applies to the shaft of a wheel is related to the horizontal force F_m that the wheel applies to the shaft when in contact with the ground as $F_m = \tau/r_w$. We will not attempt to model the complex phenomenon of wheel slip on uncertain surfaces. Rather, wheel movement will be used as a basic estimate of whether the rover should be moving forwards, turning, or stopped, and use other sensory methods to estimate the quantity of total movement. An illustration of the coordinate systems used on the μrover is shown in Figure 2.7.

Wheels are numbered starting at the front-left of the vehicle, then proceeding to front-right, rear-left, and rear-right. As nearly all rovers are supported with horizontally-opposed pairs of wheels for stability, this numbering system makes it relatively simple to add additional sets of wheels, and makes it convenient . For other wheel configurations such as the SHRIMP rover [Estier *et al.* 2000], the wheels closest to the front are numbered first, and then any wheels behind those that are horizontally opposed are numbered starting at the left side.

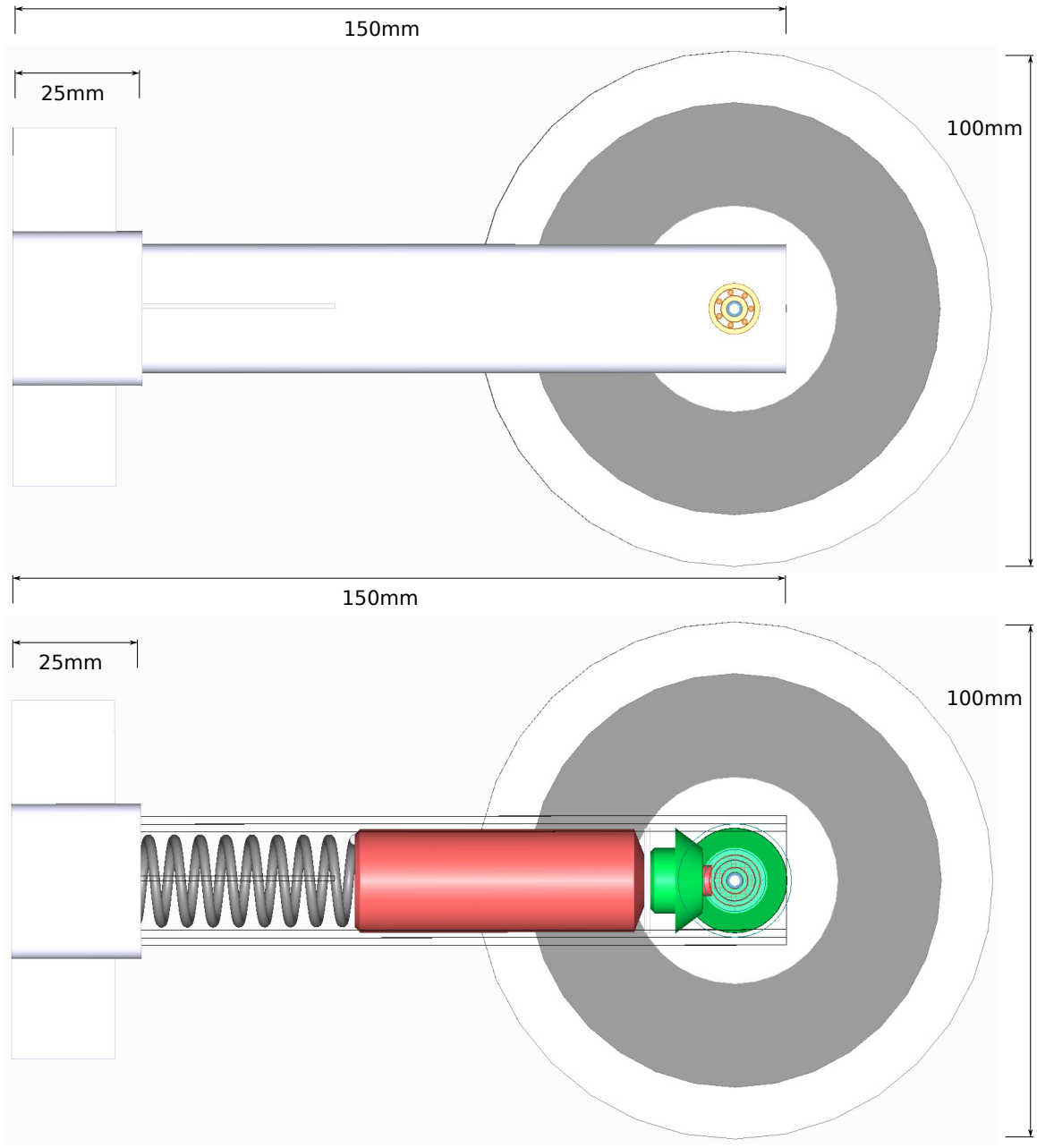


Figure 2.6: Detail of μrover suspension

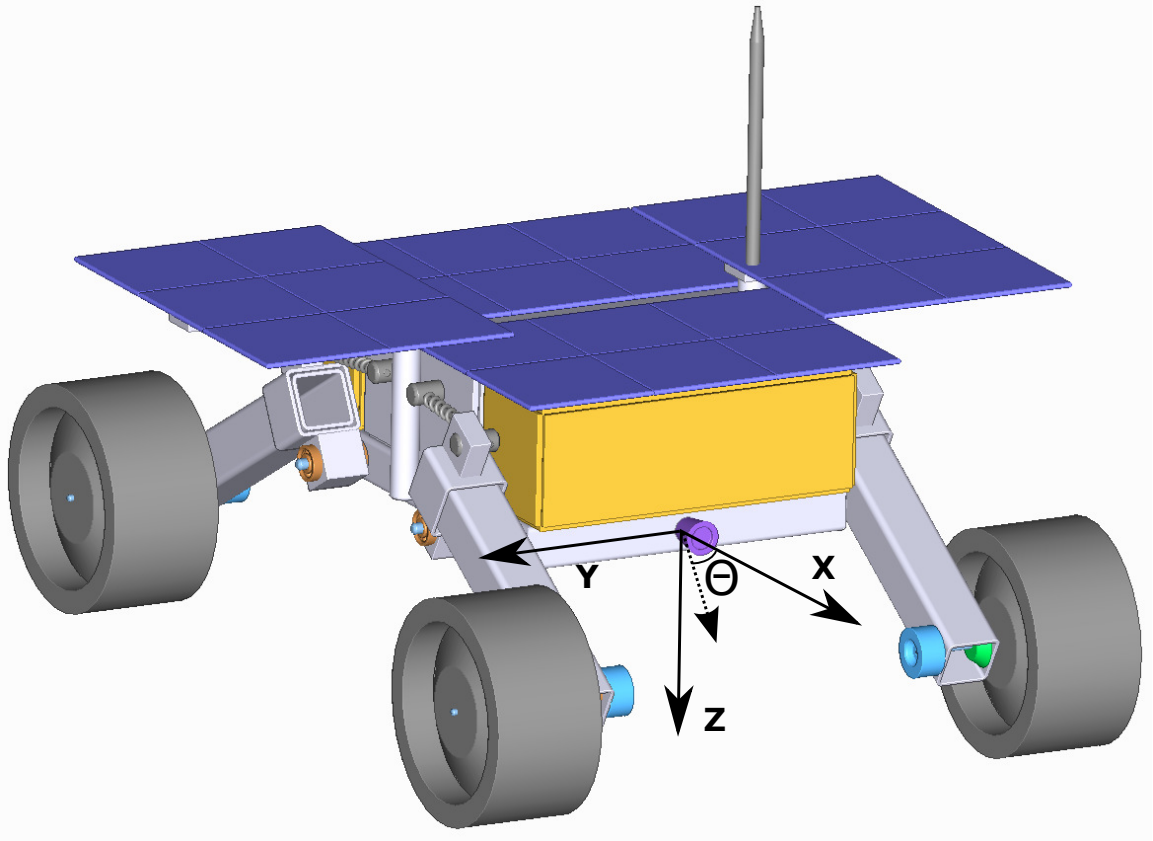


Figure 2.7: μrover model and coordinate systems

2.2 Suspension Kinematics

The μrover uses its four independently-driven wheels to apply the drive wheel forces f_{m1}, f_{m2}, f_{m3}, f_{m1} due to corresponding wheel torques speeds τ_{m1}, τ_{m2}, τ_{m3}, τ_{m4}. To derive practical control quantities, the the wheel rotation speeds are defined with respect to the motors in a right-handed manner so that counterclockwise is considered positive. The wheel forces are defined with respect to the suspension arms such that positive is away from the center of mass of the rover. Finally, the parameterization for forward velocity and steering is defined with respect to the body frame such that forward movement and right-hand turns are positive. This is necessary so that the individual control of each aspect of rover movement can be dealt with compatibly, but care must be taken when mapping one set of variables to another.

The geometry of the body and wheels is shown in top view in Figure 2.8. The μrover is differentially ("skid") steered, so heading rotation θ and rotational velocity ω in the horizontal (ideally parallel to the ground) plane is accomplished by changing the forces applied by the motors on one side relative to the other. Some wheel slip is necessary for efficient differential steering, so steering is easiest in sand or other loose media. Any difference in velocity between the front and rear wheels will cause a change in suspension geometry, so motor speeds must be controlled to maintain a desired geometry.

Without movement, the μrover kinematics follow conventional differential steering models [Chen *et al.* 2011]. Given a distance d_w between wheel centres, a wheel radius of r_w, a wheel slip coefficient μ_w and wheel speeds for the left side and right side wheels ω_{ml} and ω_{mr}, the forward motion and steering variables take the form

$$\dot{x} = v = r_w \frac{\omega_{mr} + \omega_{ml}}{2} \tag{2.5}$$

$$\dot{\theta} = \omega = \frac{r_w(\omega_{mr} - \omega_{ml})}{\mu_w d_w}. \tag{2.6}$$

and the rover state variables with respect to the ground coordinate system (x_g, y_g, z_g) are defined

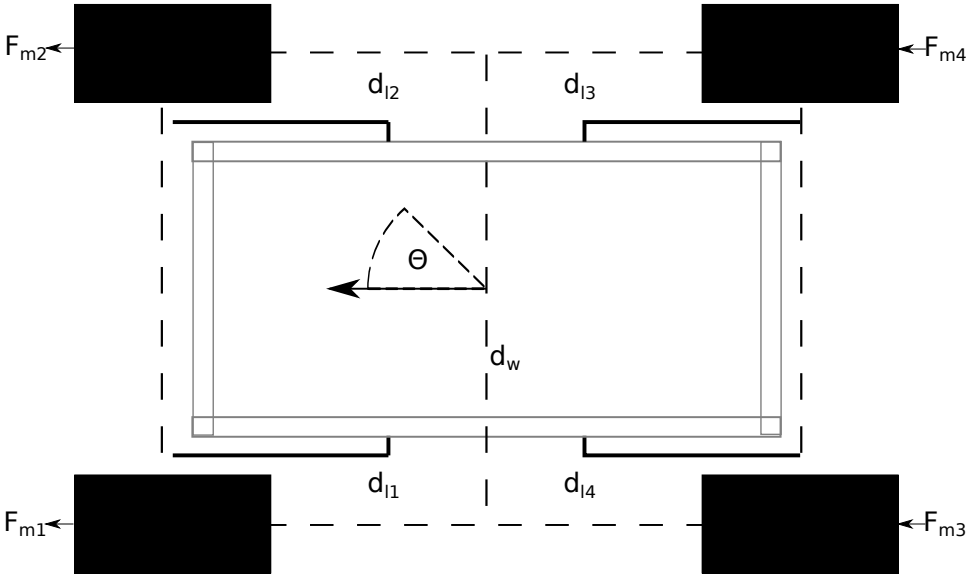


Figure 2.8: Kinematics in the horizontal plane for the μrover chassis

as

$$\dot{x_g} = v\cos(\theta) \tag{2.7}$$

$$\dot{y_g} = v\sin(\theta). \tag{2.8}$$

Since the wheel rotation speeds ω_m for each motor are defined as positive-counterclockwise, the mapping between wheel torques and wheel forces can be related as

$$\begin{bmatrix} f_{m1} \\ f_{m2} \\ f_{m3} \\ f_{m4} \end{bmatrix} \equiv \begin{bmatrix} \tau_{m1}/r_w \\ -\tau_{m2}/r_w \\ \tau_{m3}/r_w \\ -\tau_{m4}/r_w \end{bmatrix}. \tag{2.9}$$

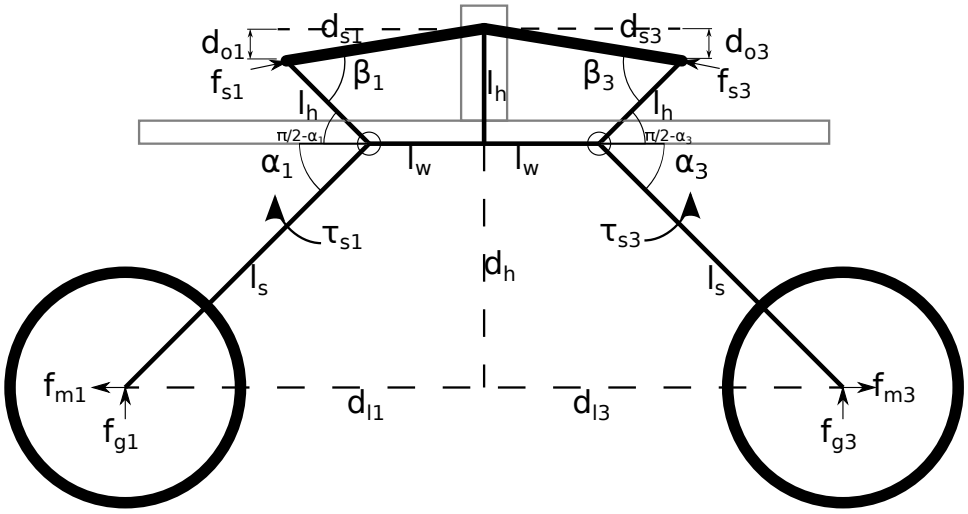


Figure 2.9: Kinematics in the vertical plane for a μrover half-chassis

2.2.1 Planar Suspension Model

Limiting the rover model to the 2-D vertical plane, a simple diagram of the suspension kinematics is shown in Figure 2.9. This model can be used to determine the spring force required for suspension of the μrover and the reaction of the suspension geometry to the motors. The current dimensions are $l_w = l_h = 50mm$, $l_s = 150mm$, and a wheel diameter of $100mm$.

The movement of the four independently-powered wheels is controlled by swing arms that pivot about the horizontal axis. Travel of the swing arms is limited to 45° in α from the horizontal plane. A spring slider connected to each swing arm provides counterforce against the force of gravity, so the swing arms remain by themselves about a position of equilibrium depending on the height of the ground under each wheel. The swing arm attachment height to the spring is made to be adjustable, but the best spring positioning for mounting on the chassis and the simplest construction of the spring mount is obtained when l_h is used, such that the offset height of the spring attachment $d_o = 0$ when $\alpha = 0$, as shown in Figure 2.10. The angles α are actively monitored by the drive controller, and can be used for feedback of the suspension geometry.

Geometrically, the ground clearance d_h of the suspension is determined by $l_s \sin(\alpha)$. The equilibrium point of the suspension with respect to gravity is determined by the spring constant k_s and the length of the spring d_s. As a wheel is lifted by a change in the height of the ground

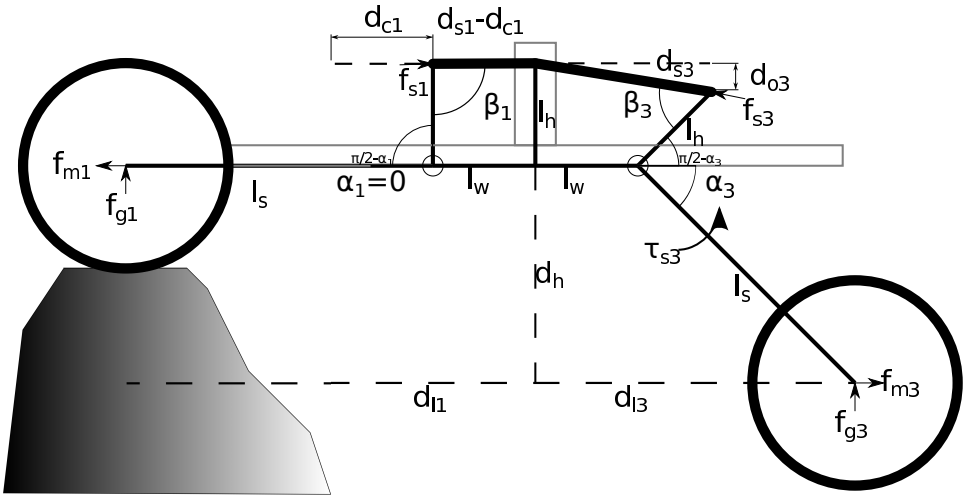


Figure 2.10: Kinematics in the vertical plane with one swing arm at maximum vertical travel

underneath, the spring is compressed by a distance d_c from its uncompressed point. At $\alpha = 45°$, the offset height of the spring d_o will be $l_h(1 - \cos(\alpha)) = 14.6mm$. The complementary angle to α of $(\pi/2 - \alpha)$ and the additional angle caused by the offset of $d_o = l_h(1 - cos(\alpha))$ is $\arcsin(d_o/(d_s - d_c)) = \arccos(l_h \sin(\alpha)/(d_s - d_c))$, comprise the complete angle to the spring of

$$\beta = \pi/2 - \alpha + \arccos\left(l_h \frac{\sin(\alpha)}{(d_s - d_c)}\right). \tag{2.10}$$

For reducing this to the relationship between d_c and α, it is desirable to simplify things with an approximation. We can use trigonometry about the swing arm joint and approximate the compressible part of the spring as a chord of a circle as $2l_h \sin(\alpha/2)$. Given that the force applied by the spring depends on d_c, which can only vary between $0 < \alpha < \pi/4$, we will assume that the maximum (uncompressed) length of the spring is $2l_h \sin(\pi/8)$ and approximate the compression d_c as

$$d_c \approx 2l_h \left[\sin\left(\frac{\pi}{8}\right) - \sin\left(\frac{\alpha}{2}\right)\right]. \tag{2.11}$$

Rearranging Equation 2.11 and considering ground clearance, we can obtain

$$\alpha = \arcsin\left(\frac{d_h}{l_s}\right) \approx 2\arcsin\left[\sin\left(\frac{\pi}{8}\right) - \frac{d_c}{2l_h}\right]. \tag{2.12}$$

2.2.2 Spring Compression

For the moment, the supporting surface is assumed to be locally flat. As the suspension swing arms are supported by springs, the position of each arm is determined by a force balance between the gravitationally-induced forces f_g normal to the supporting surface, the force f_m tangential to the supporting surface, and the forces f_s on the suspension spring. The motor force (assumed to be away from the rover body) causes a net torque on the swing arm of $\tau_s = l_s(f_m \sin(\alpha) + f_g \cos(\alpha))$. For static equilibrium, this must be balanced by the force $k_s d_c$ from the spring, which exerts a torque of $l_h k_s d_c$, so the spring compression distance for each swing arm is

$$d_c = \frac{l_s(f_m \sin(\alpha) + f_g \cos(\alpha))}{l_h k_s}. \tag{2.13}$$

From trigonometry, the spring will have a length of l_w at full compression and extend an additional $l_h \sin(\alpha)$ in the horizontal and drop $l_h(1 - \cos(\alpha))$ in the vertical as it extends, so the net spring length $d_s - d_c$ can be found as

$$d_s - d_c = l_h \sqrt{(1 - \cos(\alpha))^2 + \left(\frac{l_w}{l_h} + \sin(\alpha)\right)^2}. \tag{2.14}$$

Given that in this case, we are using $l_h = l_w$

$$d_s - d_c = l_h \sqrt{(1 - \cos(\alpha))^2 + (1 + sin(\alpha))^2} = l_h \sqrt{2(\sin(\alpha) - \cos(\alpha)) + 3}. \tag{2.15}$$

Combining Equations 2.13 and 2.15 gives us a specification for the length of spring required given a spring constant k_s as

$$d_s = \frac{l_s(f_m \sin(\alpha) + f_g \cos(\alpha))}{l_h k_s} + l_h \sqrt{2(\sin(\alpha) - \cos(\alpha)) + 3}. \tag{2.16}$$

2.2.3 Spring Selection

Estimating the spring constant required for the suspension is an important part of any suspension design. Under the temporary assumption that each wheel supports one quarter of the μrover weight and no motor input is provided, with rover mass m_r of the μrover in gravity g_r each of the four separate suspension arms must support $f_g = \frac{1}{4} m_r g_r$, we can solve for the spring constant considering the resting spring compression becomes

$$k_s = \frac{l_s(m_r g_r \cos(\alpha))}{4 l_h d_c}. \tag{2.17}$$

Substituting 2.17 into Equation 2.12 for d_c, and then into the relation for ground clearance, we can estimate the resulting d_h as

$$d_h \approx l_s \sin\left(2 \arcsin\left[\sin\left(\frac{\pi}{8}\right) - \frac{l_s(m_r g_r \cos(\alpha))}{8 l_h^2 k_s}\right]\right). \tag{2.18}$$

2.2.4 Orientation

To extend this into three dimensions starting with the model in Figure 2.9, the height above ground level for each suspension arm is at the pivot point is calculated as $l_s \sin(\alpha) + r_w$. At each side of the μrover, the midpoint of the body with respect to the front and back is where d_h is measured. This midpoint is equidistant at l_w from the suspension arm pivots, and therefore is also halfway between them vertically. For the left and right sides of the μrover with wheel/suspension numbering as in Figure 2.8, the heights $d_{h,left}$ and $d_{h,right}$ are calculated this way. Similarly, the height at the central point of the chassis $d_{h,center}$ is at the centroid between all four pivot points.

$$d_{h,left} = \frac{l_s}{2}(\sin(\alpha_1) + l_s \sin(\alpha_3)) + r_w \tag{2.19}$$

$$d_{h,right} = \frac{l_s}{2}(\sin(\alpha_2) + l_s \sin(\alpha_4)) + r_w \tag{2.20}$$

$$d_{h,center} = \frac{l_s}{4}(\sin(\alpha_1) + \sin(\alpha_2) + \sin(\alpha_3) + \sin(\alpha_4)) + r_w \tag{2.21}$$

The angle of tilt ψ of the μrover's chassis about the x axis can be determined by the difference in heights as well, as

$$\psi = \arctan\left(\left|\frac{d_w}{d_{h,left} - d_{h,right}}\right|\right). \tag{2.22}$$

2.2.5 Parameterized Control

For control purposes, it is useful to parameterize the control of the μrover's suspension into intuitive quantities. The most common parameterization of control for differentially steered vehicles is movement in the forward and reverse directions in the body frame v and rotation about the z axis ω, so that setting any combination of (v, ω) can produce movement in two dimensions. With four wheels, the set of wheel torques applied $\tau_{\mathbf{m}} = \{\tau_{m1}, \tau_{m2}, \tau_{m3}, \tau_{m4}\}$ is a combination of a calibrated scale factor k and a sign for each factor given the movement desired ς. Using this method, both linear movement and turning can be combined so that any combination of v and ω is possible. The right-side wheels must have negative angular velocity during forward movement, so a forward velocity v will require torques

$$\tau_{\mathbf{m}}(v, \varsigma_v, k) = \{\varsigma_{v1}kv, \varsigma_{v2}kv, \varsigma_{v3}kv, \varsigma_{v4}kv\} = \{+kv, +kv, -kv, -kv\}. \tag{2.23}$$

Turning right, with positive ω, requires simply that all four wheels have positive velocity, so

$$\tau_{\mathbf{m}}(\omega, \varsigma_\omega, k) = \{\varsigma_{\omega1}k\omega, \varsigma_{\omega2}k\omega, \varsigma_{\omega3}k\omega, \varsigma_{\omega4}k\omega\} = \{+k\omega, +k\omega, +k\omega, +k\omega\}. \tag{2.24}$$

For control of the passively-actuated suspension, though, all four motors must be used in different combinations to produce four degrees of freedom. Direct control of each motor is possible with the parameter set $(\omega_{m1}, \omega_{m2}, \omega_{m3}, \omega_{m4})$, and the controller allows individual commands to be sent to each motor. However, it is preferred to allow the embedded controller to manage the suspension arm positions and balance of the rover automatically while allowing the rover to move and steer as commanded. Two additional parameters are therefore added to v and ω. Heightening, or "up" u is used to control ground clearance by increasing and decreasing the suspension

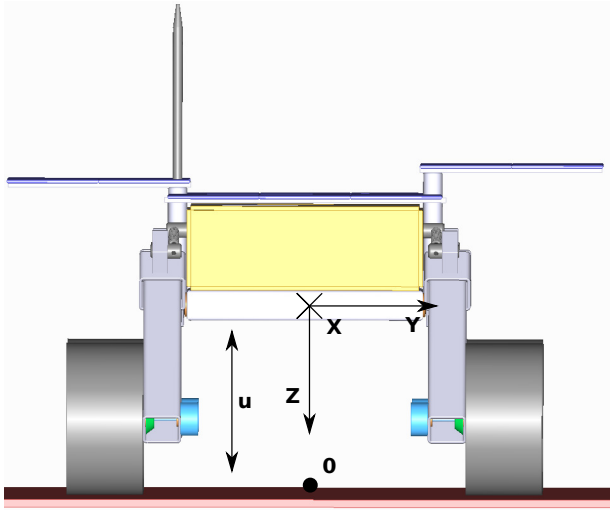


Figure 2.11: Effect of height u parameter on μrover suspension

angles α_i. This has the added benefit of making the front and back wheels closer together, which makes turning easier due to less wheel slip against the wheel travel directions. The effect of the u parameter is illustrated in Figure 2.11. From Figure 2.9 and Equation 2.13, it can be seen that applying $f_m > 0$ to all four wheels will decrease the angles α_i and applying $f_m < 0$ to all four wheels will increase α_i, so the torques required for lifting the rover are

$$\tau_{\mathbf{m}}(u, \varsigma_u, k) = \{\varsigma_{u1}ku, \varsigma_{u2}ku, \varsigma_{u3}ku, \varsigma_{u4}ku\} = \{-ku, +ku, +ku, -ku\}. \tag{2.25}$$

On a horizontal surface under normal conditions, the angle ϕ of the rover body about the y axis is always 0. The reason is that with equal spring constants on all four suspension arms, the angles α opposing each other front and back will tend to stay equal, that is $\alpha_1 = \alpha_3$ and $\alpha_2 = \alpha_4$. While there is no active method to adjust ϕ on the ground, any unevenness in terrain will cause the suspension to move in response. The general rule for passively sprung suspension is that the total potential energy contained in the springs will remain minimized.

The remaining parameter must control the bias between left and right wheels, and is called tilt ψ. By raising the suspension on one side of the rover and lowering the other side, the rover body can be tilted to the side, and for compatibility with the right-handed coordinate system, ψ is defined to be positive for clockwise rotation of the body while looking forward. The effect of the

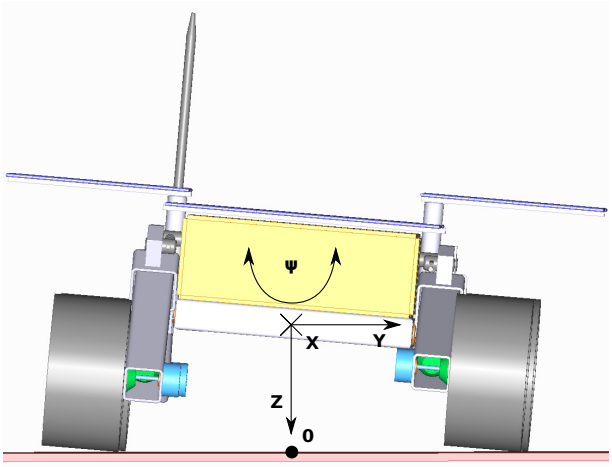


Figure 2.12: Effect of tilt ψ parameter on μrover suspension

tilt parameter is illustrated in Figure 2.12. This is particularly valuable for controlling body tilt on slopes, where the body should be kept as level as possible to avoid tipping over. The torques required for tilt are again orthogonal by sign to those defined by the other parameters, and are

$$\tau_{\mathbf{m}}(\psi, \varsigma_{\psi}, k_{\psi})) = \{\varsigma_{\psi 1} k\psi, \varsigma_{\psi 2} k\psi, \varsigma_{\psi 3} k\psi, \varsigma_{\psi 4} k\psi\} = \{-k\psi, +k\psi, -k\psi, +k\psi\}. \tag{2.26}$$

The four parameters (v, ω, u, ψ) are particularly important as they form an orthogonal basis with respect to the signs of the torques they apply. As there are only four degrees of freedom in the system, there is a single, unambiguous change of coordinates from the set (v, ω, u, ψ) to the set $\{\tau_{m1}, \tau_{m2}, \tau_{m3}, \tau_{m4}\}$. Therefore, any combination of torques $\tau_{m1}, \tau_{m2}, \tau_{m3}, \tau_{m4}$, can be created as a linear combination of (v, ω, u, ψ). For purposes of control, it is useful to define a direct mapping between motor torques and control parameters. The addition of all torque contributions from equations 2.23, 2.24, 2.25, 2.26 for each parameter results in a system of equations for the desired motor torques using a mapping matrix $\mathbf{J}$ as

$$\tau_{\mathbf{m}} = \begin{bmatrix} \tau_{m1} \\ \tau_{m2} \\ \tau_{m3} \\ \tau_{m4} \end{bmatrix} = \begin{bmatrix} k\omega - k\psi - ku + kv \\ k\omega - k\psi + ku - kv \\ k\omega + k\psi + ku + kv \\ k\omega + k\psi - ku - kv \end{bmatrix} = k \begin{bmatrix} +1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 \\ -1 & +1 & +1 & -1 \\ -1 & +1 & -1 & +1 \end{bmatrix} \begin{bmatrix} v \\ \omega \\ u \\ \psi \end{bmatrix} = k\mathbf{J} \begin{bmatrix} v \\ \omega \\ u \\ \psi \end{bmatrix}. \tag{2.27}$$

This allows the distribution of torques required from the motors to be calculated given a set of desired parameters (v, ω, u, ψ). Note that this matrix is symmetric, which reflects the completeness of the parameter space and makes it an ideal basis for coordinate transformation. Due to the symmetry of transformation, it is also possible to obtain the parameter set that generates a given a set of motor torques from the inverse of the mapping matrix $\mathbf{J}$ as

$$\begin{bmatrix} v \\ \omega \\ u \\ \psi \end{bmatrix} = \frac{\mathbf{J}^{-1}}{k} \begin{bmatrix} \tau_{m1} \\ \tau_{m2} \\ \tau_{m3} \\ \tau_{m4} \end{bmatrix}. \tag{2.28}$$

2.2.6 Orientation Representation

To obtain a comprehensive model for the movement of the μrover, the movement and rotation of the body in three dimensions relative to the ground must be considered. For this model, we will neglect the coupling between the suspension geometry and the ground friction or terrain features as the terrain and friction coefficients are constantly changing and the interactions are complex and nonlinear. Instead, we will assume that the suspension controller will act to maintain stability and traction regardless of the terrain. For purposes of positioning relative to the environment, we will also assume that the ground is horizontal and flat under the rover, and place the origin of the body coordinate system on the plane of the ground under the center of volume of the rover body. A Cartesian coordinate system is used, and in keeping with the most common convention of body-frame coordinates for mobile vehicles, the x axis is defined as straight forward (the

direction a wheeled vehicle most often rolls), the y axis is defined as orthogonally to the right of the x axis such that the $x-y$ plane is coplanar with the ground or defined horizontal plane of the body, and the z axis is defined as straight down, to form a right-handed coordinate system with respect to x and y [Guowei Cai 2011]. Using this system, "normal" movement of the vehicle is a velocity $\dot{x}$ in the x direction, an increase in pitch corresponds to an "up" increase in ϕ, and a right-hand turn results in positive angular velocity $\dot{\theta}$ about the z axis in the first quadrant of the $x-y$ plane, which is easy and intuitive to remember. It may seem non-intuitive to have the z axis pointing "down" instead of "up" but preserving the right-handedness of the system in this way is beneficial in the long term. These axes are shown in Figure 2.7.

The traditional (and most intuitive) method for representing angular orientation in three dimensions is using three Euler angles defined with reference to the axes of the coordinate system. In this case, the body angles (θ, ϕ, ψ) about the z, y, and x axes respectively form the set of angles. The term "Euler angles" is often used to represent any triple of angular rotations, but in fact this is an over-generalization, as proper Euler angles are defined with reference to the line of nodes created after a rotation, and axis-referenced angle triples as we describe them are more correctly known as Tait-Bryan or Cardan angles. Due to the common use of the term and to avoid confusing readers, we will continue to refer to them as "Euler Angles". The fundamental problem with using triples of angles, though, is that after a rotation, the body is effectively in a new coordinate system, and after two rotations, the original axis of rotation is no longer where it was to begin with. For incremental rotations and translations measured in the body frame (e.g. motion measured relative to body-mounted sensors) this is appropriate behaviour, as any transformation applied within the body frame is simply an additional multiplication by a rotation matrix without regard to the world coordinate system. However, for placing the vehicle in a relative position to world coordinate systems, such as a map or global reference point, Euler angles have several shortcomings.

Using rotation matrices, this is equivalent to premultiplying a body-oriented vector in the world frame $\mathbf{X}$ by separate rotations about the x, y, z axes in order as $\mathbf{R_z R_y R_x X}$. Because matrix multiplication is not commutative, further rotation using premultiplication by the predefined

rotation matrix $\mathbf{R_x}$ will not lead to the desired result of rotation about the global x axis because $\mathbf{R_xR_zR_yR_xX} \neq \mathbf{R_zR_yR_xR_xX}$, as the result would be if two rotations about the x axis were used originally. For the same reason, a different sequence of rotations such as z, x, y will produce a different result, which is almost never desirable. Contrary to a common misconception, this is not the "gimbal lock" problem. The solution from a programming standpoint, of course, is to use the Euler angles for reference at each time point and predefine the order of axes to rotate around. While many different standards for rotation sequences exist (e.g. proper Euler angles use only two axes such as z, x, z because of non-commutativity), for work with aerospace vehicles and rovers the sequence z, y, x is used almost universally, along with the aerospace body frame coordinate axis definitions for yaw, pitch, and roll [Koks 2008]. The sequence of multiplication is therefore yaw first (θ about z), then pitch (ϕ about y) and then roll (ψ about x), resulting in the easy-to-remember multiplication $\mathbf{R_xR_yR_zX}$. The matrices used for rotation about the axes are easily defined from the Euler angles as

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix} \tag{2.29}$$

$$\mathbf{R_y} = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \tag{2.30}$$

$$\mathbf{R_z} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.31}$$

Despite the simplicity of the rotation matrix formulation, obtaining the angles back from a given rotation matrix is more complex due to singularities and ambiguity in the arctangent function, though the common atan2 function can be used as an alternative in the latter case. Another undesirable consequence of the use of Euler angles is the presence of singularities as a

result of coupling three orthogonal rotation angles. For example, if the vehicle is headed straight forward at $\theta = 0$ and pitched vertically up to $\phi = \pi/2$ (90°), the heading angle θ becomes numerically undefined due to the co-linearity of two rotation axes causing non-unique solutions, and for $\phi > \pi/2$ or $\phi < -\pi/2$ the heading angle becomes $\theta = \pi$ (180°), experiencing a sudden discontinuity if global consistency of the heading angle is to be maintained with respect to global coordinates. This problem occurs identically between any two Euler angles, and requires that at least one of a pair of angles be defined on the range $-\pi/2 < \phi < \pi/2$. Such numerical discontinuities must be avoided for control purposes, as the lack of distinct angular values makes it complicated to compare any two attitude measurements differing by more than 90° in one axis. This generally known as "gimbal lock" because historically, when gyroscopes and inertial measurement units still used gimbals for angular measurement and actuation, the outer gimbal axis could rotate to be parallel to the inner gimbal axis, at which point the gimbal would be "locked" with respect to rotation about the now-coplanar gimbal axes [Hoag 1963]. While the problem with numerical singularities is not physically identical, it is similar enough that the term has become well-known.

For these reasons, local sensor estimation and vision within the body coordinate system is done using Euler angles, as it is generally much simpler and more intuitive to use Cartesian relations to orthogonal angles for sensors, and most sensors only deal with one or two relative angles at a time. However, orientation of mobile coordinate systems such as the μrover body frame with respect to a fixed world coordinate system or map is done using quaternions. Quaternions were conceived by Hamilton as an extension to complex numbers that would allow three-dimensional coordinate "triples" to be multiplied together, and take the form of "quadruples" of real numbers. While an entire branch of mathematics has been devoted to Hamilton's quaternion algebra, quaternions as we know them today are most recognized and widely used for the representation of orientation in three dimensions without gimbal lock [Kuipers 2000].

A quaternion can be defined as $\mathbf{q} = (w, x, y, z)$, where x, y. and z are the "vector" parts of the quaternion associated with three-dimensional space and w is the "scalar" part that provides the extra degree of freedom that makes quaternion algebra possible, much as in rotation matrices.

One consequence of this is that all quaternions that are scalar multiples of each other represent the same orientation, and only the quaternion length $|\mathbf{q}| = w^2 + x^2 + y^2 + z^2$ differs, so typically (q) is normalized after each operation to give the unit quaternion $\mathbf{q}/|\mathbf{q}| = \hat{\mathbf{q}}$. Note that $-\hat{\mathbf{q}}$ is also a unit quaternion representing the same rotation. Quaternion conversions also avoid some of the singularity and ambiguity problems present in Euler angle conversions. A unit quaternion representing a rotation Θ about a given unit axis $\hat{\mathbf{A}} = (a_x, a_y, a_z)$ can be constructed by

$$\hat{\mathbf{q}}(\Theta) = \left(\cos\left(\frac{\Theta}{2}\right), a_x \sin\left(\frac{\Theta}{2}\right), a_y \sin\left(\frac{\Theta}{2}\right), a_z \sin\left(\frac{\Theta}{2}\right)\right). \tag{2.32}$$

Also, like rotation matrices, quaternions can be multiplied together to represent additive rotation. The identity unit quaternion for multiplication is $\hat{\mathbf{q}}(I) = (1, 0, 0, 0)$. Using the simplified model of matrix premultiplication as above, a quaternion multiplication $\hat{\mathbf{q}} = \hat{\mathbf{q}}_\mathbf{1}\hat{\mathbf{q}}_\mathbf{2}$ can be calculated from

$$\hat{\mathbf{q}} = \begin{pmatrix} w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2, \\ w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2, \\ w_1y_2 + y_1w_2 + z_1x_2 - x_1z_2, \\ w_1z_2 + z_1w_2 + x_1y_2 - y_1x_2 \end{pmatrix}. \tag{2.33}$$

One downside of this use of quaternion representation is that, unlike Euler angles, a unit quaternion's values do not have a direct physical interpretation (i.e. you cannot determine what rotations are represented simply by looking at them). For numerical visualization of a quaternion as an orientation, it should generally be converted back to Euler angles. This can be done without too much complexity using the arcsin and quadrant-aware arctan 2 function as

$$\begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan 2(2(wz + xy), 1 - 2(y^2 + z^2)) \\ \arcsin(2(wy - xz)) \\ \arctan 2(2(wx + yz), 1 - 2(x^2 + y^2)) \end{bmatrix}. \tag{2.34}$$

Figure 2.13: Unbalanced turning of μrover on high-friction surface

2.3 Controller Design

Since the suspension allows passive control of ground clearance and limited rotation about the x axis, there is more control over body positioning than with most differentially steered vehicles. However, due to the need for wheel slip during turns and the extra degrees of freedom in the suspension system, there is also a noticeable amount of coupling between the ground and the suspension geometry. This often manifests itself as undesired tilting toward the outside of turns and "hopping" of the wheels if too much friction is present. Figure 2.13 shows an example of this coupling. In this case, if the degree of wheel slip is not sufficiently high or the terrain is unbalanced, a wheel driving forwards has the potential to raise the chassis and lift the other wheel on the same side off the ground. While turning on three wheels is actually more efficient than overcoming the friction for differential steering on four, it creates a higher potential for tip-over because the chassis sits higher on only three points of contact, as well as causes uncontrolled motion.

Another key focus of the controller is the fact that differentially-steered vehicles are in general easier to turn when the front and back wheels are closer together. Wheels can only provide a vector component of force that is aligned with the intersection of their plane of rotation and the plane of the surface they are in contact with. Consequently, a wheel is most efficient when driving directly forward or backward. Any energy expended that is not in the forward or back-

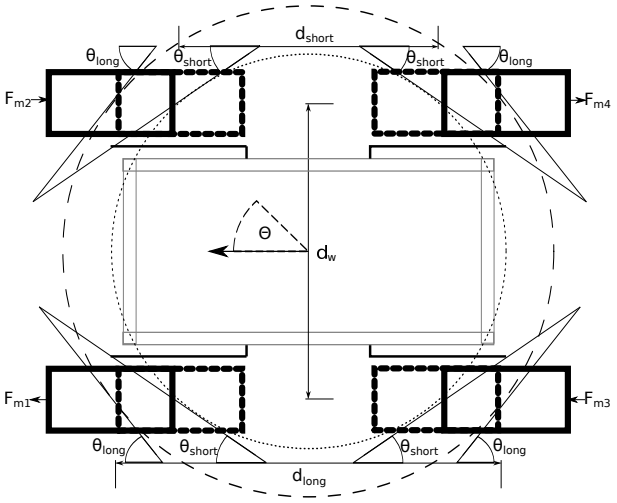


Figure 2.14: Effect of raising and lowering the α angle of the suspension on stationary turning

ward direction is generally lost to friction. But differential steering implicitly requires that some sideways movement be present, as the wheels will not be rotated to align with the direction of movement in a turn, and therefore energy will be lost in overcoming friction in that turn. Figure 2.14 illustrates this for a differential drive turn on the spot about the center of the body. Let the angle between the plane of the wheel's rotation and the movement of the ground be ϑ. The in-line force component $\cos(\vartheta)$ provides rolling movement, but the perpendicular force component $\sin(\vartheta)$ must overcome friction for the wheels to travel in a circular path about the center of rotation of the body, and represents lost energy. When α is small and the wheelbase is effectively longer, the angle between the plane of wheel rotation and the relative movement of the ground becomes ϑ_{long}. As α increases, the wheelbase becomes shorter and the angle with movement of the ground becomes ϑ_{short}. Based on the planar geometry of the suspension, $\vartheta_{long} > \vartheta_{short}$ and $\vartheta < \pi/2$, therefore $\sin(\vartheta_{long}) > \sin(\vartheta_{short})$ less energy is lost with a short wheelbase.

Any controller for differential-drive steering must also compensate for environmental effects such as varying wheel slip and intermittent contact on uneven terrain. The most basic form of this is to ensure that each wheel maintains a constant angular velocity by adjusting the motor torque. Taking the above factors into consideration, the goals of the motor control system can be summarized as:

- Set and maintain the desired velocity v as instructed by the navigation system
- Set and maintain the turn rate ω as instructed by the navigation system
- Adjust the ground clearance u to be high for turns and low for obstacles and straight movement to minimize turnover risk
- Adjust the tilt angle ψ to compensate for ground inclines to the left and right (as measured by accelerometer)

2.3.1 PID Control

To improve the movement control of the μrover, a basic Proportional-Integral-Derivative (PID) controller was implemented using 8-bit integer arithmetic on an AVR microcontroller for controlling wheel angular velocity. A separate PID loop is used for each one of the four wheel motors, and the inputs are the desired motor speeds $\omega_{\mathbf{m}}(k) = \{k_1\omega_{m1}, k_2\omega_{m2}, k_3\omega_{m3}, k_4\omega_{m4}\}$. For this case, no consideration is made for managing suspension geometry via wheel speed. It is a simple four-wheel differential drive controller intended for manual or automated motor speed control.

The traditional form of the PID controller applies a linear combination of weighted terms at each time step to obtain a control output. Three types of terms are available, and are derived from the error $e(t)$ at a given time t. The first is simply the weighted error itself, the second is the time integral of the error, and the third is the time derivative of the error. All three quantities are scaled by the scalar weightings k_p, k_i, and k_d. The traditional continuous-time PID controller with the output τ_c is then defined as

$$\tau_c(t) = k_p e(t) + k_i \int_0^t e(t)dt + k_d \frac{de(t)}{dt} \tag{2.35}$$

In this case, the control quantity is the wheel speed $\omega_{m,i}(t)$, measured by rotary encoder for each motor from $1 \ldots 4$ by dividing the number of encoder counts by the time step length Δt. The desired or "set" wheel speed for each motor is $\omega_{s,i}(t)$. The error in the wheel speed at time t is therefore obtained as $e(t) = \omega_{m,i}(t) - \omega_{s,i}(t)$. As the controller is implemented on a discrete-time

system with limitations on numerical size and precision, a modified approach is taken where the time t is considered to be an integer time step and the control output is scaled by the divisor k_s using the discrete-time formulation

$$\tau_c(t) = \frac{k_p e(t) + k_i \sum_0^t e(t)dt + k_d(e(t) - e(t-1))}{k_s} \tag{2.36}$$

Alternately, an implementation of the PID can operate on the parameterized state vector (v, ω, u, ψ) with $v = \dot{x}$, $\omega = \dot{\theta}$, and $u = \dot{h}$. In this case, the same four PID loops are used, but the errors are calculated as

$$e_v(t) = v - v_{desired} \tag{2.37}$$

$$e_\omega(t) = \omega - \omega_{desired} \tag{2.38}$$

$$e_u(t) = u - u_{desired} \tag{2.39}$$

$$e_\psi(t) = \psi - \psi_{desired}. \tag{2.40}$$

and the output for the motors is obtained using Equation 2.27.

With regard to the practicality of implementation on an 8-bit microcontroller, the input values $\omega_{m,i}(t)$ and $\omega_{s,i}(t)$ are calibrated in deciradians per second ($drad/s$), where $1drad = 0.1rad$. For an 8-bit value, this gives a useable range of $0.1 \ldots 25.5rad/s$, or about $0.95 \ldots 243.5RPM$, and is a reasonable range for a slow-moving autonomous vehicle. The weightings k_p, k_i, k_d, and k_s are also stored and selected as 8-bit values. However, as Equation 2.36 requires multiplication of variables, the error $e(t)$ and the three P, I, and D terms are stored as 16-bit values. An 8-bit timer-counter is used to control the fraction of power transferred to the motors, so the divisor k_s is necessary to scale the result back into the range of $0 \ldots 255$ so that the output $\tau_c(t)$ can map to an 8-bit value. It can be noted that because the error $e(t)$ depends only on the difference between desired and measured quantities, the output $\tau_c(t)$ can be scaled differently than the encoder measurements or desired input. Therefore, the weightings k_p, k_i, k_d, and k_s are chosen to maximize both the performance and the useable numerical range of the controller, and do not have to

explicitly convert units or quantities. For tuning, k_p, k_i, and k_d can be manually tuned while the controller is running, but should be kept low enough that each term remains within the range of $\pm 32,768$ even at the maximum error possible given the hardware. The scaling divisor k_s primarily has to prevent the entire PID sum from exceeding ± 255, although it is also useful for limiting the output of the controller. Setting $k_s < k_p$ for example, prevents the P term from having an effect at low error values since the implicit *floor* operation causes it to be zero until $e(t) > k_s/k_p$. Also, all required variables are checked when updated to ensure that it is not out of range for its data type to prevent unexpected sign changes and numerical instability. The error $e(t)$ particularly is restricted to ± 1024 with the expectation that the scaling parameters will usually be chosen in the range $0 \ldots 10$.

2.3.2 Body Dynamic Control

For dynamic analysis, it is preferable to describe the movement of the body in terms of specific angles and distances, following the suspension geometry and using conventional methods [Carlone *et al.* 2013]. We will denote these as the control quantities $\{\dot{x}_C, \dot{\theta}_C, \dot{h}_C, \dot{\psi}_C\}$ Applying Equation 2.6 provides the two-dimensional analysis, but with the modification that the distances travelled by the left and right side motors are averaged to determine the net amount of body travel for each side, so that $\omega_{ml} = (\omega_{m1} + \omega_{m3})/2$ and $\omega_{mr} = (\omega_{m2} + \omega_{m4})/2$.

$$\dot{x}_C = r_w \left(\frac{\omega_{m2} + \omega_{m4} + \omega_{m1} + \omega_{m3}}{4} \right) \tag{2.41}$$

$$\dot{\theta}_C = \omega = r_w \left(\frac{\omega_{m2} + \omega_{m4} - \omega_{m1} - \omega_{m3}}{2\mu_w d_w} \right). \tag{2.42}$$

To consider the rotation and movement of the body in the vertical plane, we use the height average of the suspension arms and calculate the tilt angle ψ from the relative heights of each side d_{hl} and d_{hr}, averaged between the front and back suspension arms. Assuming a nearly flat surface, the rotational motion of any of the four suspension angles $\dot{\alpha}$ can be related to the relative motion of the motor with respect to the suspension joint with sign dependent on the direction the

joint is facing with

$$\dot{\alpha}_C = \pm \frac{(r_w \omega_m)}{l_s \sin(\alpha)}. \tag{2.43}$$

This allows the definition of parameter rates for u and ψ using calculations of relative heights in the form of $\dot{d_{hl}} = l_s(\cos(\alpha_1)\dot{\alpha_1} + \cos(\alpha_3)\dot{\alpha_3})/2$ as

$$\dot{u}_C = l_s \left(\frac{\sin(\alpha_2) + \sin(\alpha_4) + \sin(\alpha_1) + \sin(\alpha_3)}{4} \right) \tag{2.44}$$

$$\dot{\psi}_C = \frac{\dot{d_{hl}} - \dot{d_{hr}}}{d_w \cos(\psi)} = l_s \left(\frac{\cos(\alpha_1)\dot{\alpha_1} + \cos(\alpha_3)\dot{\alpha_3} - \cos(\alpha_2)\dot{\alpha_2} - \cos(\alpha_4)\dot{\alpha_4}}{2 d_w \cos(\psi)} \right). \tag{2.45}$$

It is possible to control the pose of the μrover alone from changing the relative wheel speeds ω_{m1}, ω_{m2}, ω_{m3}, and ω_{m4} using Equations 2.42-2.45 and the measured angles α from the suspension sensors. It is simpler from a control point of view to consider this as managing the four suspension parameters v, ω, u, and ψ, which also form an orthogonal control set. The quantities measured by the motor encoders (integrated from ω_{m1}, ω_{m2}, ω_{m3}, and ω_{m4}), inertial measurement sensors (sun sensor or magnetometer), and suspension angle sensors (and kinematics) are

$$\mathbf{X_1} = [x, \theta, h, \psi]^T. \tag{2.46}$$

As the motors are controlled by speed, the control parameters are effectively

$$\mathbf{X_2} = \dot{\mathbf{X}}_\mathbf{1} = \left[\dot{x}, \dot{\theta}, \dot{h}, \dot{\psi}\right]^T. \tag{2.47}$$

We can rewrite the dynamic system with state function F and actuator function G which acts on control torque τ_c with external disturbances τ_d as

$$\begin{bmatrix} \ddot{x} \\ \ddot{\theta} \\ \ddot{h} \\ \ddot{\psi} \end{bmatrix} = F\left(\begin{bmatrix} x \\ \theta \\ h \\ \psi \end{bmatrix}, \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \dot{h} \\ \dot{\psi} \end{bmatrix}\right) + G\left(\begin{bmatrix} x \\ \theta \\ h \\ \psi \end{bmatrix}, \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \dot{h} \\ \dot{\psi} \end{bmatrix}\right)\tau_c + \tau_d. \tag{2.48}$$

2.3.3 Type-1 and Type-2 Fuzzy Adaptive Sliding Mode Controllers

For nonlinear control of the chassis and as a comparison to the PID control methodology, we design a nonlinear controller in the form of a fuzzy adaptive sliding mode controller based on prior work in nonlinear nanosatellite attitude control [Li *et al.* 2013a]. This work was done in collaboration with Li [Li 2012]. Two types of fuzzifier are available - type 1 and type 2 - with the main difference being the number of fuzzy rules required. We rewrite the dynamic system as

$$\begin{bmatrix} \dot{\mathbf{X}}_\mathbf{1}(t) \\ \dot{\mathbf{X}}_\mathbf{2}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{X_2}(t) \\ F(\mathbf{X}) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_4 \\ G(\mathbf{X}) \end{bmatrix}\tau_c(t) + \tau_d(t). \tag{2.49}$$

For this dynamic system, $\mathbf{X}(t) = \begin{bmatrix} \mathbf{X_1}(t) \\ \mathbf{X_2}(t) \end{bmatrix} \in \mathbb{R}^8$, $\mathbf{X_1}(t) \in \mathbb{R}^4$, $\mathbf{X_2}(t) \in \mathbb{R}^4$, $\tau_d(t) \in \mathbb{R}^8$, $F(X) \in \mathbb{R}^8$ and $\tau_c(t) \in \mathbb{R}^8$. It is assumed that the system function $G(X) \in \mathbb{R}^{4\times4}$ is an unknown smooth and bounded function

$$0 < g_0 I_m < G < g_1 I_m$$

where g_0 and g_1 are positive constants. For nonlinear sliding-mode control, we define a linear switching function which ensures that the reduced-order system on the sliding mode $S = 0$ is asymptotically stable, where $\iota \in \mathbb{R}^{4\times4}$ is a diagonal constant matrix with positive diagonal elements ι_i $(i = 1, 2, ...m)$. The switching function $S(t) \in \mathbb{R}^4$ is then

$$S(t) = \iota e(t) + \dot{e}(t), \tag{2.50}$$

and $e(t) \in \mathbb{R}^4$ is the tracking error defined as $e(t) = X_1(t) - x_d(t)$. Here $x_d(t) \in \mathbb{R}^4$ is the desired trajectory. $\ddot{x}_d(t)$ is the second derivative of $x_d(t)$.

2.3.4 Type-1 and Type-2 Fuzzy Logic System

Zadeh introduced type-2 fuzzy sets as a generalization of ordinary fuzzy sets [Zadeh 1975]. In a type-1 fuzzy set, the degree of membership for each point is a normal fuzzy number can have the range of [0, 1], and the number is a crisp number. A type-2 fuzzy set can have a membership function with uncertainty [Lin *et al.* 2010]. A type-1 fuzzy control system generally includes a fuzzifier, a rule base, an inference engine and a defuzzifier. For this type-1 fuzzy system, a Mamdani minimum inference engine, singleton fuzzifier and center average defuzzifier are chosen [Wang 1997]. A fuzzy system in general is a collection of if-then fuzzy rules that can be expressed as

$$\nu^k : \quad If\ x_1\ is\ W_1^k,\ And\ \cdots,\ And\ x_n\ is\ W_n^k,\ Then\ y\ is\ Z^k.$$

The output of the fuzzy system (using singleton fuzzification, product inference and center average defuzzification) can be written as

$$\vartheta^T \zeta = \frac{\sum_{l=1}^{P} \vartheta_F^l(X) \prod_{i=1}^{N} \mu^l_{Wi^l(x_i)}}{\sum_{l=1}^{P} \prod_{i=1}^{N} \mu^l_{Wi^l(x_i)}} \tag{2.51}$$

where P is the total number of fuzzy rules. The N Gaussian membership functions are $\mu_{W1^l}(x_1), ..., \mu_{WnN}(x_n)$, and ζ is a fuzzy basis function defined as

$$\zeta^l(x) = \left(\frac{\prod_{i=1}^{N} \mu_{Wi^l}(x_i)}{\sum_{l=1}^{P} \prod_{i=1}^{N} \mu_{Wi^l}(x_i)} \right). \tag{2.52}$$

To implement the adaptive fuzzy terminal sliding mode control law, type-1 fuzzy sets over the interval of x_i are defined. Seven Gaussian membership functions are used in the type-1 fuzzy

system for each variable W_i ($i = 1, 2, \cdots, 7$), defined as

$$\begin{aligned}
\mu_{W1}(x_i) &= (1 + \exp(5(x_i + 3 \times a)))^{-1}, \quad \mu_{W2}(x_i) = \exp\left(-\left(\frac{x_i + 2 \times a}{b}\right)^2\right) \\
\mu_{W3}(x_i) &= \exp\left(-\left(\frac{x_i + 1 \times a}{b}\right)^2\right), \quad \mu_{W4}(x_i) = \exp\left(-\left(\frac{x_i}{b}\right)^2\right) \\
\mu_{W5}(x_i) &= \exp\left(-\left(\frac{x_i - 1 \times a}{b}\right)^2\right), \quad \mu_{W6}(x_i) = \exp\left(-\left(\frac{x_i - 2 \times a}{b}\right)^2\right) \\
\mu_{W_7}(x_i) &= (1 + \exp(5(x_i - 3 \times a)))^{-1}
\end{aligned} \tag{2.53}$$

where a and b are different constant numbers designed according to x_i. The type-2 fuzzy sets are constructed using the same set of seven Gaussian functions, except that instead of only one function being used about one mean value $\bar{\mu_{Wi}}$, two mean values $\bar{\mu 1}_{Wi}$ and $\bar{\mu 2}_{Wi}$ are used and two functions superimposed (added) above one another.

The details of the constructed Type-2 and Type-1 membership functions are given in Table 2.1.

Table 2.1: Type-2 and Type-1 Fuzzy Membership Functions

Name	Type-2 mean $\bar{\mu 1}_{Wi}$ and $\bar{\mu 2}_{Wi}$	Type-1 mean $\bar{\mu}_{Wi}$
$\mu_{W_1}(x_i)$	2.5, 1.5	3
$\mu_{W_2}(x_i)$	2, 1	2
$\mu_{W_3}(x_i)$	1.5, 0.5	1
$\mu_{W_4}(x_i)$	0.5, −0.5	0
$\mu_{W_5}(x_i)$	−0.5, −1.5	−1
$\mu_{W_6}(x_i)$	−1, −2	−3
$\mu_{W_7}(x_i)$	−1.5, −2.5	−3

2.3.5 Adaptive Fuzzy Sliding Mode Control

A sliding mode control law can be derived using the sign of the switching function and the switching-type function H as

$$\tau_c = G(X)^{-1} \left[\ddot{x}_d - \iota\dot{e} - F(X) - \tau_d - H\text{sgn}(S)\right]. \tag{2.54}$$

System functions F, G, and disturbance τ_d are usually difficult to be obtained. The indirect adaptive fuzzy sliding mode control law is given as

$$\tau_c = \hat{G}(X \mid \vartheta_g)^{-1}\left[\ddot{x}_d - \iota\dot{e} - \hat{F}(X \mid \vartheta_f) - \hat{h}(S \mid \vartheta_h)\right] \tag{2.55}$$

where F, G and τ_d are replaced with type-1 or type-2 fuzzy system. For a type 1-fuzzy system, $\hat{F}(X \mid \vartheta_f) = \vartheta_f^T\zeta_f(X)$, $\hat{G}(X \mid \vartheta_g) = \vartheta_g^T\zeta_g(X)$ and $\hat{H}(S \mid \vartheta_h) = \vartheta_h^T\zeta_h(S)$ are used to approximate $f(X)$, $g(X)$, and the switching-type control law $H\text{sgn}(S)$. For a type 2-fuzzy system, the following equations are used to to approximate $F(X)$, $G(X)$, and the switching-type control law $H\text{sgn}(S)$.

$$\hat{F}(X \mid \vartheta_f) = \frac{1}{2}\left[\zeta_{fr}^T\zeta_{fl}^T\right]\begin{bmatrix}\vartheta_{fr}\\ \vartheta_{fl}\end{bmatrix} = \vartheta_f^T\zeta_f(X) \tag{2.56}$$

$$\hat{G}(X \mid \vartheta_g) = \frac{1}{2}\left[\zeta_{gr}^T\zeta_{gl}^T\right]\begin{bmatrix}\vartheta_{gr}\\ \vartheta_{gl}\end{bmatrix} = \vartheta_g^T\zeta_g(X) \tag{2.57}$$

$$\hat{H}(S \mid \vartheta_h) = \frac{1}{2}\left[\zeta_{hr}^T\zeta_{hl}^T\right]\begin{bmatrix}\vartheta_{hr}\\ \vartheta_{hl}\end{bmatrix} = \vartheta_h^T\zeta_h(S) \tag{2.58}$$

We define the optimal parameters of fuzzy systems as the following functions within the range D_x as

$$\vartheta_f^* = arg\min_{\vartheta_f\in\bar{\varpi}_F}\sup_{X\in D_x}\left|\hat{F}(X \mid \vartheta_f) - F(X)\right| \tag{2.59}$$

$$\vartheta_g^* = arg\min_{\vartheta_g\in\bar{\varpi}_G}\sup_{X\in D_x}\left|\hat{G}(X \mid \vartheta_g) - G(X)\right| \tag{2.60}$$

$$\vartheta_h^* = arg\min_{\vartheta_h\in\bar{\varpi}_H}\sup_{S\in D_x}\left|\hat{H}(S \mid \vartheta_h) - H(S)\right| \tag{2.61}$$

It is assumed that there exists an optimal fuzzy logic system that can approximate the nonlinear terms $F(X)$, $G(X)$ and the switching-type control law $H\text{sgn}(S_i)$ in Equation 2.55 such that

$$
\begin{aligned}
F(X) - F^*(X \mid \vartheta_f^*) &= \varpi_F(X) \\
G(X) - G^*(X \mid \vartheta_g^*) &= \varpi_G(X) \\
H(S) - H^*(S \mid \vartheta_h^*) &= \varpi_h(S)
\end{aligned} \tag{2.62}
$$

where ϖ_F, ϖ_G and ϖ_h are approximation errors and bounded in the compact set U_x, i,e., $\left\|\varpi_F\right\| \leq \bar{\varpi}_F$, $\left\|\varpi_G\right\| \leq \bar{\varpi}_G$ and $\left\|\varpi_h\right\| \leq \bar{\varpi}_h$. Approximation errors can be reduced by increasing the number of fuzzy rules.

2.3.6 Nonlinear Controller Design

The controller design involves the construction of a sliding surface containing tracking errors to ensure that the system is restricted to the sliding surface. It also involves the derivation of parameter adaptation laws and fuzzy logic feedback control gains that can drive the desired trajectory to the sliding surface and maintain it in the manifold. A nonlinear hyper-plane based sliding mode can provide a wide variety of design alternatives with fast and finite time convergence. For this, Equation 2.50 is redefined as

$$
S = \iota e + \dot{e} + K_0 e^{\frac{p}{q}} \tag{2.63}
$$

where $K_0 \in \mathbb{R}^{3\times3}$ is a constant, diagonal, positive-definite, control design matrix. and p and q are the positive odd integers ($p < 2q$).

$$
\dot{S} = \iota\dot{e} + \ddot{e} + K_0 \frac{p}{q} e^{\frac{p}{q}-1}\dot{e}. \tag{2.64}
$$

The adaptive fuzzy terminal sliding control law is then redefined as

$$
\tau_c = \hat{G}(X \mid \vartheta_g)^{-1}\left[\ddot{x}_d - E - \hat{F}(X \mid \vartheta_f) - \hat{h}(S \mid \vartheta_h) - K_1 S\right] \tag{2.65}
$$

where $K_1 = diag\{k_{11}, k_{22}, k_{33}\}$, $I = diag\{1, 1, 1\}$ and $E = \iota\dot{e} + K_0 \frac{p}{q} e^{\frac{p}{q}-1}\dot{e}$. This control term (Equation 2.65) is not well defined when the estimated matrix $\hat{G}(X \mid \vartheta_g)$ is singular. The matrix is generated online via the estimation of the parameters ϑ_g. In order to implement this control law, additional precautions have to be taken to guarantee that ϑ_g remains in the feasible region in which $\hat{G}(X \mid \vartheta_g)$ is non-singular. In order to overcome this problem, the control law is modified to be

$$\tau_{cf} = \hat{G}^T(X \mid \vartheta_g)\left[\varepsilon_0 I + \hat{G}(X \mid \vartheta_g)\hat{G}^T(X \mid \vartheta_g)\right]^{-1}[\ddot{x}_d - E - \hat{F}(X \mid \vartheta_f) - \hat{h}(S \mid \vartheta_h) - K_1 S] \quad (2.66)$$

where ε_0 is a small positive constant. The approximation of $G^{-1}(X \mid \vartheta_g)$ by the regularized inverse and unavoidable reconstruction errors of the unknown functions $F(X)$ and $G(X)$ will occur. A more robust control term for τ_{cf} defined as

$$\tau_{cf} = \varepsilon_0[\varepsilon_0 I + \hat{G}(X \mid \vartheta_S)\hat{G}^T(X \mid \vartheta_g)]^{-1}\left[\ddot{x}_d - E - \hat{F}(X \mid \vartheta_f) - \hat{h}(S \mid \vartheta_h) - K_1 S\right] \quad (2.67)$$

and is combined with the control law Equation 2.66 to give

$$\tau_c = \tau_{cf} - \tau_{cr} \quad (2.68)$$

The controller Equation 2.68 is the sum of two control terms: the robust control term τ_{cf} and a modified certainty equivalent control term τ_{cr}, where

$$\tau_{cr} = \frac{S\,\|S\|\left(\bar{\varpi}_F + \bar{\varpi}_G\left\|\tau_{cf}\right\| + \bar{\varpi}_h + \|\tau_0\|\right)}{g_0\|S\|^2 + \hat{\chi}} \quad (2.69)$$

where $\hat{\chi}$ is a design time varying parameter defined below. The adaptive parameters ϑ_f, ϑ_g, ϑ_h and design parameter $\hat{\chi}$ are updated by the adaptive laws

$$\dot{\vartheta}_f = \alpha\zeta_f S, \quad \dot{\vartheta}_g = \beta\zeta_g S\,\tau_{cf}, \quad \dot{\vartheta}_h = \varsigma\zeta_h S, \tag{2.70}$$

$$\dot{\hat{\chi}} = -\kappa_0 \frac{\|S\|\left(\bar{\varpi}_F + \bar{\varpi}_G \left\|\tau_{cf}\right\| + \bar{\varpi}_h + \|\tau_0\|\right)}{g_0 \left\|S^2\right\| + \hat{\chi}} \tag{2.71}$$

where $\alpha > 0, \beta > 0, \varsigma > 0, \kappa_0 > 0, \hat{\chi}_0 > 0$. Also, $2q > p$ are used to avoid singularities. We can now consider the rover system Equation 2.49, and a Type-2 fuzzy terminal sliding control law defined by Equations 2.66-2.67 with adaptive control laws given by Equations 2.70 and 2.71. This guarantees the following properties:

1. All signals that may be due to disturbances and changes in the closed-loop system are bounded.

2. The tracking error is UUB (Uniformly Ultimately Bounded), meaning that it converges to the neighbourhood of zero by appropriately choosing the design parameters.

The proof of this is obtained with $\tau_{c1}(\bar{X}) = \tau_{cf}(\bar{X}) - \tau_{cr}(\bar{X})$ as a control law that considers disturbances, and $\tau_{c2}(X) = \tau_{cf}(X) - \tau_{cr}(X)$ as the control law without considering disturbances. Using the mean value theorem to bound the nonlinear vector, we have

$$\|G(\tau_{c1} - \tau_{c2})\| \leq g_m \left\|\bar{X} - X\right\| \tag{2.72}$$

where $\bar{X}$ is the mean of X and g_m is a bounding constant number. Rewriting Equation 2.64, the following equation is obtained.

$$\begin{aligned} \dot{S} &= \ddot{X} - \ddot{x}_d + E = -\ddot{x}_d + E + F(X) + G(X)(\tau_{c1}) + \tau_d \\ &= -\ddot{x}_d + E + F(X) + G(X)(\tau_{c1} - \tau_{c2}) + G(X)(\tau_{c2}) + \tau_d \\ &\leq -\ddot{x}_d + E + F(X) + G_M \left\|\bar{X} - X\right\| + G(X)(\tau_{c2}) + \tau_d \end{aligned} \tag{2.73}$$

Using the fact that

$$\hat{G}(X \mid \vartheta_g)\hat{G}^T(X \mid \vartheta_g)[\varepsilon_0 I + \hat{G}(X \mid \vartheta_g)\hat{G}^T(X \mid \vartheta_g)]^{-1} = I - \varepsilon_0[\varepsilon_0 I + \hat{G}(X \mid \vartheta_g)\hat{G}^T(X \mid \vartheta_g)]^{-1} \quad (2.74)$$

and

$$\hat{G}(X \mid \vartheta_g)\tau_{cf} = \ddot{x}_d - E - \hat{F}(X \mid \vartheta_f)) - \hat{h}(S \mid \vartheta_h) - K_1 S - \tau_0 \quad (2.75)$$

we denote the disturbance torque τ_d, denoted as $\hat{\tau}_d$ to imply boundedness, as being bounded $\|\hat{\tau}_d\| < \hat{\tau}_{dM}$ and define a constant $k = \hat{\tau}_{dM} + \Gamma$, with Γ as a small constant number, and Equation 2.73 can be written as

$$\begin{aligned}
\dot{S} &\leq -\ddot{x}_d + E + F(X) + (G(X) - \hat{G}(X \mid \vartheta_g))\tau_{cf} + \hat{G}(X \mid \vartheta_g)\tau_{cf} - G(X)\tau_{cr} + \hat{\tau}_d \\
&\leq -\hat{F}(X \mid \vartheta_f)) - \hat{h}(S \mid \vartheta_h) - K_1 S - \tau_0 + F(X) + (G(X) - \hat{G}(X \mid \vartheta_g))\tau_{cf} - G(X)\tau_{cr} \\
&+ \hat{\tau}_d \\
&\leq -K_1 S + F(X) - \hat{F}(X \mid \vartheta_f) + (G(X) - \hat{G}(X \mid \vartheta_g))\tau_{cf} - G(X)\tau_{cr} - \hat{h}(S \mid \vartheta_h) \\
&+ \hat{\tau}_d - \tau_0 \\
&\leq -K_1 S + \hat{F}(X \mid \vartheta_f^*) - \hat{F}(X \mid \vartheta_f) + \varpi_F + (\hat{G}(X \mid \vartheta_g^*) - \hat{G}(X \mid \vartheta_g) + \varpi_G)\tau_{cf} - G(X)\tau_{cr} \\
&- \hat{h}(S \mid \vartheta_h) + \hat{\tau}_d - \tau_0. \qquad (2.76)
\end{aligned}$$

Multiplying S^T to Equation 2.76 gives

$$\begin{aligned}
S^T\dot{S} &\leq -S^T K_1 S + S^T(\hat{F}(X \mid \vartheta_f^*) - \hat{F}(X \mid \vartheta_f)) + S^T\varpi_F + S^T(\hat{G}(X \mid \vartheta_g^*) - \hat{G}(X \mid \vartheta_g) + \varpi_G)\tau_{cf} \\
&- S^T G(X)\tau_{cr} + S^T\hat{h}(S \mid \vartheta_h^*) - S^T\hat{h}(S \mid \vartheta_h) - S^T\hat{h}(S \mid \vartheta_h^*) \\
&+ S^T\hat{\tau}_d - S^T\tau_0. \qquad (2.77)
\end{aligned}$$

Define Ψ_{f_i}, $\Psi_{g_{ij}}$ and Ψ_{h_i} to represent the fuzzy parameter errors such that $\Psi_{f_i} = \vartheta^*_{f_i} - \vartheta_{f_i}$, $\Psi_{g_{ij}} = \vartheta^*_{g_{ij}} - \vartheta_{g_{ij}}$, and $\Psi_{h_i} = \vartheta^*_{h_i} - \vartheta_{h_i}$. We can then rewrite Equation 2.77 as

$$\begin{aligned} S^T\dot{S} \quad \leq \quad & -S^T K_1 S + \sum_{i=1}^{\varpi} \Psi_{fi}^T \zeta_{fi}(S) S_i + S^T \varpi + \sum_{i=1}^{\varpi}\sum_{j=1}^{\varpi} \Psi_{g_{ij}}^T \zeta_{g_{ij}}(S) S_i \tau_{cfj} \\ & + \sum_{i=1}^{\varpi} \Psi_{h_i}^T \zeta_{h_i}(S_i) S_i + S^T \hat{\tau}_d - S^T \hat{h}(S \mid \vartheta^*_h) - S^T \tau_0 - S^T g(X) \tau_{cr} \end{aligned} \tag{2.78}$$

where $S^T\varpi = S^T\varpi_f + S^T\varpi_G\tau_{cf} + S^T\varpi_h$.

A Lyapunov function candidate is defined as

$$V = \frac{1}{2}\left[S^T S + \sum_{i=1}^{\varpi} \frac{1}{\alpha_i}\Psi_{f_i}^T\Psi_{f_i} + \sum_{i=1}^{\varpi}\sum_{j=1}^{\varpi} \frac{1}{\beta_{ij}}\Psi_{g_{ij}}^T\Psi_{g_{ij}} + \sum_{i=1}^{\varpi} \frac{1}{\varsigma_i}\Psi_{h_i}^T\Psi_{h_i} + \frac{1}{\kappa_0}\hat{\chi}^2\right]. \tag{2.79}$$

The time derivative of V is obtained as

$$\dot{V} = S^T\dot{S} + \sum_{i=1}^{\varpi} \frac{1}{\alpha_i}\Psi_{f_i}^T\dot{\Psi}_{f_i} + \sum_{i=1}^{\varpi}\sum_{j=1}^{\varpi} \frac{1}{\beta_{ij}}\Psi_{g_{ij}}^T\dot{\Psi}_{g_{ij}} + \sum_{i=1}^{\varpi} \frac{1}{\varsigma_i}\Psi_{h_i}^T\dot{\Psi}_{h_i} + \frac{1}{\kappa_0}\hat{\chi}\dot{\hat{\chi}}. \tag{2.80}$$

Substituting Equation 2.78 into Equation 2.80 and using the definition that $\hat{h}(S \mid \vartheta^*_h) = k\text{sgn}(S)$, we obtain

$$\begin{aligned} \dot{V} \quad \leq \quad & -S^T K_1 S + \sum_{i=1}^{\varpi} \frac{1}{\alpha_i}\Psi_{f_i}^T(\alpha_i\zeta_{f_i}(S)S_i + \dot{\Psi}_{f_i}) + S^T\varpi \\ & + \sum_{i=1}^{\varpi}\sum_{j=1}^{\varpi} \frac{1}{\beta_{ij}}\Psi_{g_{ij}}^T(\beta_{ij}\zeta_{g_{ij}}(S)S_i\tau_{cj} + \dot{\Psi}_{g_{ij}}) + S^T(\hat{\tau}_d - k\text{sgn}(S)) \\ & + \sum_{i=1}^{\varpi} \frac{1}{\varsigma_i}\Psi_{h_i}^T(\varsigma_i S_i \zeta_{h_i}(S_i) + \dot{\Psi}_{h_i}) - S^T(\tau_o) - S^T G(X)\tau_{cr} + \frac{1}{\kappa_0}\hat{\chi}\dot{\hat{\chi}} \end{aligned} \tag{2.81}$$

The time derivative of V can then be written as the following.

$$\dot{V} \leq \dot{V}_0 + \dot{V}_1 + \dot{V}_2 + \dot{V}_3 \tag{2.82}$$

$$\dot{V}_0 = -S^T K_1 S \leq -\lambda_{min}(K_1) \|S\|^2 \tag{2.83}$$

$$\dot{V}_1 = S^T(\hat{\tau}_d - k\text{sgn}(S)) \leq -\Gamma \|S\| \tag{2.84}$$

$$\begin{aligned}\dot{V}_2 = & \sum_{i=1}^{\varpi} \frac{1}{\alpha_i} \Psi_{f_i}^T (\alpha_i \zeta_{f_i}(S) S_i + \dot{\Psi}_{f_i}) + \sum_{i=1}^{\varpi} \sum_{j=1}^{\varpi} \frac{1}{\beta_{ij}} \Psi_{g_{ij}}^T (\beta_{ij} \zeta_{g_{ij}}(S) S_i \tau_{cj} + \dot{\Psi}_{g_{ij}}) \\ & + \sum_{i=1}^{\varpi} \frac{1}{\varsigma_i} \Psi_{h_i}^T (\varsigma_i S_i \zeta_{h_i}(S_i) + \dot{\Psi}_{h_i})\end{aligned} \tag{2.85}$$

$$\dot{V}_3 = S^T \varpi - S^T \tau_o - S^T G(X) \tau_{cr} + \frac{1}{\kappa_0} \hat{\chi} \dot{\hat{\chi}} \tag{2.86}$$

We now rewrite $\dot{\Psi}_{f_i} = \dot{\vartheta}^*_{f_i} - \dot{\vartheta}_{f_i}$, $\dot{\Psi}_{g_{ij}} = \dot{\vartheta}^*_{g_{ij}} - \dot{\vartheta}_{g_{ij}}$, and $\dot{\Psi}_{h_i} = \dot{\vartheta}^*_{h_i} - \dot{\vartheta}_{h_i}$. Since $\vartheta^*_{f_i}$, $\vartheta^*_{g_{ij}}$ and $\vartheta^*_{h_i}$ are constant numbers, $\dot{\Psi}_{fi} = -\dot{\vartheta}_{fi}$, $\dot{\Psi}_{g_{ij}} = -\dot{\vartheta}_{g_{ij}}$, and $\dot{\Psi}_{hi} = -\dot{\vartheta}_{hi}$.

Using Equation 2.70 and rewriting Equation 2.85 we can obtain

$$\dot{V}_2 = 0. \tag{2.87}$$

Using Equation 2.69, the following equation can be written.

$$S^T G(X) \tau_{cr} \geq \|S\| (\bar{\varpi}_F + \bar{\varpi}_G \left\|\tau_{cf}\right\| + \bar{\varpi}_h + \|\tau_0\|) - \frac{\hat{\chi} \|S\| (\bar{\varpi}_F + \bar{\varpi}_G \left\|\tau_{cf}\right\| + \bar{\varpi}_h + \|\tau_0\|)}{g_0 \|S\|^2 + \hat{\chi}} \tag{2.88}$$

and we make use of the following inequality

$$S^T G(X) S \geq g_0 \|S\|^2 \tag{2.89}$$

Combining the adaptive law Equation 2.71 with Equation 2.86, we have

$$\dot{V}_3 \leq -S^T G(X)\tau_{cr} + \|S\|\left(\bar{\varpi}_F + \bar{\varpi}_G \left\|\tau_{cf}\right\| + \bar{\varpi}_h + \|\tau_0\|\right) + \frac{1}{\kappa_0}\hat{\chi}\dot{\hat{\chi}} = 0. \tag{2.90}$$

From the above Equations 2.82, 2.83 and 2.84,

$$\dot{V} \leq \dot{V}_0 + \dot{V}_1 \leq -\lambda_{min}(K_1)\|S\|^2 - \|S\|(\Gamma). \tag{2.91}$$

Based on Equations 2.87, 2.90 and 2.91, Equation 2.82 can be written as

$$\dot{V} \leq -\lambda_{min}(K_1)\|S\|^2 + \|S\|(g_1 d_1 - \Gamma) \leq -\lambda_{min}(K_1)\|S\|^2 \tag{2.92}$$

and $\lambda_{min}(K_1)$ is the minimum eigenvalue of matrix K_1. In this way, the control input and all signals involved in the control system are bounded. This provides a stable, nonlinear and model-free control system for managing μrover dynamics.

2.4 Electronics Stack

2.4.1 Component Selection

The selection of components for surface-mount hardware fabrication is important. To allow hand soldering, testing, and debugging which is sometimes necessary for critical prototypes, no components should be smaller than imperial 0603 size (metric 1608). To cope with the temperatures encountered in space and hostile environments, all integrated circuits (ICs) must be tolerant to at least the industrial temperature range of $-40°C$ to $85°C$. Military and aerospace grade parts are of course preferred, but are rare to find as COTS hardware and very expensive. To keep precision over wide temperature range changes, resistors and other linear devices should be 0.5% tolerance or better. Wattage allowances of onboard components are also higher to cope with heating in vacuum, as the lack of air cooling typically causes components to run $20°C$ or more above their normal atmospheric temperatures. Also, due to the presence of ESD from many sources, all external connectors use ESD protected drivers and fast zener diodes to limit voltage spikes. For inexpensive and reliable fabrication, PCBs should have traces of at least $0.2mm$ width and at least $0.17mm$ mil clearance between traces or vias, with vias no smaller than $0.5mm$. Standard hole sizes for through-hole components are $0.71mm$ (fine leads), $0.9mm$ (standard leads), $1.07mm$ (power devices and DIP sockets), $1.52mm$ (large connectors), and $2.18mm$ (mounting holes). Typically $1/8$“ ($3.175mm$) holes are used for board mounting and stacking.

2.4.2 On-Board Computer

The On-Board Computer (OBC) for the μrover is based on the Atmel AT91RM9200 ARM9 microcontroller, which is a low-power, multipurpose, embedded controller IC. It has an ARM920T 32-bit CPU core and a large variety of on-chip communications controllers, including a full 32-bit GPIO port, four enhanced USART serial controllers, SPI, I^2C, SDIO, and Ethernet support. μrover prototypes built from scratch use a Linuxstamp v1.2.0 embedded board, which provides an SPI dataflash, 64MB RAM, connectors for Ethernet and USB, an SD card, and through-hole headers for connection to all the functions of the AT91RM9200, and have been tested thoroughly

both in the lab and in the field so that the electronic design can be refined. An early diagram of the basic functions of the μrover is shown in Figure 2.15. While there are many potential ARM architecture microcontrollers that could be used in an OBC design, the AT91RM9200 was chosen because of its variety of peripherals, high levels of hardware maturity and support in the Linux kernel, and because of the availability of the Linuxstamp as an open platform to build on. Other programmable component functions on the μrover such as motor control, sensor monitoring, and simple payloads are handled by smaller microcontrollers that are attached by synchronous serial or SPI bus to the OBC. The Atmel AVR 8-bit architecture microcontrollers have been used almost exclusively for these functions so far because of their simple and robust programmability, low power consumption, and high levels of support within the open-source community. AVR Microcontrollers are used for the popular Arduino series of hobbyist microcontroller boards, which makes them readily available and ideal for academic use. Though many newer 32-bit microcontrollers such as the ARM-Cortex series have superior performance and similar power and size requirements, the AVR is greatly simpler to program at low levels, and due to this ease of implementation, many research prototypes of electronic hardware developed for or with the μrover have been based on the well-designed and flexible ATMega644P microcontroller.

The prototype hardware for the μrovers has been built mostly by hand using through-hole components, which makes the motherboards, daughterboards and payloads larger and heavier. A surface-mount electronic implementation in an extended version of the PC/104 form factor is currently undergoing testing. The PC/104 form factor describes a stack of PCBs with pass-through $0.1 inch$ 64-pin and 40-pin headers, and was chosen because it is a common standard for embedded systems in industry and robotics, and strikes a good balance between compactness and flexibility. The OBC base board is enlarged to allow a two-layer board to carry additional on-board computer functions and power conversion components, and the pinout is altered to contain the communications interfaces described here. The electronics stack contains all the essential functions and instrumentation for the μrover, such as motor drivers, battery chargers, DC-DC converters, current and voltage monitoring, accelerometer, magnetometer, and rate gyro sensors, and sockets for cameras, GPS, and communications. Expansion of this board is possible by using

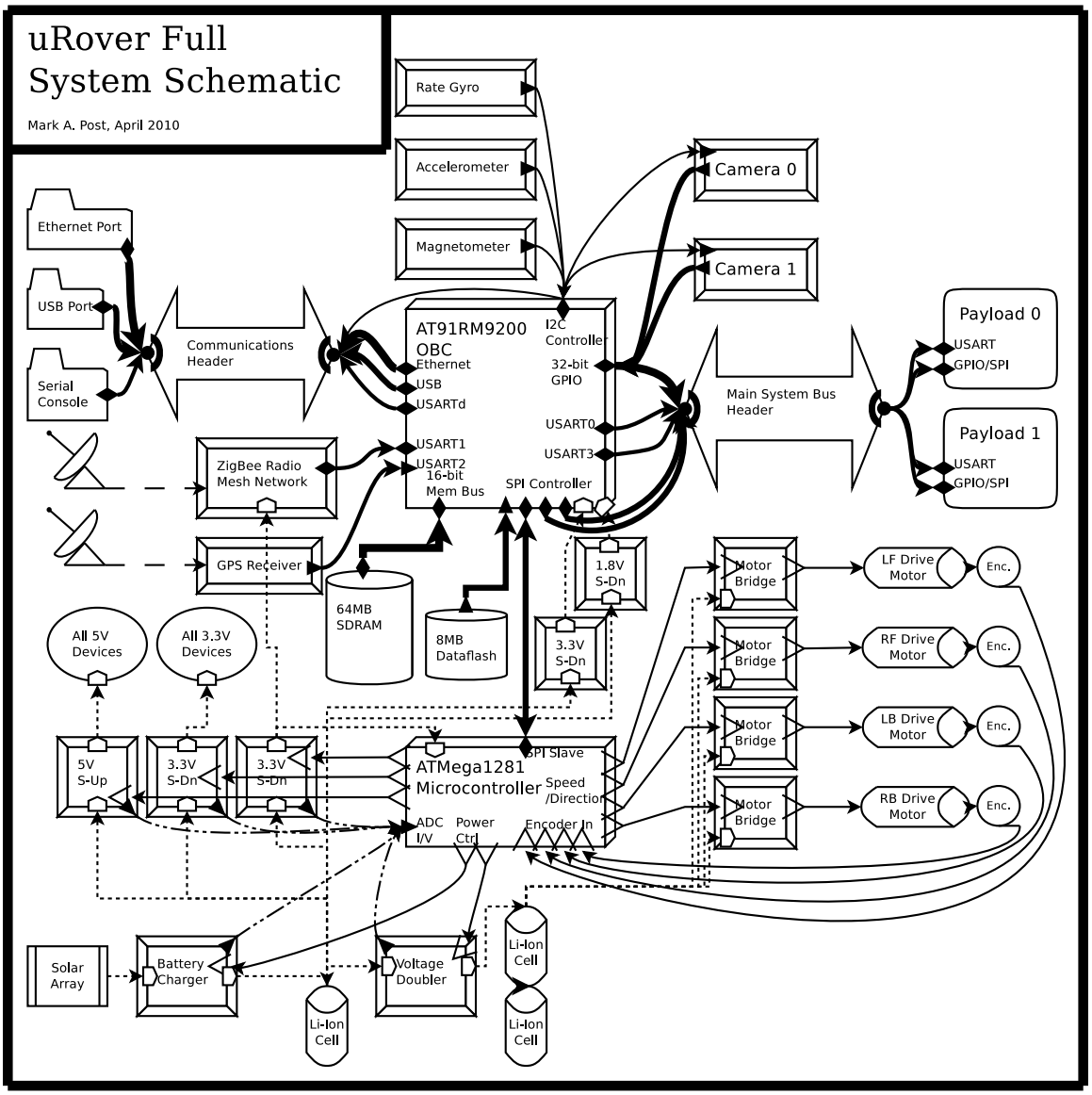


Figure 2.15: Initial hardware diagram for μrover

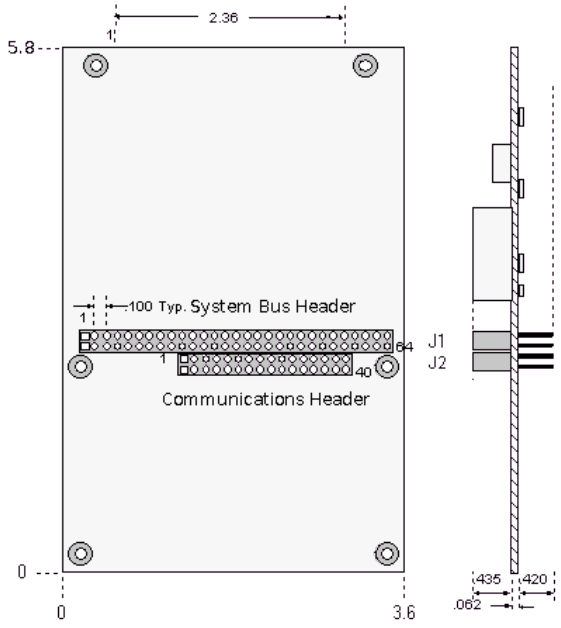


Figure 2.16: μrover Main Board Dimensions

PC/104-style 64-pin and 40-pin headers. The PC/104+ form factor is used for board size and placement of the connectors, but it will be updated to include motor drivers and debugging ports. The form factor and layout of this board is shown in Figure 2.16. A parallel interface connection to GPIO pins is used to connect an OV7670 VGA CMOS camera module directly to the OBC, which is driven from the system clock on the AT91RM9200. To aid greatly in development and troubleshooting, a standard set of wire colours will be used for chassis wiring of prototypes, shown in Figure 2.17.

2.4.3 Board-to-Board Communications

All signals are carried between OBC boards as shown in Figure 1.11 via standard 0.1*inch* pin headers. This facilitates connection between boards and wires, and makes debugging easy. Proprietary connectors are often hard to test and usually are not compatible between manufacturers and sizes. Due to the resiliency of using separate transmit, receive, clock, and frame synchronization wires, SPI is preferred for board-to-board communication at high bit rates, though a synchronous serial interface with a clock line for synchronization also provides a reliable connection.

Figure 2.17: Standard Wire Colours Used for Prototyping

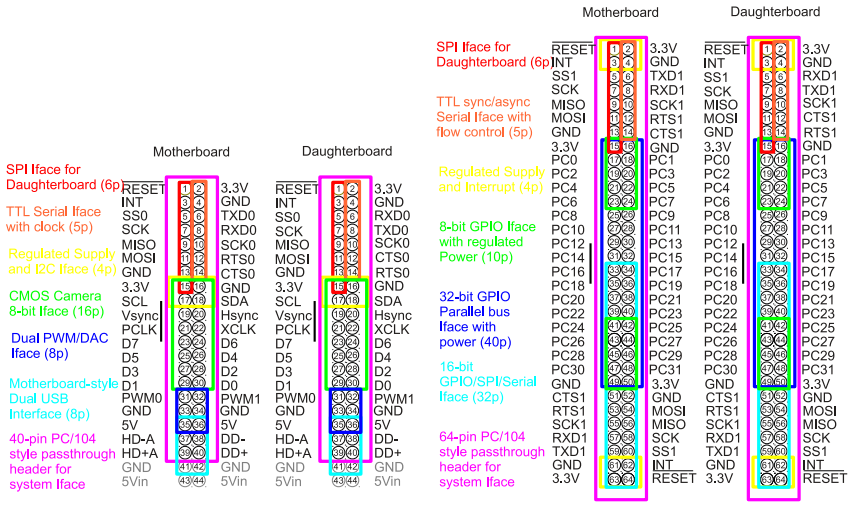


Figure 2.18: μrover Main and Auxiliary Board Headers

Communication with payloads is generally achieved using 8-bit serial communications, which is the most common low-level way to interface with embedded systems, but TTL serial interfaces are also broken out through the pin headers for use in used in board-to-board communications. The pinouts for the pass-through headers on the boards are shown in 2.18. For debugging and system recovery purposes, one serial port is designated as a system console or debug port, and this is usually available via USB-to-Serial converter IC for convenience in connecting portable PCs. GPS Devices and embedded radios are often serial devices that can be included in the OBC stack. Though very common, high speed personal computer buses such as PCI and PCI-Express do not present significant benefit to low-speed embedded systems with limited I/O resources and integrated controllers.

The number of Serial and SPI devices is only limited by the number of SPI and Serial interfaces on the ARM microcontroller. Provision is made on the board headers for at least four serial interfaces and four SPI interfaces with chip selects. Board headers do not make use of line drivers to save power. There have been no problems using direct pin connections between boards so long as the housing for the OBC stack is sealed from sources of ESD and contaminants. All onboard systems can be managed through a mixture of SPI and synchronous TTL serial interfaces. For register-level access to onboard sensors and peripherals, I^2C is a synchronous bus widely used for

register-level communications to multiple embedded ICs, but because it must embed addressing, bus arbitration, and multi-master capability using only two wires and a simple protocol, it is very sensitive to interference and noise from the environment. To maximize reliability, trace lengths to the I^2C host are minimized and daughterboards and payloads must use a microcontroller to transfer data to SPI or serial bus lines from the I^2C device. For interfacing high-bandwidth devices, CMOS and CCD cameras are interfaced directly to the central ARM microcontroller via either a parallel bus made up of general-purpose input-output (GPIO) pins, or if available, a dedicated image sensor interface (ISI) on the microcontroller. Both volatile (RAM) and non-volatile (Flash) memory is interfaced by means of dedicated memory hardware, i.e. RAM interface, NAND Flash interface, and SDIO bus. Nearly all modern ARM microcontrollers have dedicated memory interfaces. For compatibility between versions and devices, a set of standards for pin interfacing of the various types of buses is essential. The standard pin headers for the μrover and associated hardware are shown in Figure 2.19, and were derived from conventions used in many types of commercially-available embedded hardware. The board header pinouts shown in Figure 2.18 were specifically chosen to use these conventions for ordering pins so that wired devices can connect directly to the board headers without changing form, which is extremely useful for development.

Standard programming headers are also important for compatibility between devices. Figure 2.20 shows the programming headers in use on the μrover.

2.4.4 Embedded and Payload Communications

For external payload communications, RS-485 has been selected as the standard of choice. The RS-485 interface is an industry standard for differential-pair serial communications to multiple transceivers, and is already used on many robotic systems. It may be used in either half-duplex mode (separate transmit and receive pairs) or full-duplex mode (pairs connected), but full-duplex with a single master is preferred to avoid byte collisions and the need for bus arbitration. RS-422 devices are also compatible with this interface.

Additionally, the RS-485 interface is designed for bidirectional communication with up to

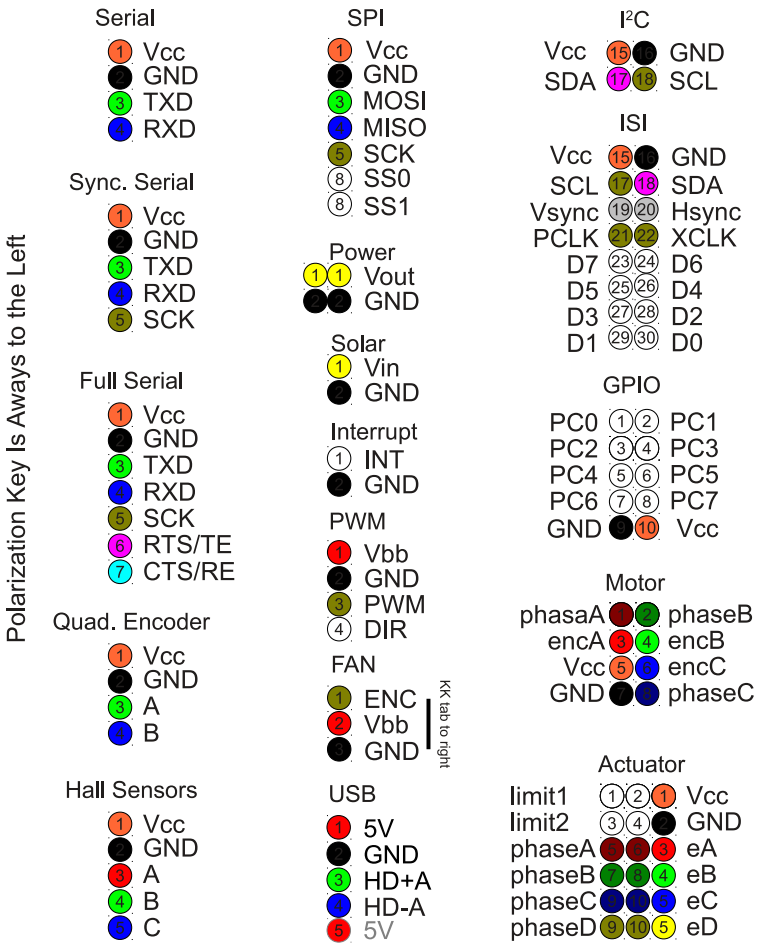


Figure 2.19: μrover Miscellaneous Headers

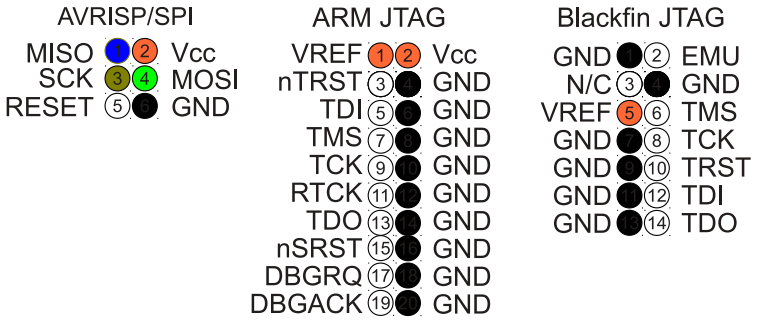


Figure 2.20: μrover Programming Headers

32 devices on a single bus, the number of potential payloads is high enough to accommodate large robotic systems. While RS-485 can operate at 10 megabits per second at 40 feet and 100 kilobits per second at 4000 feet, to allow low-speed microcontrollers to communicate reliably, the standard baud rate of 115200 baud is used as the default speed unless higher bandwidth for the specific payload is required. For long-distance disconnectable operation, the use of an RJ-45 jack and twisted-pair cabling also provides good performance. For extreme environments, an additional pair is used to transmit a clock signal for synchronous operation to ensure reliable transmission. With the exception of high-bandwidth devices such as optical sensors, memory, and media interfaces, these serial links are more than capable of real-time command transmission. As no standardized D-sub pinout has been accepted for RS-485 communications, one was created to suit the modular system with differential clock (CK) power supply, and interrupt pins. The external interfaces are situated on a board in the OBC stack with line drivers and external DE9 connectors. To pass GPIO signals and parallel buses, DB25 connectors are useful with each pin buffered against ESD and high voltage also. The pinouts used for payload interfacing over D-sub connectors are shown in Figure 2.21.

The very common RS-232 serial port, while still present in a huge number of devices, is not traditionally compatible with RS-485 interfaces and is less efficient as it is based on negative logic and $\pm 5 - 15V$ signal levels. A dedicated converter is the best method for interconnection. However, given that most modern hardware uses only $\pm 5 - 7V$ and has thresholds at $1V$, it is possible for RS-232 and RS-485 devices to communicate directly if required, for example, in new hardware testing or field debugging using RS-232. The payload pinout was chosen to be electrically compatible with RS-232 for this reason and to increase safety in case of accidental RS-232 connection. The RS-485 TX-(Z) and RX-(B) pins are connected to RS-232 RX and TX respectively to invert logic levels, TX+(Y) and RX+(A) are connected to RS-232 CTS and RTS respectively to serve as a ground reference. The CTS and RTS pins must be allowed to float to ground, and if the hardware does not allow this, TX+(Y) and RX+(A) must be connected to the RS-232 ground instead. This method works well using a MAX489 transceiver IC and is considered an option for debugging and testing with RS-232 hardware and purpose-built serial

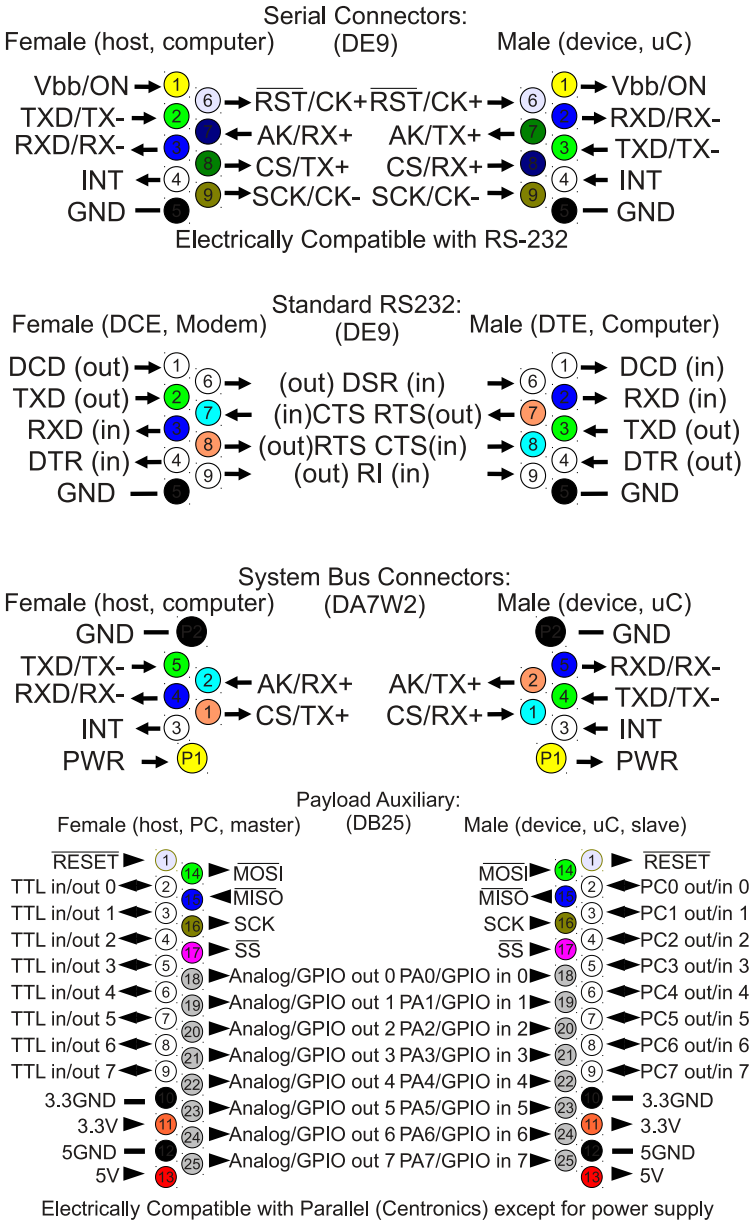


Figure 2.21: μrover Payload Serial and Parallel Pinouts

Pin	RS232	RS485	10bT	1KbT	EIA/TIA 568A	EIA/TIA 568B
1	Vcc	Vcc	TX+	BI_DA+	Wh/Green	Wh/Orange
2	Vcc	Vcc	TX-	BI_DA-	Green	Orange
3	GND	GND	RX+	BI_DB+	Wh/Orange	Wh/Green
4	RTS	TX+/Bus+	PoE+	BI_DC+	Blue	Blue
5	RX	TX-/Bus-	PoE+	BI_DC-	Wh/Blue	Wh/Blue
6	GND	GND	RX-	BI_DB-	Orange	Green
7	CTS	RX+	PoE-	BI_DD+	Wh/Brown	Wh/Brown
8	TX	Rx-	PoE-	BI_DD-	Brown	Brown

Figure 2.22: μrover Modular Jack Pinout

interfaces. But it is less efficient than only RS-485 communications and breaks the RS-232 specification by referencing signal levels to ground rather than $\pm 5-15V$. Newer RS-232 hardware and USB to RS-232 converters that have low signal voltages are useable, but older high-voltage RS-232 hardware may be unsafe to use as the RS-485 transceiver used must be able to tolerate the signal voltages. RS-485 is also often used over Unshielded Twisted Pair (UTP) cable using 8-pin RS-485 modular plugs. A compatibility table for the use of UTP cable is shown in Figure 2.22

For devices other than system payloads, USB can be used for COTS hardware, since the interface is present on many embedded devices, but is generally restricted to hot-pluggable external hardware and mass storage devices rather than hard-wired onboard peripherals. Ethernet can also be used, but has significant programming and routing overhead, so it is preferred as an external network interface for development purposes. Other industrial standards for communications such as CAN, LIN, Firewire and Spacewire, though often used in other systems for reliability-critical applications [Torre *et al.* 2010], are not present on the majority of embedded devices and are not generally used in the interest of easy interoperability. The Spacewire bus in particular, designed and used for space hardware, is very fast and robust, but is only implemented fully in hardware on a very small set of radiation-hardened devices which are difficult to source and expensive, though the LVDS standards it is based on are well-known and widely available. Consequently, as standard external interfaces to the μrover, we implement clocked RS-485 serial, buffered SPI, and clocked parallel buses for payloads and include provision for a USB port and an Ethernet RJ-45

jack that can be omitted for flight hardware to save power as needed. No provision is made for CAN, LIN, Firewire and Spacewire buses as significant extra hardware would be required, though LVDS has the potential to be a low-power replacement for RS-485 serial and may facilitate the construction of spacewire interfaces in the future.

2.4.5 Radio Communications

Wireless communications between mobile robots and control stations is a critical component in applications requiring remote control or monitoring. To simplify terrestrial work, it is desirable to use license-free ISM bands or amateur radio bands when communicating. The frequency must also be high enough to support high bit rate communications, preferably at least the basic system rate of 115200 baud. The $2.4GHz$ band is by far the most popular, but higher frequency systems have lower range and worse non-line-of-sight (NLOS) characteristics, and this band has also become very crowded with interference from consumer electronic devices. Another option is the $433MHz$ band and below, which can be used by amateur radio operators, but limits data rates on most radio hardware to 38400 baud and lower. The $900MHz$ ISM band is a good medium between these options, as it provides reasonable range and NLOS characteristics, can support bit rates above 115200 baud, and is not used excessively by consumer devices.

Another consideration is the need for multi-point communications and routing. There are many situations when several mobile robots need to stay in communications with each other and with base station units, but relative movement, terrain variations and ambient conditions make static network structures unreliable. One popular solution to this is the mesh network, which allows multiple transceivers to route data to each other using dynamically maintained and self healing network structures. The best known industrial mesh network specification is ZigBee, which is used frequently in sensor mesh networks but requires a dedicated coordinator and router nodes which limits the flexibility of the network. The B.A.T.M.A.N. (better approach to mobile ad-hoc networking) routing protocol was recently added to the Linux kernel to allow mesh networking over wireless LAN networks, but is not designed for low-level serial hardware. To provide a long-range serial mesh networking system, the Digi XBee PRO Digimesh 900 serial ra-

dio module is currently used. The XBee modules were chosen for communications because they were found to be the only readily available commercial radio module that combines transparent mesh networking, low power use, a small form factor, and a well-known command set. The XBee PRO modules were also used in several related projects for wireless serial connectivity and proved to be very dependable. The Digimesh protocol is proprietary, but allows mesh networking without a central coordinator. As these radio modules use a well-known form factor, replacing them with $2.4GHz$ modules or custom modules that change frequency, range, and protocol is simple.

2.4.6 Battery Subsystem

Onboard batteries are expected to be used for power storage, and can be charged from solar panels or other sources as needed. Battery voltage and current monitoring is done via analog-to-digital converters, which are present on most modern microcontrollers. As the batteries of the μrover are charged from solar panels, which can provide more than the required $4.2V$ to charge a single Li-Ion cell but are also inherently current-limited, it is considered practical to increase efficiency by dispensing with the commonly used step-up or step-down converter topologies for battery charging, and instead use a simple PWM-based pulse charger, where PWM duty cycle is used to directly control the current flow into a battery cell. To reduce the size and improve performance of the $3.7V$ charging system, a MAX1736 pulse-charger IC is used as shown in Figure 2.23 to control the charging process. Lithium-Ion (Li-Ion) cells have the advantage that they can be simply charged from a current and voltage-limited source with less complexity than Nickel-Cadmium (NiCd) or Nickel Metal Hydride (NiMH) cells, but do not self-balance like NiCd or NiMH cells, and consequently each cell must be actively balanced by the charging system. Testing of the Li-Ion cells [Navarathinam *et al.* 2011] and high-current DC-DC power converters has been done to verify performance in extreme environmental and operational conditions.

It is therefore necessary to step up the voltage provided by the solar panels in as efficient a manner as possible using the cell at the bottom of the stack as a buffer, leading to two potential design solutions. The basic layout of these two approaches is shown in Figure 2.24. The

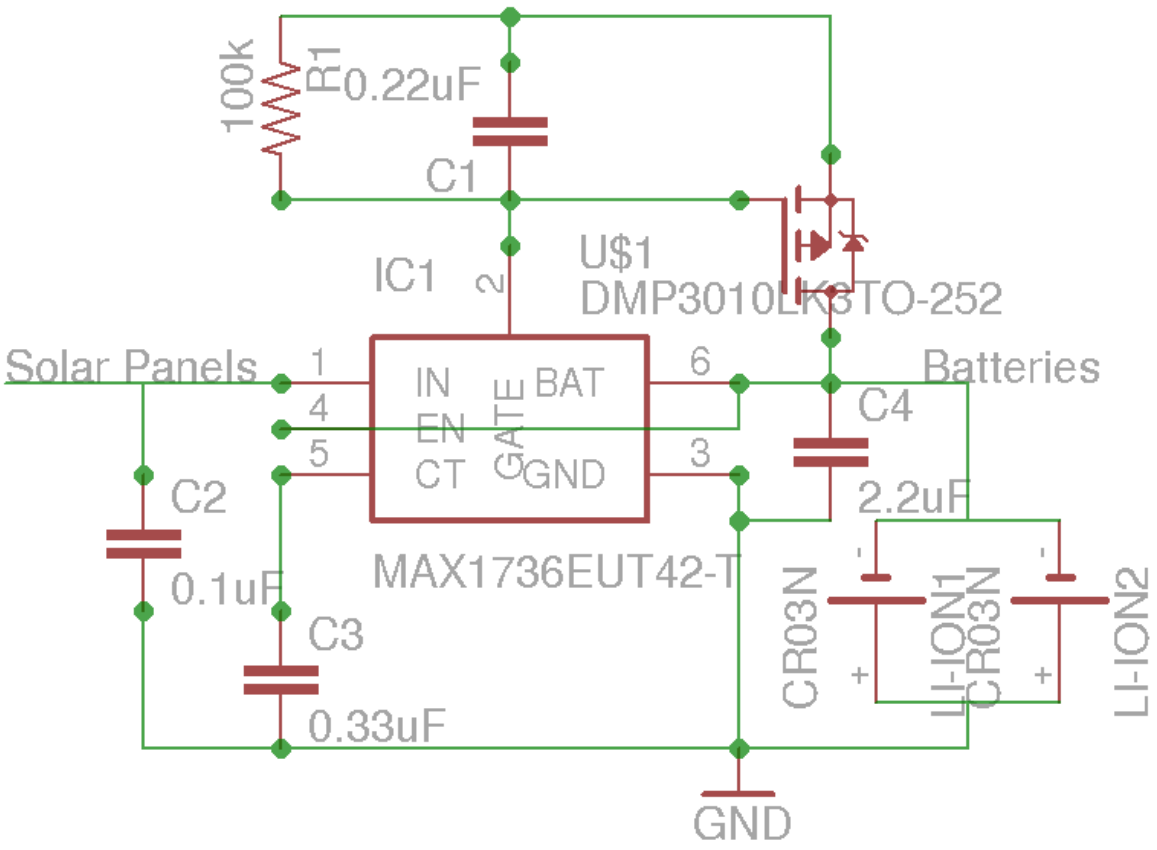
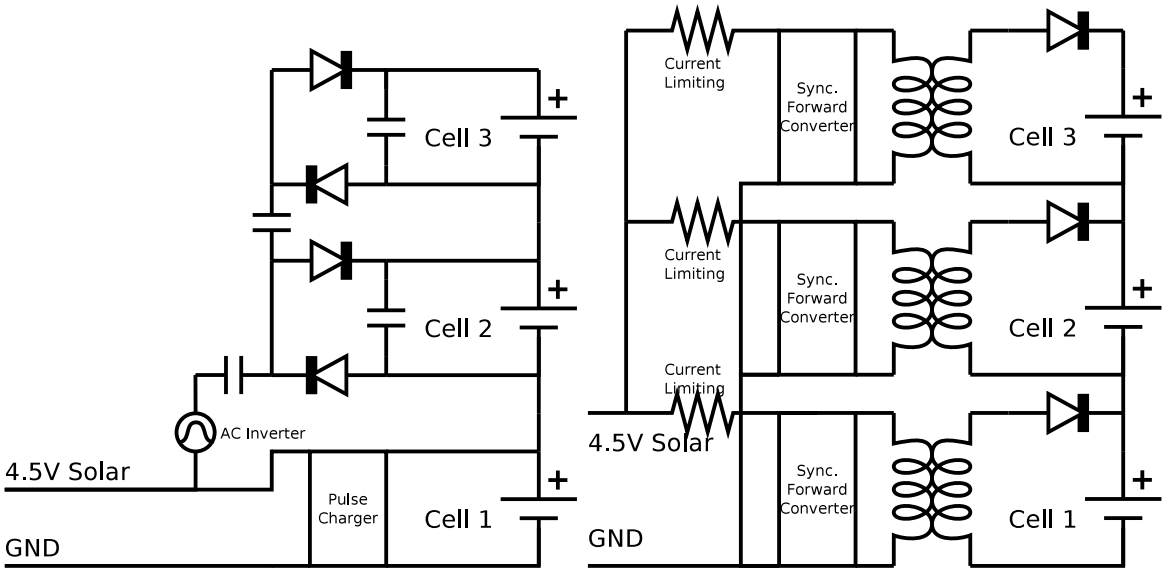


Figure 2.23: μrover Li-Ion pulse charger circuit using MAX1736

first (a) is the “charge ladder” approach, where voltage-multiplying charge pumps are used to charge each successive battery in series from the previous one. The lowest battery in the stack is charged through conventional pulse charging. This has the advantage of very high efficiency and reliability, but suffers from long charge times due to the very limited current capacity of voltage multipliers. The second is the “isolated bus” approach (b), where a single $4.2V$ bus feeds all battery cells in parallel via synchronous forward converters and transformers, which implicitly step the voltage to the level of each cell [Altemose 2008]. This parallelization of cells makes voltage regulation simple and allows charging at higher rates, but the use of transformers decreases charging efficiency and adds weight and volume to the charging system, and an efficient forward converter with isolated voltage regulation must be used.

As fast charging of batteries is not typically required for planetary rovers, reliability and efficiency can be prioritized, and the ”charge ladder approach“ was ultimately chosen due mainly to the compactness of integration, and the efficiency that could be achieved. Figure 2.25 shows the battery charger configuration on the μrover with controller and Li-Ion cells. As $3.7V$ (one cell) power is needed for almost all the electronics, but $11.1V$ (three cells) is needed for the drive motors and payloads, the total number of cells is minimized by combining the stack. Two cells are used in parallel at the bottom to provide $3.7V$, and in series with that, two higher cells are used to provide $11.1V$. Zener diodes are used to ensure that overcharging never occurs, and to allow excess charge to spill into lower cells. While combining odd sets of cells such as this is not advised in most applications, no problems have been encountered in the μrover configuration. As each parallel set of cells is charged to its own voltage separately (known as ”balance charging“), this does not cause problems with the imbalance of voltages between cells. However, care must be taken that the bottom cells do not become discharged much faster than the higher cells due to high loading of the $3.7V$ supply bus. As the low-voltage electronic components typically consume much less power than the drive and payload systems, and also because the $3.7V$ cells can be pulse-charged from the solar array the fastest without the use of charge pumps, this problem has not been encountered in normal operation.



(a) Charge ladder approach (b) Isolated bus approach

Figure 2.24: Battery stack charge topologies

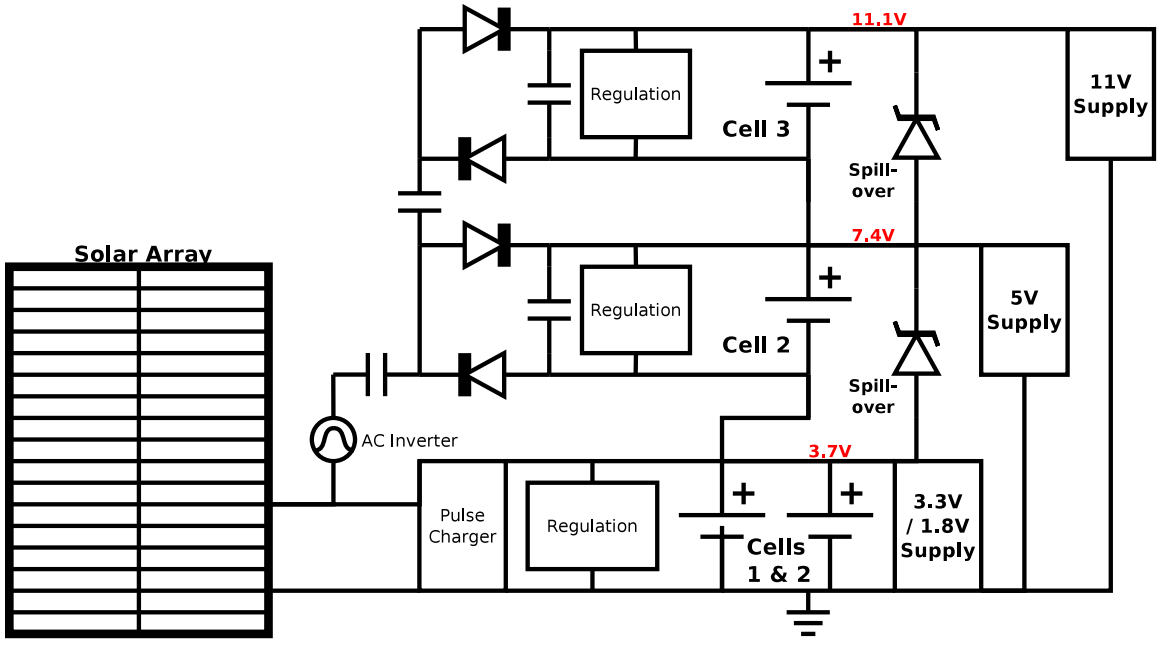


Figure 2.25: μrover battery charger topology and cell configuration

2.4.7 Power Distribution and Conversion

With the availability of low-cost highly-integrated switch-mode converters, it is feasible for each board in the electronics stack and payloads to convert DC power from the system battery to whatever level is needed. Daughterboards in the OBC generally use the system battery for power as well as regulated 3.3V and 5V supply rails provided by the motherboard. The choice of power supply is important. Simple linear voltage regulators that are commonly used on microcontroller boards limit output voltage by increasing the resistance through a transistor current mirror to lower output current. As a result, current input I_{in} always equals current output I_{out}, which means that the amount of power lost as heat in a linear regulator is proportional to the voltage ratio V_{out}/V_{in}. Hence linear regulators become less efficient relative to the input/output voltage difference, and while they can be very efficient for small voltage drops, they are generally unsuitable for use in vacuum or high-temperature environments as the only mechanism for cooling is direct radiation of heat.

There are many different methods for switched DC voltage conversion and current control, but the most relevant to the μrover are those that are efficient, require a minimal number of components, and are highly integrated in low-power ICs for commercial electronics. The most common configuration for simple voltage step-down or "buck" converters is the first-quadrant (or A-type) chopper, which uses pulse-width modulation through a MOSFET transistor to vary the total amount of energy allowed in to a lower-voltage circuit, and smooths voltage ripples with an LC filter on the output. The MAX1626 circuit in Figure 2.26 is a good example of this configuration. The limitation of this design is that efficiency is lower at low duty cycles due to very little current allowed through the MOSFET. Nonetheless, because no current is wasted, efficiencies between 85% and 95% are possible if the switching MOSFET is turned quickly and completely on and off between cycles because the MOSFET is the greatest source of resistance in such a circuit. Assuming all ideal components and constant load, output voltage is calculated as a function of input voltage V_{in} and the duty cycle t_{on}/T by [Fang Lin Luo 2003]

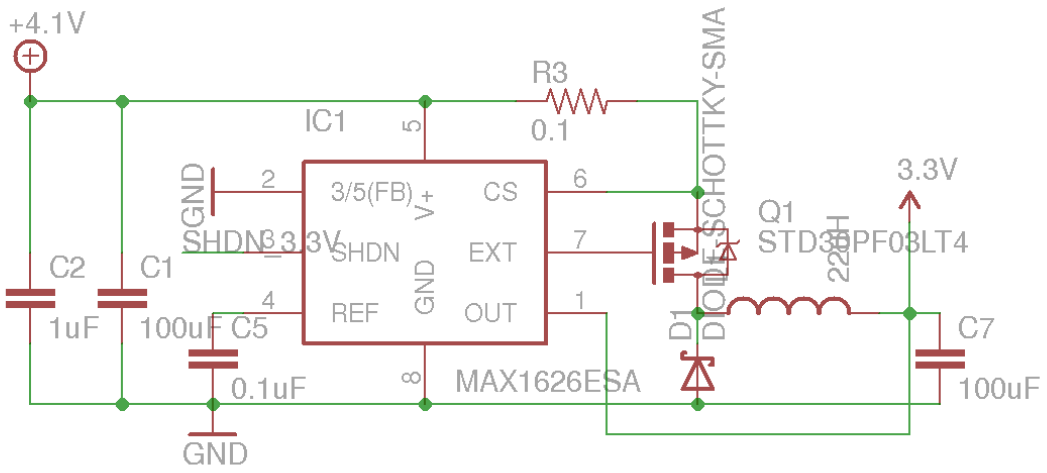


Figure 2.26: μrover buck converter circuit using MAX1626

$$V_{out} = \frac{t_{on}}{T} V_{in}. \tag{2.93}$$

Simple step-up, or "Boost" converters typically use the boost pump configuration which is derived from the second-quadrant chopper, where a MOSFET transistor is used to sink bursts of current through a large inductor, which continues to "pump" current through a diode a to a higher voltage after the MOSFET has been switched off, again typically using PWM. A typical example of a boost converter circuit is the MAX608 circuit in Figure 2.27. Again assuming all ideal components and constant load, output voltage is calculated as

$$V_{out} = \frac{T}{T - t_{on}} V_{in}. \tag{2.94}$$

The greatest drawback of this method is that when the MOSFET is switched on, a short path to ground is effectively created after the inductor saturates, which means that high duty cycles are relatively inefficient due to extra current being wasted through the inductor, and also means that failure of the controller will often cause overheating and catastrophic failure of the power supply. High efficiency in boost converter circuits is harder to obtain due to this configuration, and typically is between 70% and 85% using simple commercial converters. Of course, in all

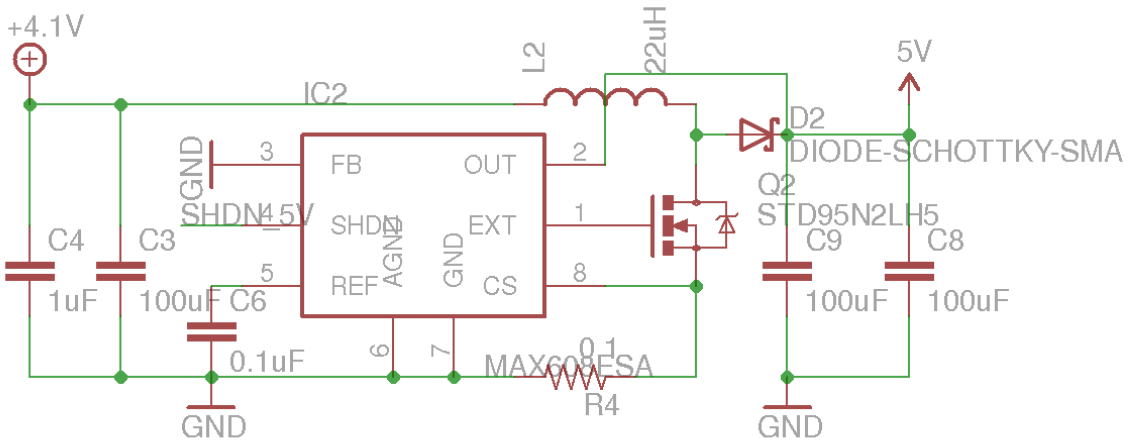


Figure 2.27: μrover boost converter circuit using MAX608

these cases, feedback control is necessary because the load impedance will change with power consumption. This feedback is provided through a resistor divider fed from the output of the converter and controls the PWM duty cycle.

Other types of voltage converter configurations are in use as well. The Cúk and Single-Ended Primary-Inductance Converter (SEPIC) topologies are multi-stage converters are based on the boost converter, but can provide both voltage step-up and step-down at the output [Fang Lin Luo 2003]. A larger number of components are required compared to the simple boost and buck configurations, but the availability of integrated power components such as the LT1513 and surface-mount inductors and capacitors helps to offset this. The biggest advantage is that the input voltage can vary greatly with respect to the output which makes these converter configurations attractive for battery charging and variable voltage supply applications. However, these converter configurations still require a boost circuit with current sinking through a MOSFET to ground, and suffer from the associated low efficiencies and potential for catastrophic failure. In testing of the LT1513, efficiency remained below 75% regardless of voltage conditions, so the SEPIC and Cúk topologies were removed from consideration for space hardware.

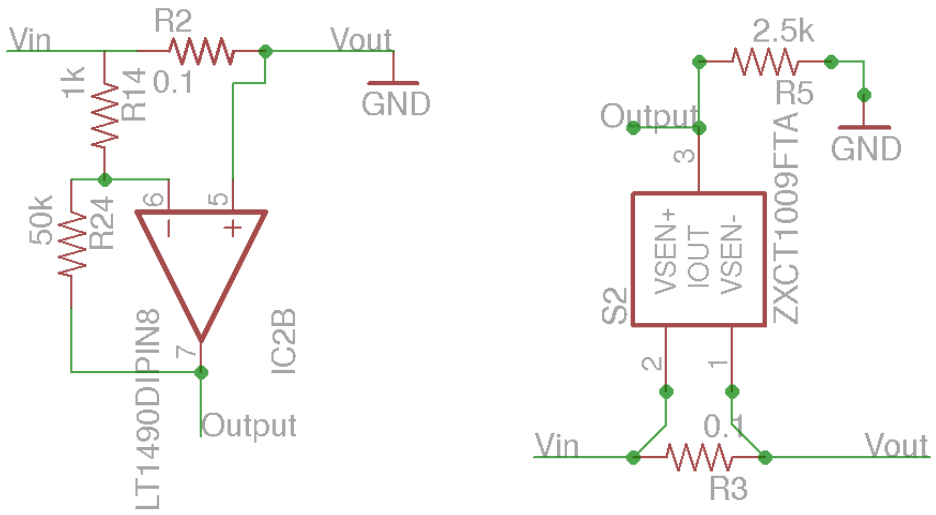
A wide variety of battery management and power conversion ICs are available commercially, so to determine the most appropriate parts for use on the μrover, a survey of components was

carried out. To achieve the highest power conversion efficiency possible, it was decided to focus on $3.3V$ components that would only require a single Li-Ion cell for power. As the payloads and drive system will require higher voltages, a stack of cells can be used for providing $7.4V$ and $11.1V$ directly rather than losing at least an estimated 20% efficiency by having to double or triple the voltage using high-current step-up converters. This also meant that switching step-down converters could be used to power $5V$ components from $7.4V$. By using only low-dropout buck converters with high input voltage tolerance, system efficiencies close to 90% can be achieved with relatively little additional complexity, save that of balanced battery charging. Also, each module is responsible for filtering its own load noise via capacitor-inductor networks to avoid injecting power supply transients back into the OBC. Zener diodes are also used to protect from ESD and over-voltages.

2.4.8 Sensors

The use of a large suite of sensors is important for intelligent robots to obtain the information they need for reliable control. Even on a small mobile robot, system health monitoring is important for intelligent operation. The solar cell current, 3.3V power draw, 5V power draw, and motor power draw are monitored by current sensing circuits like those shown in Figure 2.28. Although low-side (using a current sense resistor to ground) current monitoring is possible using a simple operational amplifier (op-amp) with a current sense resistor between the terminals as shown in (a), it is preferable to monitor current draw at the high-side (with current sense resistor connected to the voltage supply) as shown in (b) so that the voltage across the current sense resistor does not affect the ground connection of the load. The ZXCT1009 current sense amplifier IC can measure current at the high-side in this way and takes a minimum of board space by using a SOT-23 package. Temperature sensing on each board is done using a TMP102 I^2C temperature sensor, which measures board temperature over a range of $-40°C$ to $125°C$ and is both common and reliable.

The inertial sensors used for measuring acceleration, rotation rates, and magnetic field were selected primarily based on the magnitudes of quantities to be measured. Accelerations, rota-



(a) Op-Amp current sensing

(b) ZXCT1009 current sensing

Figure 2.28: μrover current sensing circuits

tional rates, and magnetic fields are all expected to be very low compared to Earth norms, and the most sensitive and modern devices that were both readily available and within budget were selected. These sensors all use I^2C communications, which has become the norm for inexpensive MEMS devices, and the circuits designed for the μrover are based on the Sparkfun 9DOF sensor stick, which integrates all three sensors into a compact package, and was used on the prototype μrover for testing. The Analog Devices ADXL345 accelerometer set to the $\pm 2g$ range is used for gravity vector sensing, the InvenSense ITG-3200 three-axis MEMS Gyroscope with a maximum range of $2000^\circ/s$ measures angular rates, and the Honeywell HMC5883L three-axis magnetometer with a minimum resolution of $73nT$ is used for magnetic field sensing. Again, the I^2C bus may not operate reliably in harsh environments with long electrical path lengths, all I^2C sensors are placed within $2.5cm$ of the host microcontroller they are attached to. Figure 2.29 shows the connections and external components for these sensors.

For external imaging, an OV7670 CMOS camera module provides still images at VGA resolution in 16-bit color. At the time of early prototyping, the OV7670 module was the only readily-available parallel-interface CMOS camera that could be inexpensively obtained mounted to a module, since the CameraChip packages used by most integrated CMOS imagers cannot be read-

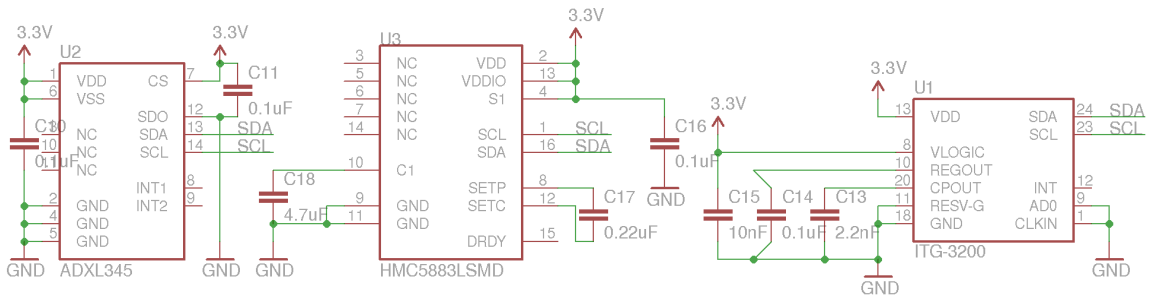


Figure 2.29: μrover inertial sensors: ADXL345, ITG-3200, HMC5883L

ily hand-soldered. As higher resolution and significant amounts of processing power are needed to use this camera for automated navigation of the μrover, a set of six Sharp GP2Y0A02YK IR range sensors are used to sense nearby obstacles and avoid collisions. These range sensors are used because the GP2Y0A02YK has the longest range of all triangulation-based IR sensors that could be obtained readily and within budget. The IR range sensors work by precise triangulation of a focused spot of infrared light reflected from an object ahead of the μrover. One side of the sensor produces a focused infrared beam directly ahead of the sensor. The other side incorporates a lens that focuses reflected light from the spot on to a small horizontal photodiode array. The closer the reflecting object is to the sensor, the further along the photodiode array the reflection will be detected, in a similar manner to the sun sensors developed to aid μrover navigation. As not enough ADC channels were available on the OBC itself for reading the IR sensor outputs, a small Atmel ATTiny461 AVR microcontroller, which is very small and efficient but has 11 ADC channels and an integrated temperature sensor, was connected to the OBC over SPI bus and programmed to read and transmit range readings when queried by the OBC.

2.5 Motor Control

Small DC motors for high-torque applications are generally either brush DC or brushless DC. In prototype development, the wheels are driven directly from brush DC motors, which have the advantage of being simple to control using pulse-width modulation of DC. However, DC motors have several significant drawbacks, primarily that the brushes that commutate power between the coils on the rotor will become worn or clogged after extended operation, and that the brushes will not always be in contact with a coil, which causes power line transients, arcing, and low efficiency. Brushless DC motors overcome these problems by placing the electromagnetic coils on the outside of the motor, and using a permanent magnet or air core rotor on the shaft. Commutation of current between coils is done by electronically switching the current using transistor bridges, which requires knowledge of the rotor position at all times. This is usually accomplished by sensing the position of a permanent magnet on the rotor with three Hall sensors set at 60° angles to one another. Therefore, no contact is necessary between the rotor and the stator other than the rotor bearings. For these reasons, brushless motors are essential for use in space applications, where reliability and efficiency is of paramount importance.

Ideally, both BLDC and DC motors could be driven with only software changes, so any changes to the system could be made by a microcontroller. Many small microcontrollers are suitable for implementing feedback and control of motors, and some PIC and AVR models include dedicated hardware for this purpose, while still being programmable in C and using minimal power. Flexible designs for multiple motors have been achieved using FPGA hardware and DSP ICs, such as in [Zerigui *et al.* 2007]. While this approach provides high control speeds, it requires more power to run the FPGA and introduces significant complexity in design and programming. To enable a common system to drive both brush DC and brushless DC motors, a hybrid approach is required. DC motors are usually controlled by using a fixed input and pulse-width modulation (PWM) to manage the amount of total power provided to the motor. Brushless DC motors are controlled by shunting power to different motor coils (or ”phases“) one at a time in synchronization with rotor speed, usually measured by Hall sensors, and speed control is achieved by

changing the time that power is shunted. Both systems usually use field-effect transistors (FETs) in a half-bridge configuration to control current flow, two for a brush motor in the ”H-bridge“ configuration, and one for each phase of a brushless motor. Brushless motors usually have up to three phases, so a hybrid controller can be built with three such half-bridges.

Motor control for the μrover was designed and tested in stages. The first stage involved establishment and electronic prototyping of the motor model and drive methodology to aid in motor selection and ensure that the commutation methods worked properly. Research was used to verify that motor performance would be acceptable at lower than nominal voltage levels, since many candidate motors are designed for a $12V$ supply and operation at $7.4V$ or $11.1V$ is necessary based on the μrover battery stack. The second stage involved prototyping and development of high-current environmentally-tolerant motor drive designs to evaluate the performance and problems of proposed designs. The York University Rover Team's $50kg$ prototype rover provided a good opportunity for extensive field testing, and a custom high-current motor control board was designed for the 2011 competition year, with lessons learned being incorporated into the μrover motor control design. In the third stage, the motor control design was hand-built on the full μrover prototype and tested in the field before the design being laid out as a schematic for PCB design.

2.5.1 Motor Model

DC motors with permanent magnets to create a constant magnetic field vector **B** (as opposed to induction motors) are characterized by two constants: the speed constant K_v, and the torque constant K_t. The speed constant K_v provides a measure of now fast the motor will run for a given input voltage, as voltage provides the motivating force (EMF) for current flow, and is defined as

$$K_v = \frac{\omega_m}{V}. \tag{2.95}$$

Similarly, the torque constant provides a measure of how much torque the motor will produce given a quantity of input current, and torque is produced as a result of the interaction of a permanent, perpendicular magnetic field **B** with an electromagnetic field from a coil of wire with n

turns of length l about a radius r that carries a current I. The equation for this is

$$K_t = \frac{\tau_m}{I} = 2nlr|\mathbf{B}|. \tag{2.96}$$

These constants are typically complimentary. A high K_t is generally desirable for high torque in robotic actuation, while a high K_v is desirable for rotational speed in rotors and propellers. For an ideal motor, the product of constants $K_v K_t = 1$. This can be seen because it involves both mechanical power $P_m = \omega_m \tau_m$ and electrical power $P_e = VI$ as

$$K_v K_t = \frac{\omega_m \tau_m}{VI} = \frac{P_m}{P_e} \tag{2.97}$$

which is less than 1 for $P_m < P_e$ which makes practical sense for a motor. If it were greater than 1, this would imply that more mechanical power is being generated than electrical power put in, which could only happen if it were acting as a generator. So for a motor fed by DC current, $K_v K_t$ refers to the peak electromechanical *efficiency* of the motor, and should be as high as possible. This is a design parameter, and is generally maximized in a given motor. As the motor actually spins, the movement of the permanent magnetic field against the coils induces an additional voltage, which is naturally oriented in the same direction as the applied voltage to the motor, again working like a generator. The magnitude of voltage depends on the speed constant K_t, and is

$$V_{EMF} = \frac{\omega_m}{K_v}. \tag{2.98}$$

This V_{EMF} is known as the back-EMF, and has the effect of cancelling part of the applied drive voltage to the motor, which is a very good thing, because without this back-EMF, the coils would have only their very small characteristic DC resistance R to resist current and I would remain very high! Fortunately, the resistance of the motor is only R at the moment of start-up or when stalled by external torques. At any given time, the electrical losses to heat within the motor are dominated by I^2R. We can more completely define the voltage V_m that must appear at the motor terminals by adding together the voltage contributions of the minimally resistive wire IR

using Ohm's law, the back-EMF V_{EMF}, and the voltage generated as a result of changing current through an inductor, which naturally produces a voltage to resist current change. The complete equation is

$$V_m = IR + L\frac{dI}{dt} + \frac{\omega_m}{K_v}. \tag{2.99}$$

To estimate how a motor will respond, we can solve this equation for ω_m.

$$\omega_m = \frac{V_m - IR - L\frac{dI}{dt}}{K_v} \tag{2.100}$$

Note that this equation does not give us a simple closed-form solution for the motor speed as the current I and applied voltage V_m are related, and the differential term causes the system to change when the current changes. However, in steady-state unloaded conditions, we can assume that $dI/dt = 0.$ and the term IR is constant. Halving V_m will generally also have the effect of halving I, and therefore also halving ω_m. This model does not consider many other factors such as changes in temperature, bearing friction, and loading. However, it does indicate that speed will scale linearly with applied voltage, all else being equal. If the motor has to apply a torque to a load at the output shaft, this will cause a decrease in ω_m, which causes a decrease in V_{EMF} and an increase in I. Neglecting frictional losses, the amount of torque applied must be equal for both the load and the motor, and is determined by Equation 2.96.

2.5.2 Drive Topologies

For the model to be valid, the orientation of the active motor coil in a right handed movement system with respect to the permanent magnetic field must be fairly consistent. This is why motors must have several sets of windings, or *phases*, and commutate current through them sequentially. To increase torque and decrease speed, each phase can have several coils known as *poles*, each of which carries the current in its phase and is placed in staggered order with the other phases. Brush DC motors are mechanically commutated, and can have many poles, but usually only run current through a single coil at a time, and the more poles that are present, the more smoothly the

motor runs as the angle between the phase and the permanent magnetic field is more consistent. Brushless DC motors typically have three phases separated, which are interconnected at three separate terminal points where current can enter or exit, and effectively operate in the same way as three-phase permanent magnet synchronous motors, the main difference being the way that the motor is driven - from switched DC bridges rather than AC sinusoids. There are two kinds of winding styles signified by symbols that mirror their topologies, and the Δ winding is less popular than the simpler, centrally-connected Y winding. However, to be able to drive brushless DC motors, each pole must be electronically switched to allow current through only one path in one direction at a time. One pole is held at high input voltage V_m, one pole is held at GND, or $0V$, and one pole is disconnected. In a Y wound BLDC motor, this will force current through two phases at a time in one direction, with one sourcing and the other sinking the drive current.

Although it is possible to measure the rotor position by sampling the back-EMF created as each coil is passed by the rotor (known as sensorless BLDC), reliable detection of rotor position requires higher rotor speeds and a higher K_v to create a larger V_{EMF}, which in turn requires higher gearing and correspondingly higher losses in the gearbox. To avoid this, sensored motors are preferred for better speed and position control of the motor at low rotation speeds such as those in high-torque robot drivetrains. The operation of a sensored BLDC motor was tested using both software and hardware commutation methods in the first stage of driver development and proved that both are viable for motor control.

2.5.3 High-Current DC Drive Prototype

As the second stage of the development and testing of the drive system, a high-current four DC motor drive board was developed for use on the 50kg rover used by YURT in 2011. As a pure DC drive system, it used discrete H-bridges and PWM for current control. As the stall current of the $24V$ DC motors used was $17A$ and little capacity for heatsinking was available, the driver was designed with very large current capacity. Each motor drive channel uses four flat-mounted IRFP3206 $120A$ N-channel power MOSFET transistors in an H-bridge configuration. To increase capacity to handle back-EMF, regenerative breaking, and inductive spikes, a reverse-

biased STPS20120 20A diode is connected in parallel with each MOSFET. Current sensing for each channel is accomplished with CSNX25 25A integrated hall-effect current sensors. Traces for the high-current parts of the two-layer board are expanded and reinforced with soldered wire to allow the driver to provide a continuous 25A per channel in the forward or reverse direction. The design is complicated by the fact that like many power MOSFETs with high gate capacitance, the IRFP3206 MOSFETs require a high gate-to-source voltage and a large instantaneous gate current for efficient turn-on. This was provided by using an IR2104 dedicated half-bridge MOSFET driver IC on each side of the H-bridge, which also provides a PWM input and high-low side select that prevents shoot-through and greatly simplifies control. However, the IR2104 has a relatively narrow maximum voltage tolerance of 20V and no suitable half-bridge driver substitute was available, so the 24V input was stepped down to 15V using an LM2576 step-down switching regulator.

The motors are controlled using an ATMega644P AVR microcontroller in a similar fashion to the μrover devices. However, due to the high currents and voltages used, the microcontroller and all low-voltage electronics are physically isolated from the high-power components using ILD2 optocouplers for control signals and high-speed HCPL3140 optocouplers on the PWM channels. TTL inputs on the microcontroller are used to read the four motor speeds for feedback and speed control. Board temperature and voltage sensing is also implemented. A serial port is used for communications, and provision is made for an Ethernet connection using an ENC28J60 SPI PHY chip as well. For easy assembly and modification, all through-hole parts were used on this board. This required a very large 260mmx106mm board area to fit all the components, which was suitable for the YURT rover but much too large for a μrover. A schematic of this motor drive is shown in Figure 2.30 and the assembled board is shown in Figure 2.31. Regardless, many useful lessons were learned and applied to the design of the μrover drive system.

The YURT drive board functioned quite well overall, and was able to provide smooth speed control while under high loads in the harsh conditions of the Utah desert in URC2011. However, after extended use some limitations became apparent. The optocouplers and IR2104 MOSFET gate drivers could not switch fast enough for ultrasonic operation (being limited to several kilo-

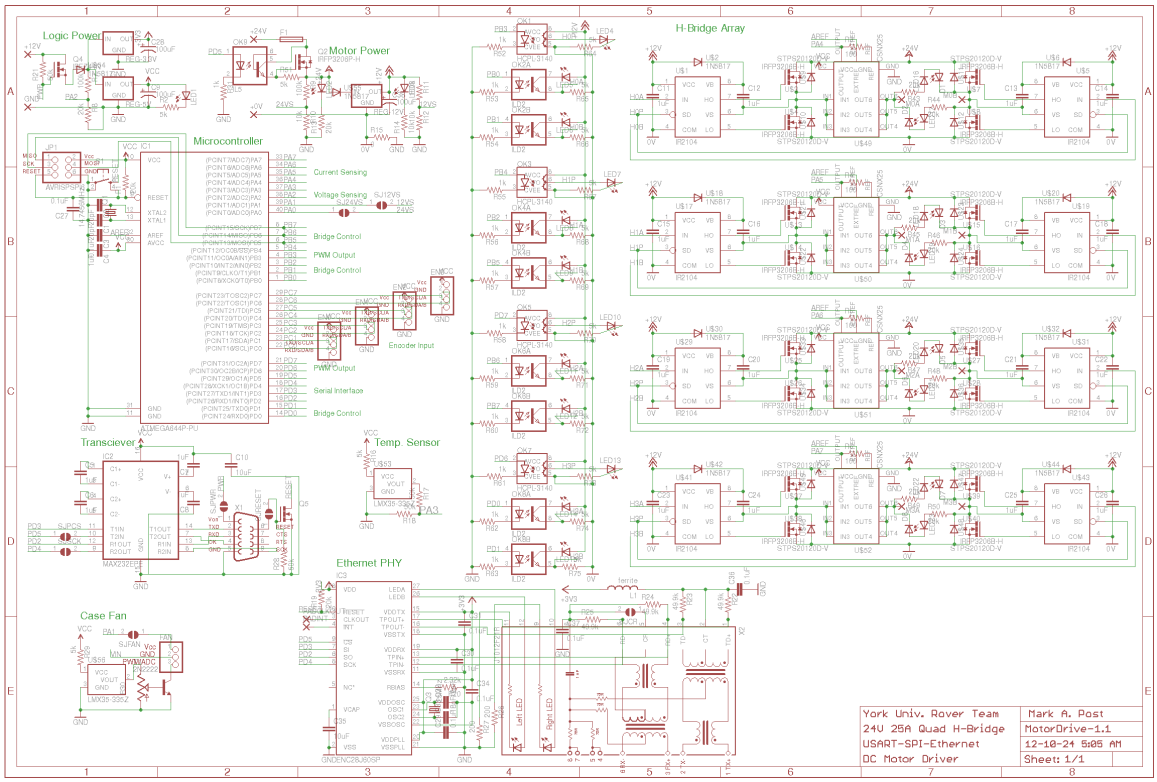


Figure 2.30: YURT 2011 quad motor drive schematic

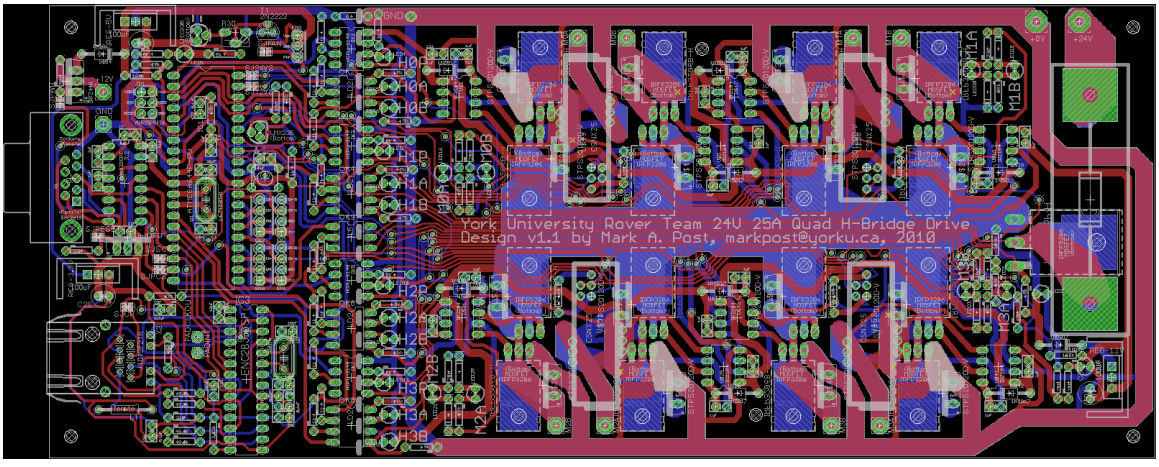
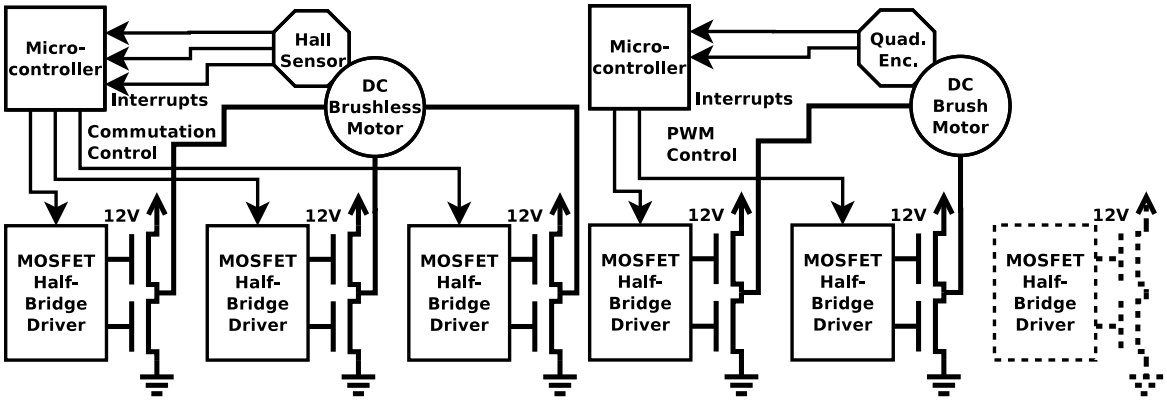


Figure 2.31: YURT 2011 assembled motor drive board

hertz) and created considerable inductive vibration noise in the motor coils. The original step-down power supply for the IR2104 used a linear regulator which was prone to overheating and failure. After replacement with the LM2576 switching regulator, higher reliability was achieved with the exception of occasional failures of the attached IR2104 ICs. The cause of failure for the IR2104 was never positively identified, but it is suspected that prolonged operation under high voltages in the system and transients that likely often travelled through the common ground and breached the protection of the LM2576 regulator were sufficient to degrade the chip's performance until failure occurred. Rather than redesign the gate drive circuit, it was decided to use fully-integrated power half-bridges designed for the automotive market that had become recently available. The use of TTL control voltages and integrated fault protection obviated the need for gate drives and physical isolation, and more compact packaging coupled with the use of surface-mount components made it possible to fit four motor drivers within a PC/104 form factor while retaining significant current capacity and integrated control.

2.5.4 Hybrid Drive Methodology

Brush DC motors are more common and inexpensive, while brushless DC motors are harder to source, but are more efficient and reliable [Lee *et al.* 2003b]. Space-qualified μrovers will be equipped with brushless motors, but for testing purposes and overall flexibility, it would be ideal if both BLDC and DC motors could be driven with only software changes without trade-off of BLDC operation. To enable a common system to drive both brush DC and brushless DC motors, a hybrid approach is required. DC motors are usually controlled by using a fixed input and pulse-width modulation (PWM) to manage the amount of total power provided to the motor. Brushless DC motors are controlled by commutating power to different motor coils (or "phases") one at a time in synchronization with rotor speed, usually measured by Hall sensors, and speed control is achieved by changing the time that power is shunted. Both systems usually use field-effect transistors (FETs) in a half-bridge configuration to control current flow, two for a brush motor in the "H-bridge" configuration, and one for each phase of a brushless motor. Brushless motors usually have up to three phases, so a hybrid controller can be built with three such half-bridges.



(a) Brushless Motor Configuration (b) Brush DC Motor Configuration

Figure 2.32: DC motor controller basic configurations

Figure 2.32 shows the hybrid system in both brushless DC (a) and brush DC (b) configurations. For three-phase brushless use, a single set of three half-bridges are cycled in sequence to supply power to the three motor phases, while for brush use, two of these form an H-bridge to supply pulse-width modulated DC power to the motor. Control feedback is via a 3-phase Hall sensor in the brushless case and bridges are commutated by setting GPIO lines, while a quadrature (Gray code) encoder is used in the brush case and pulse-width modulation control is used in hardware. A supply of 12V is assumed to be the drive voltage in this diagram, but any appropriate motor voltage can be used. Each motor requires the use of one set of drive bridges, though a fast microcontroller can usually drive more than one bridge at a time. For small mobile robots and rovers, all needed drive motor controllers can usually be implemented on an OBC daughterboard, but for larger robots, the drive controller is implemented as an external payload to gain more space and better isolation. A hybrid system of this type has been built for the micro-rover, and shows promise for larger systems as well. The pinouts for the different kinds of motor connectors that can be used are shown in Figure 2.33.

In testing, it was found that the microcontroller must perform commutation consistently in-phase with the rotor for efficient running, but since the microcontroller is single-threaded and interrupt-driven, it will lag slightly if more than one motor requires commutation at a given moment. To remedy this, a faster or more complex microcontroller may be used, separate mi-

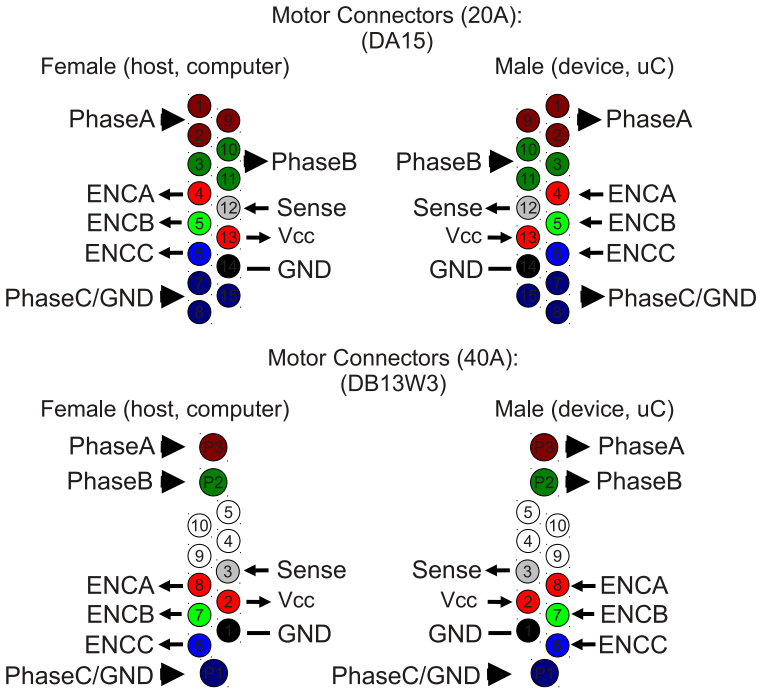


Figure 2.33: μrover Motor Drive Ports

crocontrollers for each phase may be used, or decoding of the Hall sensors in embedded logic hardware may be used. A logic circuit was synthesized from the commutation tables used for a three-phase brushless motor and is shown in Figure 2.34. Using dedicated logic has the added benefit of unloading the commutation task from the microcontroller while retaining centralized multiple-motor control, but the limitation is that auto-commutation for motor starting is not built in and must still be initiated directly by the microcontroller. With the addition of signals that manually commutate the motor phases for starting, this dedicated logic is feasible, and fallback to brush DC motor operation is also possible.

2.5.5 Embedded Drive Controller

The four-channel hybrid drive controller developed for the μrover must be both physically compact and tolerant of high currents so that minimal heating will occur in vacuum or near-vacuum conditions, and extreme temperatures will not affect the electrical operation of the motors. At the same time, it is essential to conserve power and for the system to operate at low voltages for

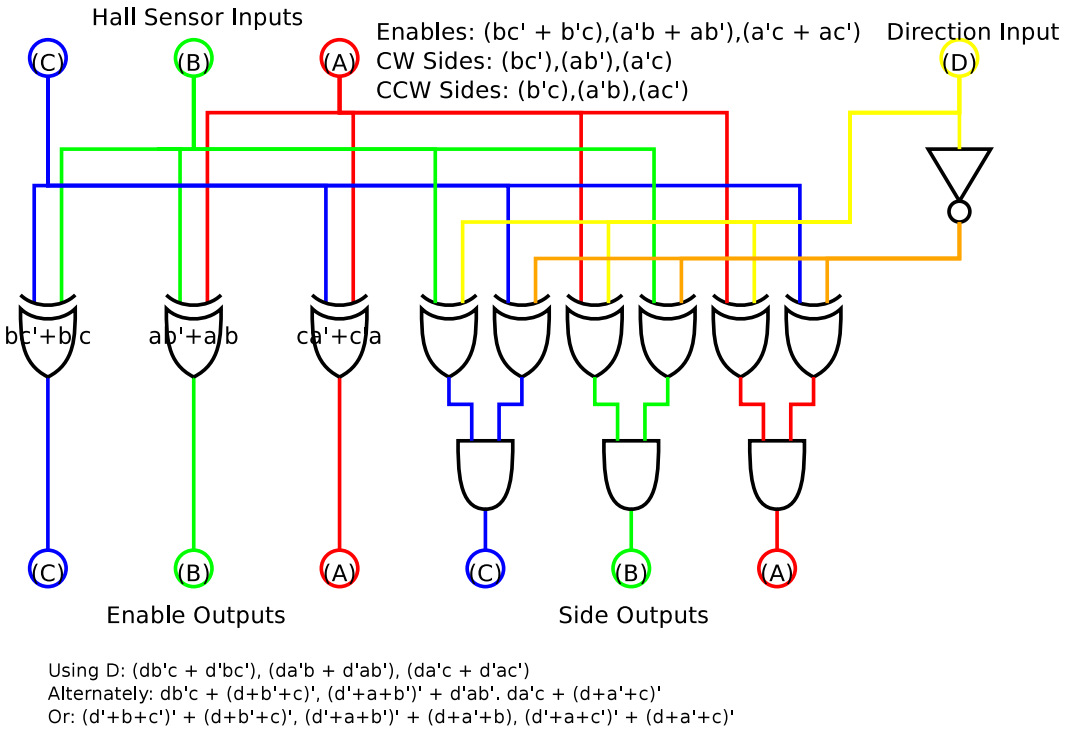


Figure 2.34: Commutation logic for driving a three-phase brushless DC motor

microcontroller signalling to work efficiently. To accomplish these goals, it is necessary to use as many highly-integrated components as possible to minimize component count, space, and complexity. The hybrid drive topology requires three-half-bridges that can be individually controlled, while maintaining some measure of thermal and current control for safety, and while commutation through pure logic is preferred, it makes the wiring and control significantly more complex, so the current drive motor controller uses pure software control through GPIO pins, although hardware timers are used to provide PWM and commutation timing within the microcontroller. Although early motor controller prototypes used the ATMega644P microcontroller as the first μrover prototype and high-current DC motor drive board did, the large number of pins necessary to control four brushless DC motors and read the suspension and inertial sensors required the use of a larger 64-pin ATMega1281 microcontroller, which is still able to be hand-soldered and shares a common architecture and peripheral interface layout with other AVR devices to greatly simplify the porting of code. The pinout used for the ATMega1281 is shown in Figure 2.35,

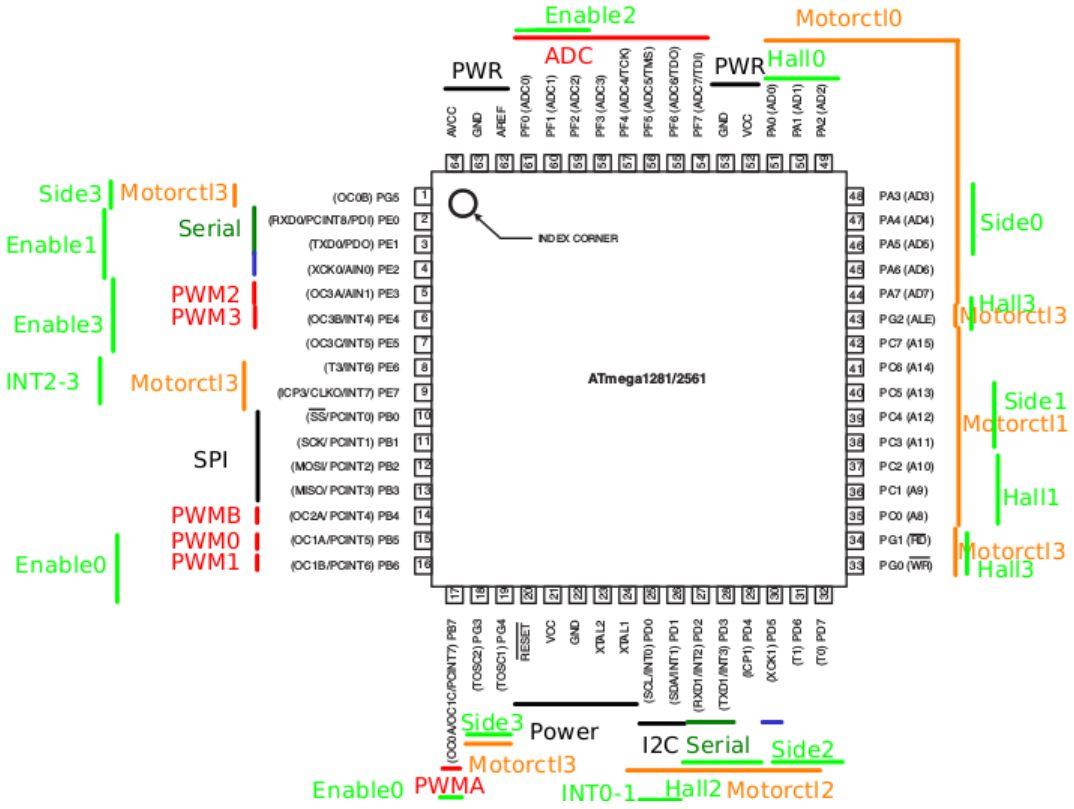


Figure 2.35: ATMega1281 Connections for Motor Control

superimposed on the ATMega1281 pinout from the Atmel datasheet [Atmel 2012], and the drive controller communicates with the OBC over an SPI bus.

Based on experience gained in the development of large-scale motor controllers for YURT and DC motor control for the μrover prototype, the controller for the third stage of motor drive development was designed around the use of half-bridges with integrated switching and current protection designed for power electronics in the automotive industry. The Infineon BTN7971B was chosen as the most appropriate basis for a low-voltage high-current hybrid drive. The value of the BTN7971B is that it has a minimum $50A$ and peak $70A$ current capacity with a total path resistance of only $16m\Omega$ at $25°C$, and also has separate high/low side selection and PWM on/off pins to simplify control. Slew rate adjustment and error flag pins allow fine tuning and monitoring of controller status. As the supply range is $3V$ to $28V$, the bridges can operate at a very wide range of voltages and be controlled directly from the microcontroller pins even with

$10k\Omega$ resistors recommended for reducing load on the GPIO drivers. The space saved by not having to include many other external components allowed four channels to be built within the PC/104 form factor, and the wiring and traces used on the drive board are made much larger than usual to ensure that minimal resistance is encountered outside of the half-bridges. Due to the level of specialization, these half-bridges are relatively expensive and hard to obtain, and the original drive prototype used the $40A$ BTN7960B instead, which still performed very well under all conditions including thermal vacuum.

2.6 Sun Sensors

Because Mars and the Moon have magnetic fields that are not strong and uniform enough for magnetic navigation by the use of a magnetometer, other methods such as sun sensing must be used to establish heading vectors and coarse positions. Two different methods of sun sensing for μrover navigation were tested in hardware and compared [M.A.Post *et al.* 2013]. First, direct measurement of the solar angle is performed using a photodiode array sensor placed below a slit design in the top that allows light to fall on the sensor element. Second, the solar angle is inferred using separate current measurements from the array of solar cells used for power generation on the exterior. In both cases, the solar angle is calculated by a microcontroller from the geometry of the sun sensor with respect to the body.

It is necessary to build both sun sensing implementations from scratch because commercial sun sensors suitable for small space hardware are purpose-built and extremely expensive, and most are also too large or power-hungry for use on the μrover. For this study, it is assumed that at least a 90° field of view is necessary for each sun sensor, so that complete coverage of the exterior is possible with a sun sensor on each face of the rover body. Figure 2.36 shows the μrover with solar panels covering three sides, so that sun vector localization can be done. The array-based sun sensor is placed in the rear payload enclosure, and is not visible as the μrover is pictured. Note that this figure shows the μrover in a testing configuration before the motors could be geared to right angles from the wheels. To test and validate the sensors, a prototype nanosatellite ADCS is rotated on the spot by a rotary gimbal with angular measurement, and simulated sunlight is provided at a fixed position. Using embedded fixed-point processing, measurements are processed and made available by serial interface to the on-board computer, which performs the filtering and attitude determination operations [S. Chouraqui 2005].

2.6.1 Array Sensor

An established and simple method for directly sensing solar angles is to use a simple slit or other aperture design through which light is allowed to fall at one location on a spatially-varying

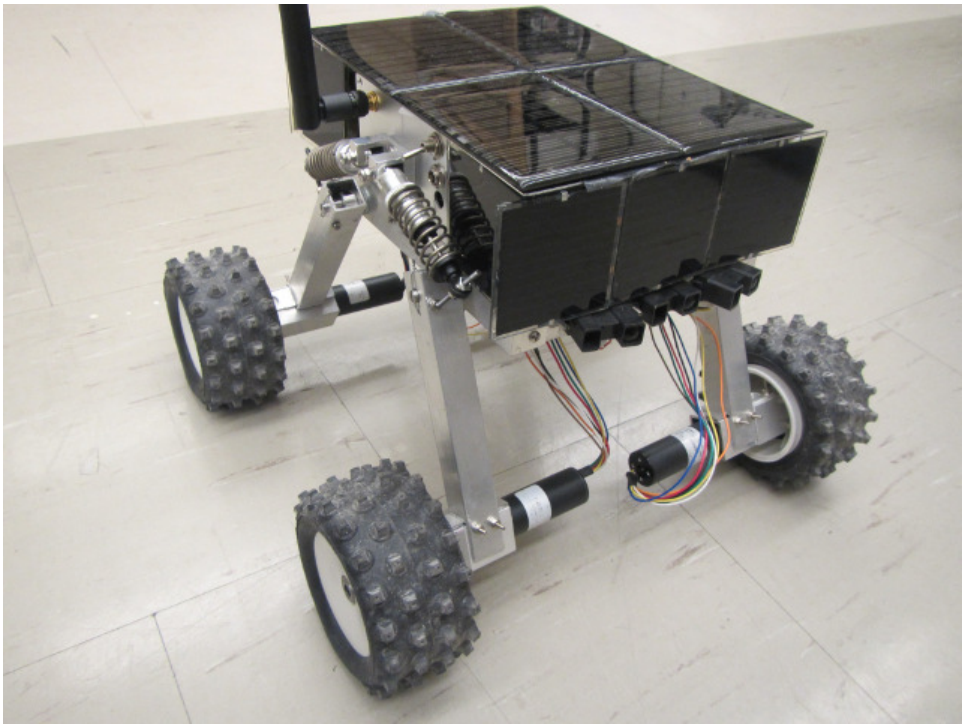


Figure 2.36: μrover with Sun Sensor and Solar Panels on Three Surfaces for Testing

sensor, dependent on the incident angle of light. This design uses a 256-element linear photodiode array, which is both flexible in operation by allowing the distribution and magnitude of light to be determined, and efficient by utilizing only a single line of sensing elements rather than a matrix such as a Charge-Coupled Device (CCD) array. While some systems use a Position-Sensitive Diode (PSD) as a sensor, an array increases linearity of measurement and provides more flexibility in processing of light distribution data. The TAOS TSL1402R integrated linear photodiode array is used in this design, and represents the smallest and highest-resolution of photodiode arrays that were readily available. Reading of data and calculation of solar angles is performed by an Atmel ATMega168PA AVR microcontroller, chosen for its size, ease of use, and code portability to other AVR devices, and connections are made as shown in Figure 2.37 [Post *et al.* 2013].

For single-axis sensing, a simple linear slit in a mask over the sensor is used, as shown in Figure 2.38. The slit is positioned vertically centred on the sensor element such that light falls

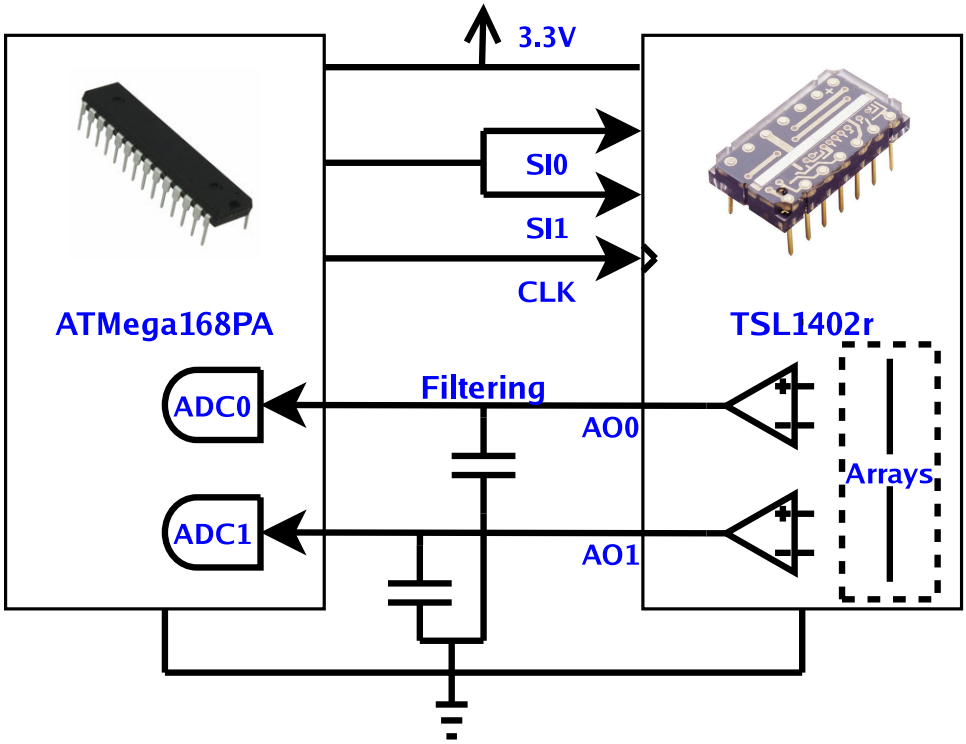


Figure 2.37: Diagram of Photodiode Array Sensor

on to the sensor array at angles up to 45° in every direction for a 90° total field of view. For a linear photodiode array with length L, a light vector falls at an angle θ from the normal along the element axis, and at an angle ϕ from the normal along the perpendicular to the element axis. The photodiode voltage output of the sensor falls at the point of contact with the light vector, at a distance d from the center of the array. If the slit is positioned at a distance h from the array, this distance is simply

$$d = h\tan(\theta). \tag{2.101}$$

The voltage output must be inverted so that illumination becomes positive. Then, a centroiding algorithm [M.-S. Wei & You 2011] is used to determine the center of illumination across the sensor. The centroid positions d are then used in Equation 2.101 for each angle n sampled to determine the corresponding solar angle by

$$\theta_n = \text{atan}(d/h). \tag{2.102}$$

To ensure that $|\theta| \leq 45°$, the sensor distance should be $h = d = L/2$, and to ensure $|\phi| \leq 45°$, the slit length should be $b = 2h = L$. It is preferable for higher precision measurements to use as small a slit width a as possible without significant diffractive effects, but the material thickness around the slit is constrained by $c < a/\tan(\theta)$, which limits how small the slit width a can be, as the beam of light will be "pinched" off at high angles. In this design, $L = 16mm$ and the dimensions used were $h = 8mm$ and $b = 18mm$. In testing, metal foil with width $c = 0.5mm$ provided acceptable performance using a slit width of $a = 0.8mm$.

Linear arrays typically provide only one axis of attitude estimation, but because a pattern on the array can be measured, it is also possible to measure elevation across most angles by using an N-slit [M.-S. Wei & You 2011], as shown in Figure 2.39. By adding additional slits positioned at an angle δ to the central slit, a pattern with multiple illumination maxima will be projected on to the sensor array, and the transverse angle ϕ can be determined by the separations $d1$ and $d2$ of the central slit and one of the side maxima.

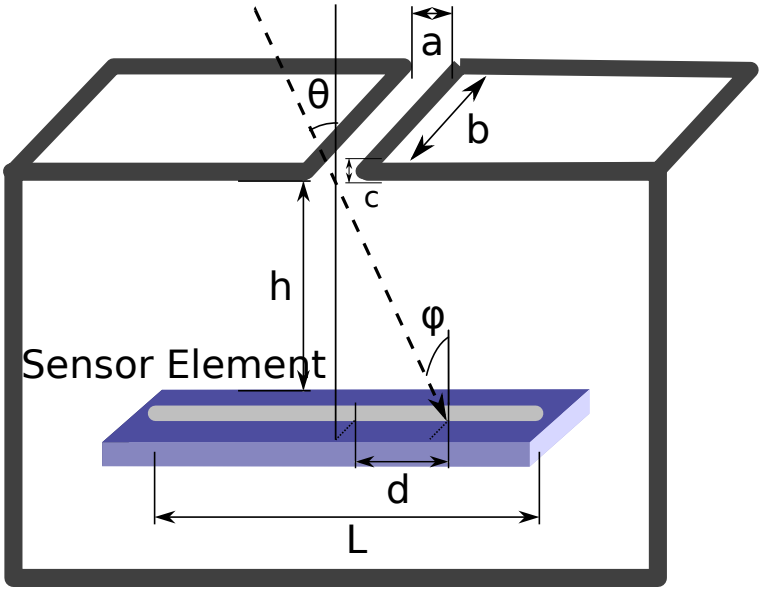


Figure 2.38: Sun Angle Sensing by Photodiode Array

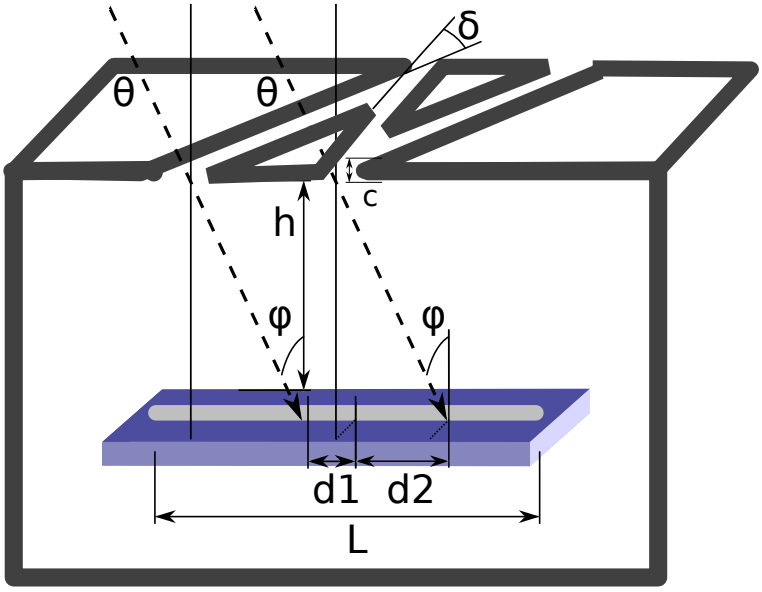


Figure 2.39: Sun Angle Sensing by Photodiode Array and N-slit

As the field of view of the sensor is limited, more than one sun sensor is needed to cover wide angles. Two single slit sensors or one N-slit sensor can cover two solid angles of 90° with a pyramidal volume of view. For μrover use, a full 180° of view (horizon to horizon) can be covered by four photodiodes attached to the frustum of square pyramid [Toyokazu *et al.* 2009] [Springmann & Cutler 2013]. In the current study, though, a single N-slit sensor is being studied for partial sky coverage on the μrover to minimize space required for additional electronics within the chassis and surface area required for sensors around solar panels on the top of the rover body.

To estimate the pose of the μrover from solar angles, the angle of the body with respect to the ground and the angle of the sun with respect to the ground must be considered. The former can be obtained with inertial measurements from an accelerometer at rest measuring the gravity vector, and the latter can be obtained from solar ephemeris data and current time. Assuming the accelerometer is aligned with the sun sensor, with the angle θ about the forward-pointing X-axis and the angle ϕ about the Y-axis simplifies the analysis. The rover body angles about the X and Y axes with respect to the ground frame θ_g and ϕ_g can be calculated from the gravity vector (g_x, g_y, g_z) for angles less than 90° using common aerospace relations.

$$\begin{aligned} \theta_g &= \text{atan}\left(\frac{g_y}{g_z}\right) \\ \phi_g &= \text{atan}\left(\frac{-g_x}{\sqrt{g_y^2 + g_z^2}}\right) \end{aligned} \tag{2.103}$$

To obtain the sensed horizontal azimuth α_s of a vertically-centred sun sensor measurement, the angles θ and ϕ must be adjusted for rover body angle and projected on the horizontal plane using the quadrant-aware arctangent function.

$$\alpha_s = \text{atan2}(sin(\theta + \theta_g), sin(\phi + \phi_g)) \tag{2.104}$$

The solar ephemeris data must be computed separately from an estimate of the current position, which can be done with a variety of available software. The rover's heading with respect

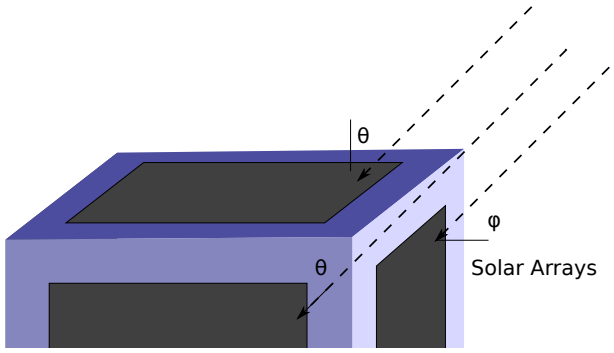


Figure 2.40: Sun Angle Sensing by Solar Cell Output

to true north α_{rover} can then be calculated using the solar azimuth α_e and sensed azimuth α_s by [Trebi-Ollennu *et al.* 2001]

$$\begin{aligned} \alpha_e > \alpha_s \Rightarrow \alpha_{rover} = \alpha_e - \alpha_s \\ \alpha_e <= \alpha_s \Rightarrow \alpha_{rover} = \alpha_s - \alpha_e. \end{aligned} \tag{2.105}$$

2.6.2 Solar Current Sensing

An alternate method of sun sensing on the μrover was developed that focused on processing of other available data, rather than discrete sensors. One method of doing this is to sample the currents generated by the solar arrays which is generally available for peak-power tracking or battery charge monitoring. This has the advantage that a range of angles spanning a set of independently-measured non-coplanar solar panels can be measured without an external sensor, but is in general less accurate than discrete sensors due to the nonlinearities involved. In this study, we consider a cubic body with a fixed solar panel on each orthogonal face, as shown in Figure 2.40. No more than three solar panels are exposed to sunlight at a time if a point source and negligible reflection from the earth are assumed, with angles θ_1, θ_2, and θ_3. For simplicity, a cubic body is assumed for the μrover and the solar panel measurements are scaled in practice to compensate for the different panel areas on the different sides.

The linearity of this measurement varies depending on the solar cells used. Also, nonlinearity

are introduced by variations in the load presented to the solar arrays. For a nanosatellite with a linear or pulse regulated battery charge system, this generally arises from changes in battery charge rate as the battery state changes, and can be compensated for by including solar cell voltage or an estimation of the charge system state in the solar calculation to determine total power and current output.

The current sensing circuit is constructed from a bank of differential amplifiers that are read by using the Analog-to-Digital Converter (ADC) channels available on the ATMega168PA microcontroller that also reads the linear array. A current sense resistor of 0.1Ω creates a voltage difference from current flowing from the solar panels, which is amplified by an OPA2340 rail-to-rail op-amp, chosen because of its linearity and excellent rail-to-rail voltage performance as proven in several related projects. The op-amp is used in differential configuration with a gain of 100, connected are made as shown in Figure 2.41 [Post *et al.* 2013]. The output gain with respect to the solar panel current is then $10V/A$. It is assumed that no more than $500mA$ will be sourced from the solar panels, so an ADC reference of $5V$ can be used in measurement. As custom-constructed solar panels often vary slightly in output, it is still necessary to calibrate the ADC measurements performed by the microcontroller.

The amount of current a solar panel produces increases with on the total area of sunlight intercepted by the panel area A_0, as changes in angle change the effective area A_e of the solar panel intercepting solar energy. For sunlight with an even power density (W/m^2) and constant loading or by using compensation calculations, the effective solar current I_e relative to the maximum solar current I_0 intercepted by the solar array varies with the incident angle θ can be expressed in general terms as a ratio

$$\frac{I_e}{I_0} = \frac{A_e}{A_0} = \cos(\theta). \tag{2.106}$$

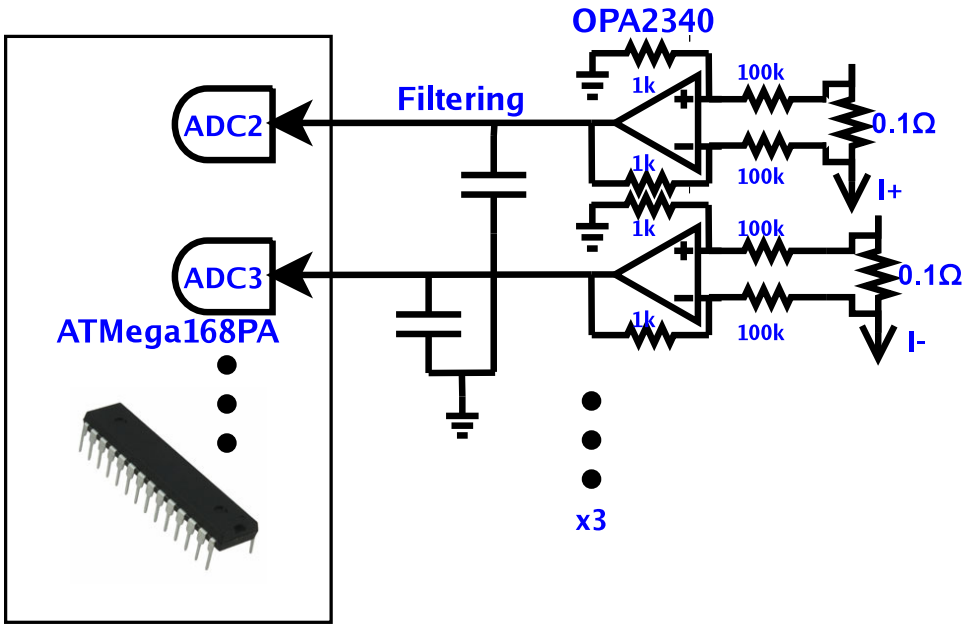


Figure 2.41: Diagram of Solar Current Sensors

2.7 Vision Board

As development of the μrover progressed, it became evident that the AT91RM9200 OBC microcontroller was not well suited to the task of processing image data for navigation. To allow high-speed processing of visual data from one or two cameras, a vision board was developed that is based on the $500MHz$ Analog Devices ADSP-BF537 Blackfin DSP processor. The ADSP-BF537 was chosen because several other vision boards have been developed for the BF537, the level of Linux and API support from Analog Devices is known to be quite good, and it is well-suited for fixed-point image processing and algorithm implementations. Similar boards include the Surveyor SRV-1 [Cummins *et al.* 2008] and the open-hardware LeanXCam [Kwiek *et al.* 2010]. Figure 2.42 shows a schematic of the current board version under testing. After significant effort in manual routing, the BF537 in BGA208 package was successfully connected using only two signal layers and $0.17mm$ clearances to break out all pins, meaning that a 2-layer board could be fabricated with standard clearances, but hot air or infrared rework equipment is needed to place the BF537 IC.

The better-known SRV-1 was used as the basis for the vision board design, but significant changes were made based on desirable aspects of the LeanXCam and to allow use of more easily-available and robust components. Primarily, multiplexing of two cameras for use in stereo vision was implemented by using the lower 8 bits of the Blackfin's 16-bit Parallel Peripheral Interface (PPI) for the left camera and the higher 8 bits for the right camera. The camera headers were also rearranged to connect to an Omnivision OV7725 camera breakout board, which is similar and largely compatible to the OV7670 camera originally used for the OBC, but provides higher image quality and frame rate. Also, a JTAG interface, modified internal voltage generation circuit with linear regulator backup, MAX1626 external voltage supply, and extra breakout pins were added to increase flexibility and make the board useful for the μrover. Also, provision was made for an ethernet PHY and jack based on the LeanXCam so that the camera could be used in other projects for high-bandwidth network video capture.

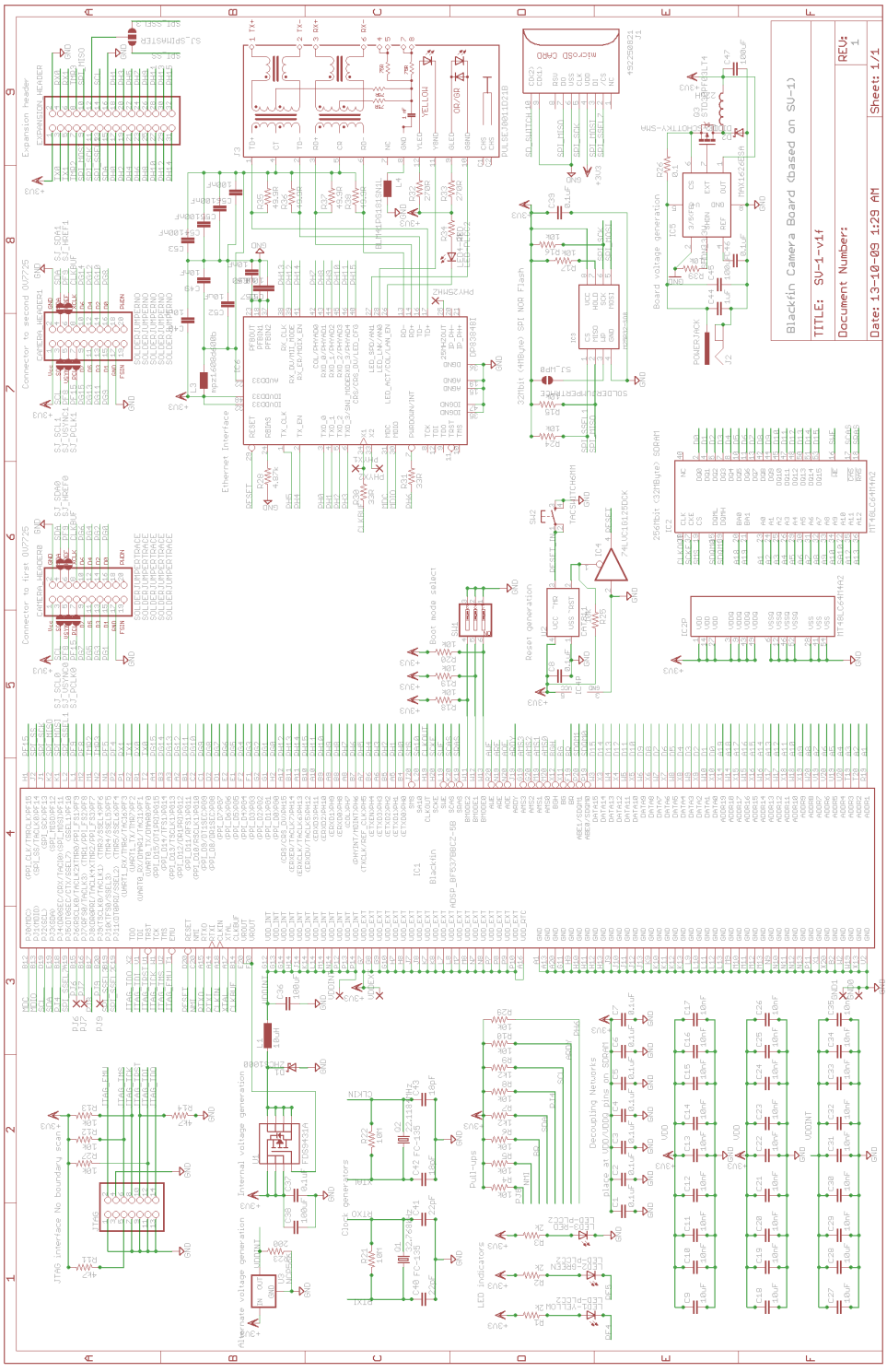


Figure 2.42: Blackfin-based camera vision board

2.8 Language Selection

Like virtually all modern robotic systems, the μrover is controlled by programs running on computer hardware. As both system complexity and power consumption must be minimal for reliable operation in harsh environments, code reliability and operational efficiency are critical for successful operation. In the past, languages used for aerospace systems and robotic programming included Ada, Fortran, Lisp, and a variety of less-known languages such as ALGOL and PROLOG [Anderson & Dorfman 1991]. Ada in particular is considered to be an excellent language designed specifically for aerospace projects and embedded, real-time systems, but due to a lack of both modern programming tools and trained programmers, the language faces an uncertain future even within the aerospace community [Smith 2003]. There are dedicated, proprietary, and consequently very restrictive languages used exclusively for space hardware as well such as STOL, PLUTO, and others. However, as the μrover is built on free software and is expected to operate on modern hardware, it is prudent to restrict the languages in use to those that are actively used in the FOSS community and supported by the Gnu Compiler Collection (GCC) or other open architectures. There are several such candidates for the main programming language for use on the μrover. Languages that have been considered for use on the μrover include:

- C/C++
- Java
- Python
- Octave/MATLAB

C and its object-oriented descendant C++ are the most widely-used languages in robotics. They compile directly to native machine language using GCC, have the most efficient execution of the candidate high-level languages, and generally offer the highest portability to embedded systems. We group C and C++ together as C++ is backwards-compatible to C and the GCC supports both languages on most platforms. While variants such as Objective-C and Objective-C++

exist, they generally have more runtime overhead due to the use of dynamic runtime libraries, VMs, and garbage collection to remove unused objects, and are less commonly used on embedded platforms and for FOSS libraries. Even though high-level concepts such as message passing and dynamic typing are handled better in many cases by other languages, C-based languages retain popularity for general embedded use as they are compiled directly to standalone machine code. However, as they are relatively low-level in terms of programming abstraction, C is essentially the most time-consuming and difficult language to program in even considering the vast number of libraries and existing applications. C++ offers greater abstraction and built-in object oriented concepts, but at the cost of greater overhead and some level of incompatibility with C code and certain embedded systems. Nonetheless, Linux and most available real-time operating systems are programmed in C, retaining the highest level of compatibility with native C code, and nearly all embedded systems support C compilation, frequently as an alternative to direct assembly programming. It is this degree of portability, re-usability, and efficiency that make C, and to a similar extent C++, attractive languages. Note that it is not necessary to use an "object-oriented language" to write object oriented code. It is simpler to create objects when object-oriented abstractions are built into the language, but conversely, it is not necessarily more efficient due to the overhead of managing those abstractions automatically. The Linux kernel itself is an example of an object oriented program written in C, a non-object-oriented language.

Java is an object-oriented VM-based language that is extremely popular for online and cross-platform programming. The use of a dedicated VM allows bytecode as well as source code to be easily ported between platforms and run more conveniently within secure systems. The syntax of Java is very similar to C++, with exceptions being mainly high-level object-oriented functions and handlers [Gosling *et al.* 2005]. However, this also means that for purposes of low-level programming, Java adds little to the basics of C++ and adds the requirement of a VM and the overhead of functions such as garbage collection. There are many math and control libraries available, usually adding classes that provide abstract concepts for specific uses. One of the main drawbacks to Java has traditionally been that despite the wealth of open-source code and libraries available for it, the virtual machine required to run it is essentially proprietary,

having been developed and maintained originally by James Gosling at Sun Microsystems and more recently by Oracle Corporation, after they acquired Sun in 2009. Although Sun released the javac compiler under the GPL in 2006 and the VM and libraries for Java have been made progressively more open, the number of embedded platforms that Java runs on is still relatively limited, and Linux support has often lagged behind other operating systems. An alternative is the Dalvik VM, developed by Dan Bornstein and Google as the main development language for the Android OS, which runs a derivative of the Java language. Dalvik differs from the Sun Java VM in several ways, but mainly in that it is a register-based architecture as opposed to the original stack-based Java architecture [Bornstein 2008]. The OpenJDK project now provides an almost completely open implementation of Java as well, with only the SNMP components being proprietary. The many derivatives of the original Java, though, have made the original Java mandate of "universal compatibility" somewhat complex, and although many Java-based robots exist, it is not an ideal system for embedded robotics.

Python is a high-level object-oriented language created by Guido van Rossum in 1991 that has become one of the most popular programming platforms for common use in modern computing. It combines many of the popular features of C++, Java, Haskell, Lisp, MATLAB, and other popular languages, uses strong and dynamic typing, and runs on an interpreter that can operate both as a scripting engine and as a VM for compiled Python bytecode [VanRossum & Drake 2010]. Due to this high level of flexibility, Python bytecode runs slower than machine code or a dedicated Java VM. Some measure of improvement can be made by just-in-time compilation of known types and structures into accelerated code, which is the method used by the Psyco compiler [Rigo 2004]. In addition to the original C-based Python interpreter, Python implementations have also been written in several different languages, including Pypy which is written in Python itself [Biham & Seberry 2006]. There have been a very large number of class libraries developed for Python, including ports of OpenCV and GStreamer for video work, low level I/O libraries such as Portio and socket, and the Numpy and SciPy mathematics libraries that allow Python to operate in a similar manner to MATLAB - as a mathematics-centred scripting language - but with significantly more flexibility. There has been interest in using Python as a satellite opera-

tions language that is much more productive, expandable, and portable than most space-centric languages. After being properly audited and scheduled, Python bytecode can be uploaded and stored for execution on a satellite [Garcia 2008]. Python has not yet been used in orbit, but it has already been adopted as a language for terrestrial robotics by several groups, including the Pyro API for use in robots [Blank *et al.* 2003]. Also, some effort has been made to make Python run natively on resource-constrained embedded systems. For example, the PyMite project targets a native Python VM on 8-bit microcontrollers in as little as 64KiB of Flash and 4KiB of RAM [Dean 2003].

Octave and MATLAB are largely cross-compatible mathematical interpreted languages. While MATLAB is a proprietary Java IDE and a set of associated toolboxes with a significant number of machine-specific optimizations and specialized functions, Octave is a completely GPL open-source console application written in Fortran and C++. A large amount of user code is available for both, and Octave has been effectively developed into an open-source alternative to MATLAB by attempting to mirror most of the basic MATLAB functions so that code will often run identically on both systems. One significant difference is that MATLAB is only developed and optimized for x86-based PC hardware, while Octave has been ported to other platforms such as ARM, which makes it possible to run Octave code on Linux-based embedded systems. MATLAB makes up for this by providing a dedicated compiler and embedded design tools that convert MATLAB functions down to C code and compile it into native machine language for a target machine. However, in compiling such a binary, program modifications and the advantages of scripting engine are traded off for faster execution time. The compilation of dedicated MATLAB code is therefore deemed too restrictive and complex for use on the μrover given the availability of the other languages mentioned here. Octave is usable under Linux on the μrover, but given the limited storage space and computation time available, the use of Python as a general scripting engine is preferred as Python is much more versatile and can handle mathematics well by using Numpy and SciPy.

Basic system programming in C with higher-level functions in C++ where necessary is considered to be the best choice of languages due to efficiency, ubiquity and portability. As Linux

and most other embedded or real-time operating systems (such as RTEMS) are implemented in C, this makes integration of the rover systems most convenient from an programming standpoint. In some cases, the extra overhead of C++ is justified, for example in the use of OpenCV functions on platforms that support high-level languages and operating systems. Python also has great potential for scripting and high-level functions that are easy to modify and integrate as bytecode, and will be considered for use in future functions such as real-time uploading of scripts for scheduled execution. Although Python requires a UNIX-platform or similar OS, it is still well-suited for applications that are not deeply embedded, require frequent modification, or must be ported between platforms that have a Python VM available. The prototype GUI for the μrover is programmed in Python for easy portability between operating systems and simplicity of modification when needed, as it does not require efficiency or close bindings to the host OS. Figure 2.43 shows a diagram of how the μrover software fits together in layers.

2.9 Embedded Optimization

Most embedded controllers and small processors, including the ARM9 and AVR microcontrollers used on the μrover, do not have built-in floating point units (FPU), and any floating-point operations are performed implicitly in software by code additions made by the compiler, in this case the GNU compilers (GCC). As this is very slow and often causes problems on some microcontrollers, such as the AVR, it is preferable to use a fixed-point mathematics implementation that can be tuned to the needs of the system. Integer arithmetic is not always a suitable solution for embedded mathematics, as an appropriate scaling factor must be used, and conversions to and from the scaled numbers may not be intuitive for mathematical operations. Using fixed-point math provides a decimal representation of numbers on integer calculation hardware applying direct addition with no changes, and simple bit shifting for multiplication. Fixed-point math implementations are used for acceleration in many gaming engines and other applications where efficiency takes priority over accuracy. The only drawbacks are that the range of representable numbers and hence the precision and range of the arithmetic are still limited by the bit width of the integer

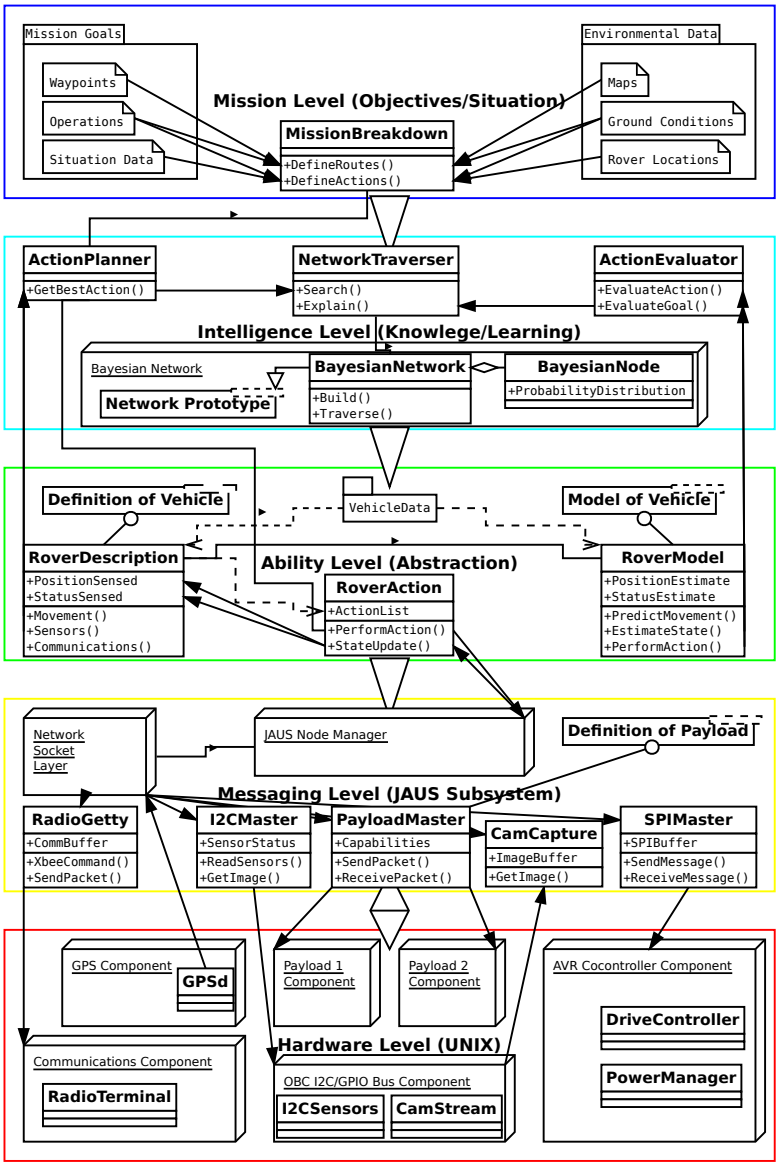


Figure 2.43: μrover Software Diagram

word used, and care must be taken to avoid overflows by using longer words for arithmetic where feasible [Lauha 2006]. Also, operators cannot be overloaded in C, so named function calls must be used for arithmetic operations on deeply embedded systems, but it is worthwhile to be able to perform fast floating-point calculations on such systems.

Fixed-point calculations split a binary word into two parts: the bits that represent the integer component of the number and that part that represents the fractional component, and typically the total number of bits corresponds to the natural word length on a given computing architecture. Figure 2.44 illustrates the composition of a fixed point number with 8 bits of integer and 8 bits of fractional, known as *8.8* fixed point format. The AVR is an 8-bit microcontroller, but contains instructions suitable for 16-bit operations as well with the compiler providing (less efficient) facilities for 32-bit arithmetic, and this fixed-point format suits it well. The closest approximation to the example of $32\pi = 100.53096491$ to 8 decimal places using *8.8* fixed point is 100.52734375, which is only accurate to one decimal place but in this case is within 99.5% of the actual value. The maximum value and error of a fixed-point number is determined by the number of bits allocated to the integer component and the fractional component, which do not have to be the same but usually are aligned on 8-bit boundaries for computational efficiency, 8 bits (a byte) being the smallest addressable unit in nearly all current computing architectures. An unsigned integer component with N_i bits can represent a maximum value of $2^{N_i} - 1$ in increments of 1, and a fractional component can represent by itself a fractional value of up to $(2^{N_f} - 1)/2^{N_f}$ in increments of $1/2^{N_f}$, and these two components are simply added together in terms of real number values. Hence the maximum bounded error of a fixed-point number with respect to the real number it is supposed to represent is $|1/2^{N_f}|$. To represent signed numbers, the highest-order bit in the integer component is used just as in conventional two's complement arithmetic. The maximum value for *8.8* fixed-point is 255.99609375 with a fractional increment of $1/2^N$. As most mathematical operations involve bit shifting, the number of bits can be changed fairly easily by changing the bit shift size, so the fixed-point implementation can be scaled depending on the word size w_p of the host processor, generally to $N_i = w_p - N_f$, and the precision required in each of the two components [Van Verth & Bishop 2008].

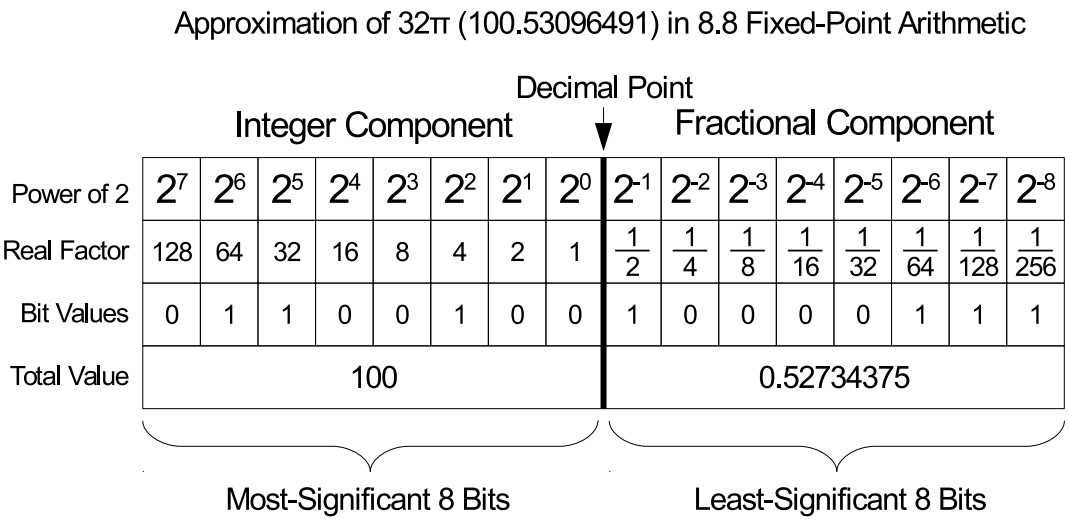


Figure 2.44: Structure of a Fixed-Point Number

The most common 32-bit fixed-point representation is referred to as *16.16*, which uses N_i = 16 bits for the integer part of the number and N_f = 16 bits for the fractional part in a 32-bit word, the most common word length for modern computing [Stalker & Boeren 1995]. The maximum value for *16.16* fixed point is 65535 + 65535/65536 = 65535.999984741211, which can provide general accuracy to 4 decimal places, and we use this format for most numerical processing on the ARM platform. Mathematical operations are implemented as follows [Street 2004]. For two fixed-point numbers a and b on which an operation is performed to yield a result c, addition and subtraction have the advantage that they can be done in the same way as integers.

$$c_{a+b} = a + b \tag{2.107}$$

$$c_{a-b} = a - b \tag{2.108}$$

$$\tag{2.109}$$

Due to the fixed decimal location, multiplication and division must be done with bit shifting

as well as multiplication, with $<<(N_f)$ denoting a left shift by N_f bits and $>>(N_f)$ denoting a right shift by N_f bits. To prevent overflows from occurring due to the multiplication, a and b must be temporarily cast to a longer integer data type, which is generally well supported by the compiler.

$$c_{a\times b} = (a \times b) >> (N_f) \tag{2.110}$$

$$c_{a\div b} = \frac{(a << (N_f))}{b} \tag{2.111}$$

A slightly faster method for inverting a number than the division in Equation 2.111 uses the fact that 1 in fixed-point format is created by $1 << (N_f)$.

$$c_{1\div a} = \frac{(1 << (2N_f))}{a} \tag{2.112}$$

The square root operation is performed iteratively using Turkowski's algorithm [Turkowski 1994], which is not reproduced here, but is a very popular method for performing square root operations in fixed-point representations. The atan2 function, used frequently for conversion from coordinates to absolute angle, is performed using an approximation by Shima [Shima 1999]. Sine and Cosine operations are performed by look-up tables precalculated on a floating-point machine and written automatically in integer format (so they can be read natively as integers) to a header file that is included in the C library. The fact that in virtually all common computer languages, fixed-point numbers are not natively supported within the compiler (Ada is the only known exception) makes conversion to and from natively-supported floating point formats for input and output important. Conversion to and from integer values is simply a matter of shifting the integer into the integer component of the fixed-point number and back such that $a_{int} = b_{fix} >> N_f$ and $b_{fix} = a_{int} << N_f$. Conversion to and from a floating-point number a_{float} to a fixed-point number b_{fix} must include adding 0.5 to round the number rather than truncating it, and is performed by

Table 2.2: Table of Constants in Decimal and Fixed-Point Format

Constant	Decimal	Fixed-Point
Zero	0.0	0
One	1.0	$1 << N_f$
π	3.14159	205887
2π	6.28318	411775
e	2.71828	178144
$\sqrt{2}$	1.41421	74804L
$\sqrt{3}$	1.73205	113512
φ	1.61803	106039

$$b_{fix} = \begin{cases} a_{float}((1 << N_f) + 0.5) & a_{float} \geq 0 \\ a_{float}((1 << N_f) - 0.5) & a_{float} < 0 \end{cases}$$

$$a_{float} = \frac{b_{fix}}{1 << N_f}. \tag{2.113}$$

In addition, several commonly-used constants are hard-coded as integer macros for use immediately as fixed-point values. Table 2.2 lists the constants available. This implementation is used in all floating-point arithmetic needed for the μrover, including Kalman filters and Bayesian inference, and the type "fix" is defined from "int32_t" on ARM and from "int16_t" on AVR. As it is much clearer to write and not significantly more complex to calculate matrix quantities in these algorithms rather than hand-calculating each element separately, vector and matrix handling functions were implemented in C using the fixed-point math functions for calculation. Vectors are stored as simple one-dimensional arrays of fixed-point values, and matrices are two-dimensional arrays. While it would make sense to define dedicated structures for vectors and matrices that includes important information such as the number of rows and columns, this would also lead to more complex C functions, less clarity in operation, and redundant storage of such information (for most matrix applications, sizes are well known in advance, and they must be consistent for most operations). Therefore all matrix functions also require the number of columns, and rows for matrices, as arguments as well as the pointer to the vectors/matrices to operate on. As

the software for the μrover is primarily written in C, C++, and Python, matrices are represented in row-major order, where the primary direction of data storage is in "horizontal" rows and the secondary direction is in "vertical" columns to maintain the array storage methodology of the language. Consequently, vectors are considered to be row vectors unless column vector multiplication in an operation is specifically called for.

Table 2.3 shows the list of functions for fixed-point math that are currently implemented in the μrover's library, where the "fix" data type is cast appropriately for the machine architecture as stated above. In all functions, output scalars are simply returned by the function and output vectors or matrices are the *first* pointer argument to the function. All functions have been implemented identically in native "double" floating-point format as well, and all μrover code can very easily be converted to use floating-point facilities simply by changing the header file that is included. Implementation details for the vector and matrix functions are not detailed here as matrix operations are very well known and most functions are simple iterative calculation routines. The exceptions are the functions for matrix inversion, which for speed and reliability are hard-coded formulas for square matrices of sizes 2, 3, and 4 obtained via modification of code generated from Maple.

Table 2.3: Fixed-Point Mathematics Functions Implemented for μrover

Mathematical Operation	Function Name	Arguments to Function
Multiply	fix fmult	fix n1, fix n2
Divide	fix fdiv	fix num, fix den
Square	fix fsquare	fix n
Invert	fix finv	fix n
Square Root	fix fsqrt	fix n
Exponent	fix fpow	fix n, int exp
Sine	fix fsin	fix angle
Cosine	fix fcos	fix angle
Tangent	fix ftan	fix angle
Create Vector	void fvmake	fix *vout, int length, elements...
Copy Vector	void fvcopy	fix *vout, fix *v1, int length
Vector Zero	void fvzero	fix *vout, int length
Vector One	void fvone	fix *vout, int length
Vector Scalar Addition	void fvscalaradd	fix *vout, fix *v1, fix s, int length
Vector Addition	void fvadd	fix *vout, fix *v1, fix *v2, int length
Vector Subtraction	void fvsub	fix *vout, fix *v1, fix *v2, int length
Vector Length	fix fvlength	fix *vout, int length
Vector Scalar Multiplication	void fvscale	fix *vout, fix s, int length
Vector Dot Product	fix fvdot	fix *v1, fix *v2, int length
Vector-Matrix Multiplication	void fvmult	fix *mout, fix *v1, fix *v2, int length
Create Matrix	void fmmake	fix *mout, int rows, int cols, elements...
Create Diagonal Matrix	void fmmakediag	fix *mout, int size, ...
Copy Matrix	void fmcopy	fix *mout, fix *m1, int rows, int cols
Matrix Zero	void fmzero	fix *mout, int rows, int cols
Matrix One	void fmone	fix *mout, int rows, int cols
Matrix Identity	void fmeye	fix *mout, int size
Matrix Diagonal	void fmdiag	fix *mout, fix *v1, int length
Vector to Diagonal Matrix	void fmdiagdiag	fix *mout, fix *m1, int length
Matrix Scalar Addition	void fmscalaradd	fix *mout, fix *m1, fix s, int rows, int cols
Matrix Addition	void fmadd	fix *mout, fix *m1, fix *m2, int rows, int cols
Matrix Subtraction	void fmsub	fix *mout, fix *m1, fix *m2, int rows, int cols
Matrix Transposition	void fmtrans	fix *mout, fix *m1, int rows, int cols
Matrix Scalar Multiplication	void fmscale	fix *mout, fix s, int rows, int cols
Matrix Vector Multiplication	void fvxform	fix *vout, fix *m1, fix *v2, int rows, int cols
Matrix Multiplication	void fmmult	fix *mout, fix *m1, fix *m2, int size
Matrix $\mathbf{M_1 M_2 M_1^T}$ Multiplication	void fmmultsq	fix *mout, fix *m1, fix *m2, int size
Matrix Square Root	void fmchol	fix *mout, fix *m1, int size

2.10 Communications

Reliable communication is a critical part of any unmanned system, and particularly so for space systems that cannot be reached physically in most cases. While the European Cooperation for Space Standardization (ECSS) has published a wide range of standards for space engineering, including detailed specifications for ground-to-space and space-to-space communication protocols, a full implementation of such a protocol was considered to be too complex for current μrover development, and consequently a customized protocol was developed for reliable end-to-end communications on small, embedded space platforms. A diagram of the situation assumed to be present for the μrover is shown in Figure 2.45, where a base station or lander forms a fixed reference point and one or more mobile rovers must remain in contact as much as possible. For message routing purposes, each network, vehicle, component, and device is assigned a separate address in comparable fashion to IPv4 addresses used for Internet routing. This hierarchical organization is used to provide a consistent protocol between networks, vehicles, and hardware so that no intermediate translation layers are needed that could potentially introduce extra complexity and faults. Implemented as a mesh network, it is possible for messages to be transferred not only between onboard components and adjacent vehicles, but also in multiple-hop communications along a string of vehicles, with each acting as a "relay" to extend range. Currently, only one network (addressed as 0 by default) is implemented, but connecting multiple networks in remote locations would be possible with additional routing and potentially Internet tunneling logic added.

2.10.1 Layered Model

In general, a communications system can be considered to be layered, such as in the Open Systems Interconnection (OSI) model for standardized communications, which visualizes any communications system (from the bottom up) in terms of Physical, Data Link, Network, Transport, Session, Presentation, and Application layers. It is proposed here that for an embedded, reliability-critical space hardware system, the following needs must be addressed:

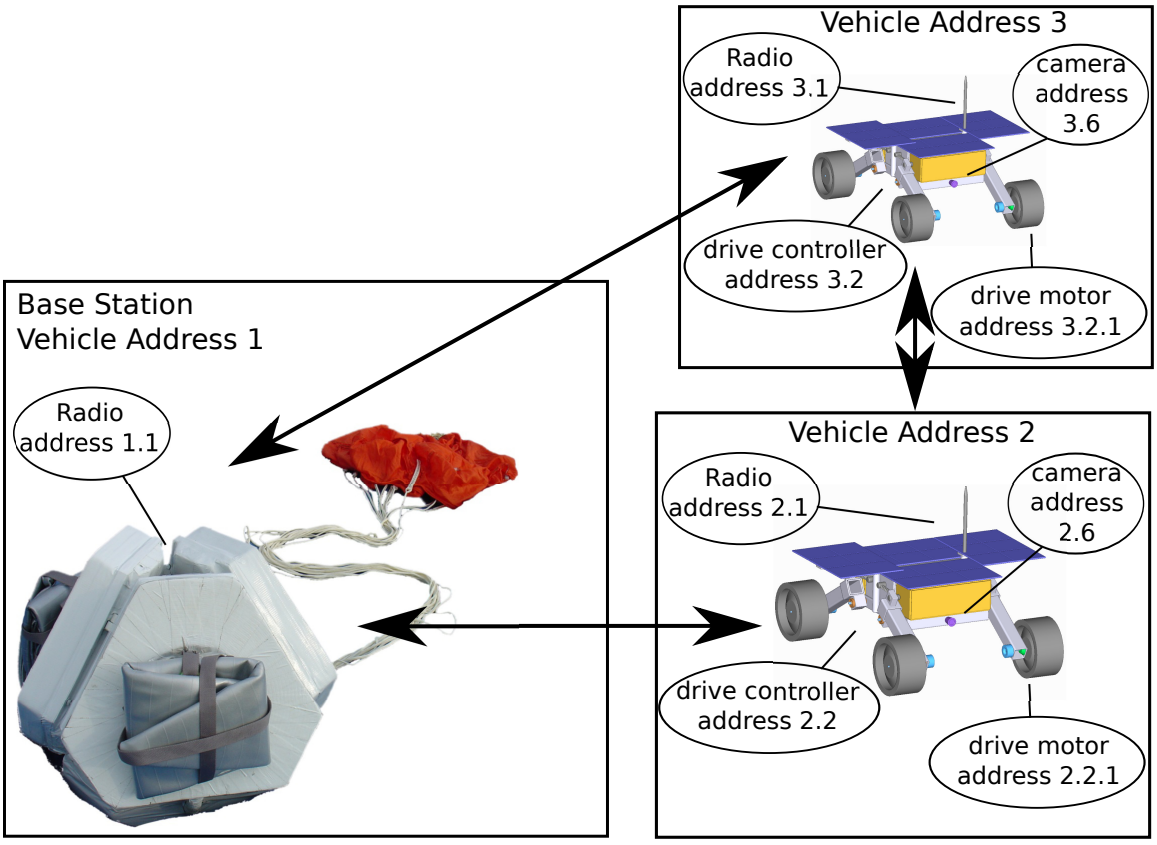


Figure 2.45: Diagram of μrovers and base station communication topology

1. Physical layers from simple byte-level serial on embedded controllers and upwards must be usable

2. The Data Link layer must provide CRC checking of data and automatic retransmission

3. Point-to-point Mesh Networking between both components and vehicles must be possible at the Network layer

4. The Transport layer must manage data payloads of arbitrary size and content, from single-byte signalling to video streaming.

In the current system, layers above the transport layer are effectively subsumed by the μrover API, as sessional contexts are only meaningful in terms of whether valid connections exist between rovers and components. Consideration here of OSI layers above 4 is not necessary, as the functions for session management and data formatting are included in the API for μrover programs and are also handled transparently by the XBee radios using the Digimesh protocol. Applications running on the μrover and base station simply call the necessary functions from the API with destination addresses to transmit data. A visualization of this layered system is shown in Figure 2.46. While the communications network formed is similar in many respects to an Ethernet network using TCP/IP, the implementation is much simpler and can be accomplished with little overhead and without an explicit networking stack and integrated kernel support. Security measures such as host validity checking and encryption are also not considered at the moment as such measures should be unnecessary for robotic exploration.

2.10.1.1 Physical Layer

For flexibility in application, the physical layer could be any communications channel that supports transmission and reception of bytes. TTL serial byte streams and SPI are used between internal electronics packages and RS-485 is used to external hardware and payloads. Typically, vehicles are connected by radio links, and components are implemented as either separate microcontrollers on board a vehicle or separate program processes running on the vehicle operating

#	OSI Layer	µrover Implementation
7	Application Layer	UNIX Processes using network sockets as interface
6	Presentation Layer	API functions for broadcast detection of networked devices with their attributes and addresses, and transmission of arbitrary data to known addresses. Connection is always present when communications channel is established.
5	Session Layer	
4	Transport Layer	
3	Network Layer	4-level hierarchical addressing with caching at each level
2	Data Link Layer	byte header framing with data byte stuffing & CRC check
1	Physical Layer	Sockets (s/w);TTL Serial (int.);RS-485 (ext.);Digimesh (radio)

Figure 2.46: Comparison of OSI model with μrover communication layers

system (OS) (although multiple "vehicle" on-board computers on a single hardware platform are also possible). Network sockets are used for interprocess communication between client programs running on an OS and the communications channel, with each device claiming a different port. The use of sockets, as in applications such as X11, has the additional benefit that when using Ethernet or other network communications channels, program communications is accomplished by simply connecting to a port on the target platform, which corresponds to a component that can contain several devices. For serial or radio communications, a driver program will manage the hardware port and simply manage transmission and reception of packets. Packets received by the driver program are passed on to the port and received by the connected hardware. Ports are mapped starting at port 30000 to ensure they do not interfere with other programs that may be running. The advantage to using a one to one mapping of ports and devices is simplicity of routing and debugging, which only requires a simple socket connection. However, this places a heavy responsibility on the communications API that is initiating the connection, so it is preferable that the task of connection management is distributed over more than one component. Currently, the radio driver initiates connections for the passing of control packets to and from each component

in the μrover, but the API is structured such that other programs could separately interconnect if needed.

2.10.1.2 Data Link Layer

Reliably passing data over communications channels that may or may not be reliable requires a number of measures to ensure that as much information as possible is accurate and that errors and miscommunications cause as little damage as possible. The data link layer is structured to provide a built-in, transparent process of stateful caching and intelligent retransmission when necessary. The following measures are used to ensure reliable transmission.

1. To minimize bandwidth use and eliminate redundant commands, each state variable is time-tagged, and only updated when a more recent and changed state value is sent and received.

2. To ensure that command packets are not lost in transmission and subsequently ignored as redundant, each client sends a confirmation to the host whenever a variable is updated, or else the command is retransmitted after an interval. These confirmations need only be requested or sent if the variable has not been modified for a while, so that rapid bursts of communication activity do not cause unnecessary overhead.

3. Read-only variables such as those on sensors also are tagged as requested when the user requests a sensor reading, and a time tag prevents overly-frequent requests.

4. Hard-coded initial values and limiting values for each state variable are used to prevent unintended movements or telemetry on power up and out-of-range values.

Each client maintains each variable, flag, and value for its own devices, and a unique identification value is used to identify each device when regular keep-alives are sent. A fundamental problem with data links of this type is that when framing data packets, packet control values must be separate in some sense from the data carried in the packet. Sending bytes in a specific frame order is not a reliable solution, as bits or whole bytes can be lost or scrambled in transmission, changing the framing synchronization with random results. Byte stuffing such as

that used in Serial Line IP (SLIP), Point-to-Point Protocol (PPP), and the AX.25 packet radio protocol provides a solution by converting reserved data values into longer sequences that do not contain these values. The more recent Consistent Overhead Byte Stuffing (COBS) algorithm guarantees that the extra overhead presented by these sequences is no more than one byte in 254. [Cheshire & Baker 1999].

The data link layer of all communications uses COBS with a CRC32 checksum at the end of each packet, and the packet structure shown in Figure 2.47. The purpose of this structure is to provide a standard 8-byte (64-bit) data frame structure delimited on byte boundaries to avoid bit-shifting overhead that is easy to parse but flexible enough to allow both simple commands and long data packets to be sent efficiently. Fundamentally, a Packet Header control code is used to ensure that each packet is distinguishable and that asynchronous framing is reliable. The control code used for this is $0x80$, which is not frequently seen in data as it represents -128 in signed decimal while the most common 8-bit values used are in the range of -127 to 127, and is not as common as the limiting values $0x00$, $0xFF$, or the common ASCII/UTF-8 codes between $0x00$ and $0x7F$. A 7-bit Flags byte is reserved for for transfer control information, but is not used at present. Two bytes are used to store the size information, leading to a 2^{16} ($16KiB$) maximum data size, larger than the 1500-byte MTU maximum used for Ethernet and sufficiently large for most buffered asynchronous systems. The data payload, containing the contents from the network layer and transport layer follow the packet header. This ensures that only the flags and size information are not protected as a packet will not be recognized with a corrupt packet header byte, and if the size information is wrong, the checksum will fail anyway.

The limitation by using COBS is that the flags cannot be equal to the packet header byte (or it may be recognized as COBS data for $0x80$), so to avoid this, the MSB of the packet flags is always defined to be 0 (Flags must be less than 128 in unsigned representation, or greater than 0 in signed representation) and the packet is considered bad if this MSB is 1. To maintain this reliability, all additional control codes and specifically packet headers to be added in future are recommended to have value greater than 128, which also keeps them outside the ASCII range.

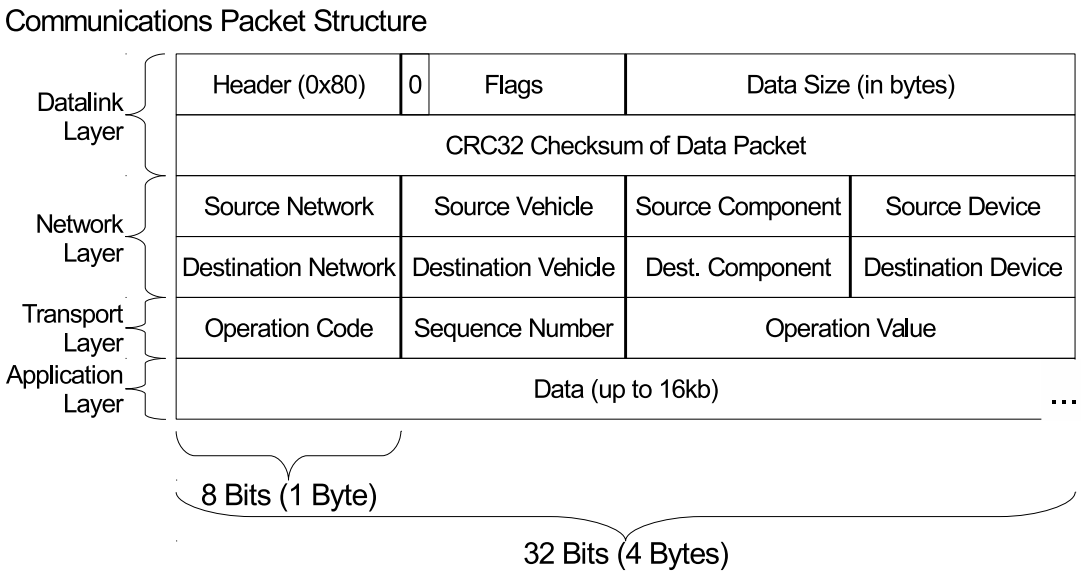


Figure 2.47: Structure of a communications packet

2.10.1.3 Network Layer

The network layer is based on the idea that any part of the network of μrovers should be accessible from any other part. For this purpose, a hierarchy of addresses is used to locate each hardware device within a component, located on a vehicle within a network. The network address bytes within a communications frame are shown in Figure 2.47, and the only practical limit on what information could be used as an address is the width of the address field in each data frame, currently set to 8 bits but easily changed if necessary. To allow devices to respond to packets, the source address of the packet must also be included in the packet structure. A device requested to send a response will send a packet with the destination address set to the source address of the requesting packet.

To make use of the mesh networking capabilities of the XBee radios, static maps of the 8-bit vehicle addresses to the 64-bit XBee hardware addresses are defined in a header file for use in all software communications. This mapping is shown in Table 2.4. Components on a vehicle are numbered starting at 1, but as ports are mapped starting at 30000, the first device is available

Table 2.4: XBee hardware address mapping table

Radio Location	Vehicle Address	XBee Address High Byte	XBee Address Low Byte
Base Station	1	0013A200	4030EA6D
μrover	2	0013A200	40538A11
μrover Prototype	3	0013A200	4030E9F3

Table 2.5: Component port address mapping table

Component Name	Address Numbers	Attached Devices	Device Numbers
Process Control	1	Start/Stop Control	0
Packet Radio	2	Character Transceiver	0
Mobility Control	3	Wheel Motor Drives	0,1,2,3
I^2C Sensors	8	Accel, Mag, Temp, Gyro	29,30,72,104
Range Sensors	10	3 Front and 3 Back Sensors	0,1,2,3,4,5
Camera	16	Image Frame	0

on port 30001. A list of currently-used μrover components is shown in Table 2.5. Devices are also enumerated starting with 1 and respond to packets that are addressed to them. Device addresses are typically inherited from the logical structure of the device itself, for example an I^2C controller enumerates device addresses to correspond to the 7-bit I^2C addresses of attached devices. At every level of addressing, the address 0 is reserved for broadcasting, in particular for device detection. A vehicle address of 0 indicates a packet will be sent to all vehicles on the network, a component address of 0 indicates a packet will be sent to all components on each addressed vehicle, and a device address of 0 indicates a packet will be sent to all devices on each addressed component. Zeros for all four address levels indicate a general "broadcast" packet that is used for identifying all connected vehicles, components, and devices on all networks by using an operation code of $0x01$ to request an identification response.

2.10.1.4 Transport Layer

The transport layer for communications currently only needs to include information on the type of operation being performed. An 8-bit operation code, or opcode, is used to define this. A list of currently implemented opcodes is given in Table 2.6. Note that opcodes are organized by the bits that are set in their opcodes to aid in decoding. The low-level command set is designed to occupy

the range of $0x00$ to $0x7F$ to avoid using byte stuffing starting at $0x80$. The basic operation code is "Request Identification", which is multicast to all devices in the network or on a vehicle to identify all attached devices and their capabilities. As transfer of large amounts of data via multiple packets is needed in some cases (e.g. for video streaming or image transfer) an 8-bit sequence number is used to ensure that multiple packets arrive in the correct order. Packets are cached internally until the preceding sequence number packet is received before data is dispatched to the client program. This sequence number resets to zero after 255 packets have been sent, under the assumption that even a continuous video stream will not experience a packet arrival delay of more than 254 packets. To further increase the awareness and reliability of the transport layer, an acknowledgement packet is sent to acknowledge the successful reception of each command that does not involve returning data. For example, a set actuator position command will elicit an acknowledgement, but a get actuator position command will cause the device to return a set actuator position packet with the current position enclosed. An acknowledgement packet with data field zero, essentially an acknowledgement to an acknowledgement, is considered to be a bad packet (one possibility is a packet of all zeros) and is ignored.

The transport protocol is a descendant of the simple protocol used for real-time control of some previous YURT rover prototypes, known as "4-byte" protocol from its packet size, and has the main advantages of simple parsing and small packet size. In this version, the sequence number replaces the device number used in the "4-byte" protocol as addressing information has been moved to the network layer. The reason for retaining the functionality of this protocol at the transport level is that for simple operations that must be performed in real time with a minimum of networking overhead, it is desirable to be able to send only the operation code and a 16-bit data word without the need for additional variable-length data provided at the application layer. Examples of this include writing to an individual device register, setting a motor speed directly, or reading from a 16-bit ADC, all of which often must be performed in real time on small devices such as 8-bit microcontrollers that do not have space for caching or processing large packets. For these cases, a 20-byte packet that includes all layers of communication processing up to the transport layer can be used that does not include variable-length data following the 16-bit

Table 2.6: List of currently implemented operation codes

Operation Name	Opcode	Associated 16-bit Data	Additional Data
Acknowledgement	$0x00$	Opcode & Seqnum	None
Query Identification	$0x01$	None	None
Get Identification	$0x02$	Device Type	Capability List
Query Status	$0x03$	None	None
Get Status	$0x04$	Status Flags	None
Set Actuator Position	$0x11$	Actuator Position	None
Set Actuator Speed	$0x12$	Actuator Speed	None
Set Actuator Scaling	$0x13$	Actuator Scaling	None
Set Parameterized Position	$0x18$	Control Mapping	None
Get Sensor Value	$0x21$	Sensor Value	None
Get Vector of Sensor Values	$0x28$	Value Type	Value List
Get Actuator Position	$0x31$	None	None
Get Actuator Speed	$0x32$	None	None
Get Actuator Scaling	$0x33$	None	None
Get Parameterized Position	$0x38$	Control Mapping	None
Capture Still Image	$0x41$	Image Parameters	None
Query Still Image	$0x42$	Image Format	Image Data
Start Video Stream	$0x44$	Video Format	Image Data
Stop Video Stream	$0x45$	None	Image Data
Set Position Control Parameters	$0x51$	Controller Type	Parameter List
Set Speed Control Parameters	$0x52$	Controller Type	Parameter List
Get Position Control Parameters	$0x61$	Controller Type	Parameter List
Get Speed Control Parameters	$0x62$	Controller Type	Parameter List
Go To Global Position	$0x71$	Position Format	Position Coords.
Get Global Position	$0x72$	Position Format	Position Coords.

transport data field. Higher level commands such as sending a whole file, setting navigation waypoints, or grabbing an image frame from a camera make use of the variable-length data field and multiple-packet transmission capabilites on larger devices, as packets are simply dropped and not processed if sent to a small device that cannot handle the specific packet opcode. The command set is designed to be flexible, and is still under development for new systems.

2.10.1.5 Application Layer and Formats

The programs that use the communications system call the provided API functions directly. These functions transparently provide the port connection and network routing logic that is needed to

ensure that the appropriate component or next routing node is reached properly. As only minimal contextual data is explicitly enumerated within the communications protocol, the software must be aware of the context that messages are sent and received in. For example, image formats and their corresponding data codes for transfer of camera data must be pre-established between programs that need to communicate. This indicates that for general data storage and communication, a very flexible common language should be used that can encapsulate any data that needs to be communicated. A knowledge-based system must have a specific language format for describing stored knowledge. It is desirable that this language be:

- Able to store or reference any kind of data
- Well-defined, to eliminate ambiguity in meaning
- Extendable for all reasonable future use
- Human-readable, to facilitate understanding and debugging
- Common in format, so that existing tools may be used for parsing

To fill these requirements, Extensible Markup Language (XML) is used as a common medium for storing and communicating knowledge between μrovers. There is sufficient flexibility in the XML format to allow commands, mission data, machine learning, and hardware descriptions to be stored and transmitted in common ASCII text as well, so XML will form the basic information storage system for the μrovers. XML is used to store Bayesian network information and transmit it between rovers as well.

The complete communications system is both reliable and efficient for control of intelligent mobile robots over communications links of uncertain quality. A schematic of the flow of data through the system is shown in Figure 2.48.

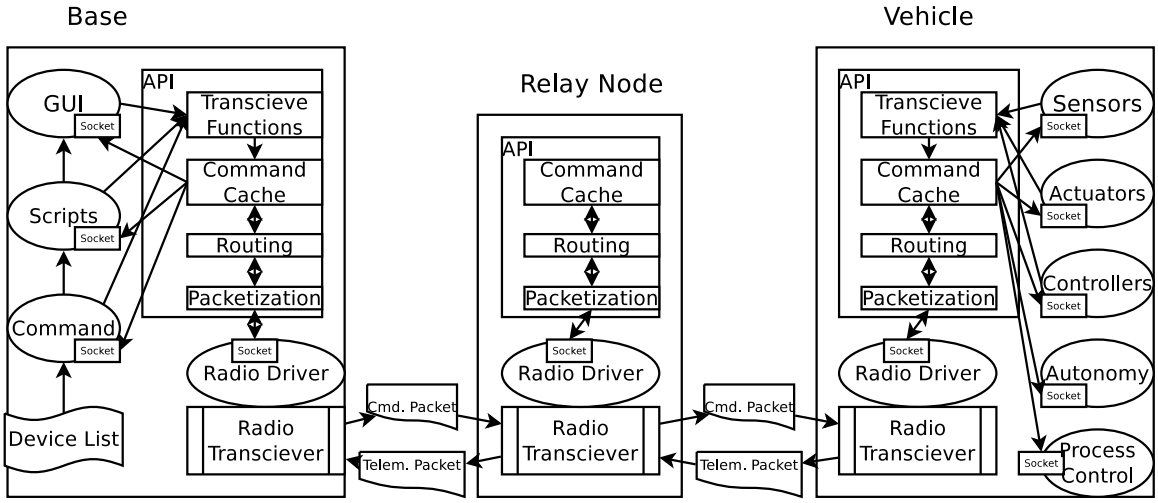


Figure 2.48: Schematic of μrover communications system

2.11 Predictive Filtering

The Unscented Kalman filter has been often used for parameter estimation of inertial measurement systems because of its ability to handle nonlinear systems very well [Bae & Kim 2010]. The formulation for parameter estimation with a UKF includes the computation of weights, the establishment of sigma points, the prediction of the mean and covariance of both states and measurements, the prediction of cross covariance, the gain calculation, and the state update step. However, the computational load is a limiting factor, which is proportional to the number of sigma points. There are three common sets of sigma points used in the literature with N state variables: the symmetric set ($2N + 1$ points), the reduced set ($N + 1$ points), and the spherical set ($N + 2$ points). In a recent publication [Menegaz *et al.* 2011], a new minimum sigma set with equal size to the reduced set but better high-order performance is proposed and applied to localization and map building.

The sensor suite on the μrover uses a MEMS accelerometer and gyroscopic MEMS rate sensors for rotational feedback which experience slow angular drift over time. An Unscented Kalman Filter is used to compensate for drift and sensor noise in the system and increase the accuracy of inertial estimation. We also apply an adaptive algorithm to improve the estimate of the noise covariance based on the residual [Li *et al.* 2013d]. In addition, we can apply a new sigma set which uses $N + 1$ sigma points rather than the traditional $2N + 1$ to the adaptive unscented Kalman filter [Post *et al.* 2012a]. We compare our results to the new Cubature Kalman Filter (CKF) for the same application.

2.11.1 Unscented Kalman Filter

The Unscented Kalman filter (UKF) is a proven method for highly non-linear inertial estimation systems that experience noise and large disturbances. A state vector $\mathbf{x}$ is the numerical quantitity to be estimated, and changes at each time step t. The statistics of $\mathbf{x}$ are known and assumed to be Gaussian, with mean $\bar{\mathbf{x}}$ and covariance matrix $\mathbf{P}$. Several distinctions are important: The covariance matrix $\mathbf{P}$ is not to be confused with the probability distribution over a random variable

P(X), and is further qualified as an autocovariance of a single variable x, written $\mathbf{P_{xx}}$, or a cross-covariance of two variables x and y, $\mathbf{P_{xy}}$. As a symmetric autocovariance matrix, the diagonal consists of the variances $\mathrm{E}((\mathbf{x} - \bar{\mathbf{x}})^2)$ of the state variables $\mathbf{x}$. Also, the mean vector $\bar{\mathbf{x}}$ and covariance matrix $\mathbf{P}$ do not refer to noise as the noise covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ do. Rather, they statistically refer to the expected *changes in state* of the system over time. For example, in the case of tracking a vehicle's heading, a vehicle that is expected to make an equal number of left and right turns would have a mean heading of 0, and if irregular fast turns are expected, a larger covariance will be estimated than if regular slow turns are expected.

The filter assumes a Markov model such that each successive state $\mathbf{x}(t+1)$ is determined entirely by the previous state $\mathbf{x}(t)$, a control input $\mathbf{u}(t)$, and additive Gaussian noise η that is characterized by known mean $\bar{q}$ and covariance matrix $\mathbf{Q}$. Since it represents a real value with an additive random quantity, the size N state vector can be considered to be an N-dimensional continuous random variable with Gaussian statistics, and will be treated as one for purposes of probabilistic algebra. The formulation for the UKF proceeds as follows. Consider the discrete nonlinear system at time step t

$$\mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t), t) + \eta_{\mathbf{s}}(t) \tag{2.114}$$

$$\mathbf{y}(t+1) = g(\mathbf{x}(t+1), \mathbf{u}(t+1), t+1) + \eta_{\mathbf{m}}(t+1). \tag{2.115}$$

For this system, $\mathbf{x}(t) \in \mathbb{R}_S$ is the size n state vector, that contains all the states of the system at time t. The measurement vector $\mathbf{y}(t) \in \mathbb{R}_M$ provides the sensor measurement at time step t, $\mathbf{u}(t)$ is the control input at time t, $\eta_s(t)$ represents white noise with a random value at time t, a mean of $\bar{q}$ and covariance $\mathbf{Q}$, and $\eta_{\mathbf{o}}(t)$ represents white noise with a random value at time t, a mean of $\bar{\mathbf{r}}$ and covariance $\mathbf{R}$. The conventional UKF [Xiong *et al.* 2006] is based on the determination of $2N + 1$ sigma points (oddly denoted as χ to distinguish them from the standard deviation σ as they are not necessarily located at one standard deviation from the mean), in a configuration commonly known as the *symmetric set*. The zeroth sigma point χ_0 is obtained from the mean of the state vector, while the rest of the sigma points are obtained by shifting each state variable

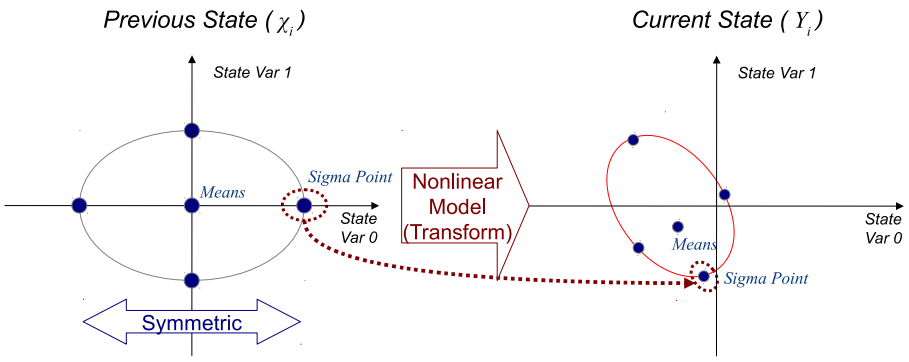


Figure 2.49: The Unscented Transform, Traditional UKF

in turn positive and negative by the scale factor $\sqrt{(N+\kappa)\,\mathbf{P}_i(t+1)}$, creating a symmetric set of $2N$ points surrounding the mean χ_0 in N-dimensional space. The constant κ controls the spread of sigma points, and is usually set to 0. This process for the Unscented Transform is illustrated in Figure 2.49, and starts with the selection of *sigma points* that represent the prior probability distributions of the state variables from

$$\chi_0(t) = \mathbf{x}(t) \tag{2.116}$$

$$\chi_i(t) = \mathbf{x}(t) + \sqrt{(N+\kappa)\,\mathbf{P}_i(t)} \qquad \forall i = 1, 2, \ldots, N \tag{2.117}$$

$$\chi_{i+N}(t) = \mathbf{x}(t) - \sqrt{(N+\kappa)\,\mathbf{P}_i(t)} \qquad \forall i = 1, 2, \ldots, N \tag{2.118}$$

where i represents the choice of state variable in $\mathbf{x}$, and N is the number of states. The first $i = 1 \ldots N$ sigma points are shifted positive in state space one variable at a time, the second $i = N{+}1 \ldots 2N{+}1$ sigma points are shifted negative. To calculate the square root of the covariance matrix, a Cholesky decomposition is used as it provides more robustness against singularities than alternative matrix square-root methods. The *model prediction* of statistics for the next time step $\mathbf{x}(t+1)$ follows. Each sigma point is run through the nonlinear system model f with the addition of system noise q implicitly assumed to yield a set of transformed points that follow the statistics of the nonlinear system. The nonlinear system model includes the control inputs applied to the real system u to predict the effect of actuator control on the system state, and the system

noise q generally is associated with uncertainty in the system actuation. The system noise mean $\bar{q}$ also is generally assumed be zero for white noise. This essentially attempts to predict what the probability distributions will look like after the next time step for comparison to measured information.

$$\chi_i(t+1) = f(\chi_i(t), u(t)) \tag{2.119}$$

The weightings used for statistical estimation from the sigma points naturally must incorporate information about the distribution of the sigma points themselves. Weightings Wm_i are applied to the sigma points to obtain the mean $\bar{\mathbf{x}}$, and weightings Wc_i are applied to the autocovariance to obtain $\mathbf{P_{xx}}$. These weightings are calculated as

$$Wm_i = \frac{\kappa}{N+\kappa} \tag{2.120}$$

$$Wc_i = \frac{1}{2\,(N+\kappa)}. \tag{2.121}$$

The transformed values are then utilized to reconstruct the posterior probability distributions for the state variables. The state mean $\bar{\mathbf{x}}$ is predicted as a simple weighted sum of sigma points, and the system covariance $\mathbf{P_{xx}}$ with system noise covariance matrix $\mathbf{Q}$ is estimated using this mean and the transformed sigma points. This mean vector can be considered to be the most likely system state based only on the system model and actuators, while the covariance provides a measure of how uncertain this state is.

$$\bar{\mathbf{x}}(t+1) = \sum_{i=0}^{2N} Wm_i \chi_i(t+1) \tag{2.122}$$

$$\mathbf{P_{xx}}(t+1) = \sum_{i=0}^{2N} Wc_i \left(\chi_i(t+1) - \bar{\mathbf{x}}(t+1)\right)\left(\chi_i(t+1) - \bar{\mathbf{x}}(t+1)\right)^T + \mathbf{Q} \tag{2.123}$$

We can now perform the *measurement prediction* for the next time step in the same way that we predicted the model. The model-transformed sigma points $\chi_i(t+1)$ are again transformed by a nonlinear sensor measurement model g again with the addition of measurement noise r implicitly

assumed. The measurement model is an estimate of the most likely measured state from the sensors given the updated state from the system model.

$$\tilde{\chi}_i(t+1) = g(\chi_i(t+1), \mathbf{u}(t+1)) \tag{2.124}$$

The predicted observation vector from the sensors $y(t+1)$ and its predicted autocovariance $\mathbf{P_{yy}}$ with sensor noise covariance matrix R are defined in the same way as $\bar{\mathbf{x}}$ and $\mathbf{P_{xx}}$, as

$$\bar{\mathbf{y}}(t+1) = \sum_{i=0}^{2N} Wm_i \tilde{\chi}_i(t+1) + \mathbf{R} \tag{2.125}$$

$$\mathbf{P_{yy}}(t+1) = \sum_{i=0}^{2N} Wc_i \left(\tilde{\chi}_i(t+1) - \bar{\mathbf{y}}(t+1)\right)\left(\tilde{\chi}_i(t+1) - \bar{\mathbf{y}}(t+1)\right)^T + \mathbf{R}. \tag{2.126}$$

To compare the statistical properties of the estimated state and estimated observation, a cross-covariance $\mathbf{P_{xy}}$ is calculated from the sigma points for both state and measurement, using the covariance weightings.

$$\mathbf{P_{xy}}(t+1) = \sum_{i=0}^{2N} Wc_i \left(\chi_i(t+1) - \bar{\mathbf{x}}\right)\left(\tilde{\chi}_i(t+1) - \bar{\mathbf{y}}(t+1)\right)^T \tag{2.127}$$

Using the covariances from the state and measurement estimates, the Kalman gain can be obtained. The purpose of the Kalman gain is to control the "innovation" on the posterior state, or simply, to control how much of an effect the sensor measurement has on the state. Although the system model is not linear, the covariances of the posterior distributions $\mathbf{P_{yy}}$ and $\mathbf{P_{xy}}$ can still be linearly related by

$$\mathbf{K}\mathbf{P_{yy}}(t+1) = \mathbf{P_{xy}}(t+1) \tag{2.128}$$

where $\mathbf{K}$ is the Kalman gain. Inverting $\mathbf{P_{yy}}$ allows the matrix $\mathbf{K}$ to be determined, but this matrix inversion is a significant cause of instability and inefficiency in Kalman filtering algorithms. For this reason, closed-form solutions for $N \times N$ matrices of size up to $N = 4$ have been implemented, and $\mathbf{P_{yy}}$ is checked to ensure the determinant is nonzero to prevent singularities. The calculation

for Kalman gain is then

$$\mathbf{K} = \mathbf{P_{xy}}\mathbf{P_{yy}}^{-1}. \tag{2.129}$$

Finally, the Kalman gain is then applied to the residual, which is the difference $(\mathbf{y}(t+1) - \bar{\mathbf{y}}(t+1))$ between the current actual sensor measurement vector $\mathbf{y}$ and the estimated sensor measurement $\bar{\mathbf{y}}$, and is added to the propagated state mean $\bar{\mathbf{x}}(t+1)$ to obtain $\mathbf{x}(t\tilde{+}1)$, the current estimate of the system state which is assumed to be the correct system state in this iteration and propagated to the next iteration as $\mathbf{x}(t)$.

$$\tilde{\mathbf{x}}(t+1) = \mathbf{x}(t+1) + \mathbf{K}\left(\mathbf{y}(t+1) - \bar{\mathbf{y}}(t+1)\right) \tag{2.130}$$

The system covariance is also updated based on the Kalman gain $\mathbf{K}$ and posterior sensor autocovariance $\mathbf{P_{yy}}$ from this iteration, which allows the filter to make better estimates based on prior statistics.

$$\tilde{\mathbf{P}}(t+1) = \mathbf{P}(t+1) - \mathbf{K}\mathbf{P_{yy}}\mathbf{K}^T \tag{2.131}$$

2.11.2 Adaptive Unscented Kalman Filter

In this way, the Kalman filter algorithm adapts system covariance based on the uncertainty in the sensor measurements. But the noise covariances $\mathbf{Q}$ and $mathbfR$ normally do not change over time, although the noise itself may. Sensor noise in particular is subject to change based on the environment. So if the prior statistics of the noise are not known or change over time, an adaptive algorithm can be used to adjust the noise covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ as shown in Figure 2.50. In this study, the statistical estimator is based on work by Mohamed and Schwarz [A. H. Mohamed 1999], which is applied as follows. An estimate of the sensor innovation covariance $\mathbf{S}$ is obtained by averaging the residual from the UKF over a window of length N as

$$\mathbf{S}(t+1) = \frac{1}{N}\sum_{j=k-N+1}^{k}\left(\mathbf{y}(t+1) - \bar{\mathbf{y}}(t+1)\right)\left(\mathbf{y}(t+1) - \bar{\mathbf{y}}(t+1)\right)^T. \tag{2.132}$$

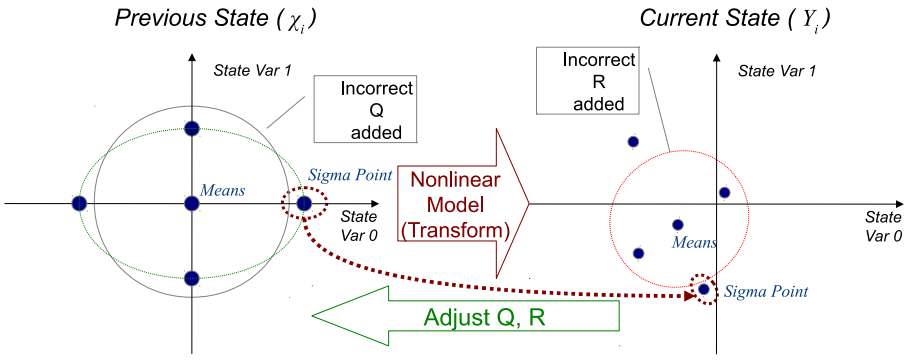


Figure 2.50: The Unscented Transform, Adaptive UKF

Then based on the whiteness of the filter innovation sequence represented by $\mathbf{S}$, the statistical matrices can be updated at the end of each cycle by the method of Pourtakdoust and Ghanbarpour [Pourtakdoust *et al.* 2007].

$$\hat{\mathbf{R}}(t+1) = \mathbf{S}(t+1) + \mathbf{P}(t+1) - (\mathbf{K}(t+1)\mathbf{P}_{\mathbf{yy}}\mathbf{K}(t+1)^T \tag{2.133}$$

$$\hat{\mathbf{Q}}(t+1) = \mathbf{K}(t+1)\mathbf{S}(t+1)\mathbf{K}(t+1)^T \tag{2.134}$$

It should be noted that adaptation of both $\mathbf{Q}$ and $\mathbf{R}$ simultaneously is not advisable, as the noise covariances build on each other and instability can result. Additionally, a measure of fault detection can be provided based on whether the actual residual from the UKF exceeds the theoretical residual. Using the method of Soken and Hajiyev, a metric for fault detection ς can be calculated from [Soken & Hajiyev 2009]

$$\varsigma = (\mathbf{y}(t+1) - \bar{\mathbf{y}}(t+1))^T \left(\mathbf{P}_{\mathbf{yy}}(t+1) + \mathbf{R}\right)^{-1} (\mathbf{y}(t+1) - \bar{\mathbf{y}}(t+1)) \,. \tag{2.135}$$

If this metric exceeds a set threshold ($\varsigma > 5$ is used in testing), it can be assumed that a fault in the sensing or actuator hardware has occurred. This can be used to scale up the system covariance matrix and scale down the Kalman gain so that innovations from faulty assumptions can be minimized, increasing robustness. The effects of the adaptive UKF can be visualized as shown in Figure 2.50.

2.11.3 Reduced Sigma Point Kalman Filter

The unscented Kalman filter's computational load is largely dependent on the number of sigma points, as propagation through the nonlinear model and mean and covariance calculations comprise a significant fraction of the total processing time. We attempt to offset this load by applying an asymmetric sigma point set of size $N + 1$ using statistical estimation that is based on work by Menegaz et al. [Menegaz *et al.* 2011], and modify the sigma set as shown in Figure 2.51. First, the constant w_0 and matrix $\mathbf{W}$ are defined as

$$Wm_0 = \frac{\kappa}{N + \kappa} \tag{2.136}$$

and

$$\mathbf{W} = \sqrt{I_{N\times N} - \frac{1 - w_0}{n} * 1_{N\times N}} \tag{2.137}$$

where $0 < W_0 < 1$, $I_{N\times N}$ is the $N\times N$ identity matrix, and $1_{N\times N}$ is the $N\times N$ matrix with all entries set to 1. Using these as constants, the reduced-set sigma points are obtained in a similar fashion to the UKF, replacing Equations 2.116, 2.117, and 2.118 by the asymmetric sigma point set

$$\chi_0 = \mathbf{x}(t) - \frac{\sqrt{(N + \kappa)\mathbf{P}_i(t + 1)}}{\sqrt{Wm_0}} \left[\sqrt{\frac{1 - Wm_0}{n}} \right]_{N\times 1} \tag{2.138}$$

and

$$\chi_i(t) = \mathbf{x}(t) + \sqrt{(N + \kappa)\,\mathbf{P}_i(t + 1)} * \mathbf{W} * \mathbf{ISW}_i, \qquad \forall i = 1, 2, \ldots, N \tag{2.139}$$

where $\mathbf{ISW}_i$ is the Cholesky decomposition of $diag(Wm_0 * \frac{1-Wm_0}{n}\mathbf{W}^{-1} * 1_n * (\mathbf{W}^T)^{-1})$. Using the same structure as the original UKF, Equations 2.125, 2.126 and 2.127 have to be rewritten with the new defined weights as

$$\bar{\mathbf{y}}(t + 1) = \sum_{i=0}^{n} Wm_i \tilde{\chi}_i(t + 1) + R(t + 1), \tag{2.140}$$

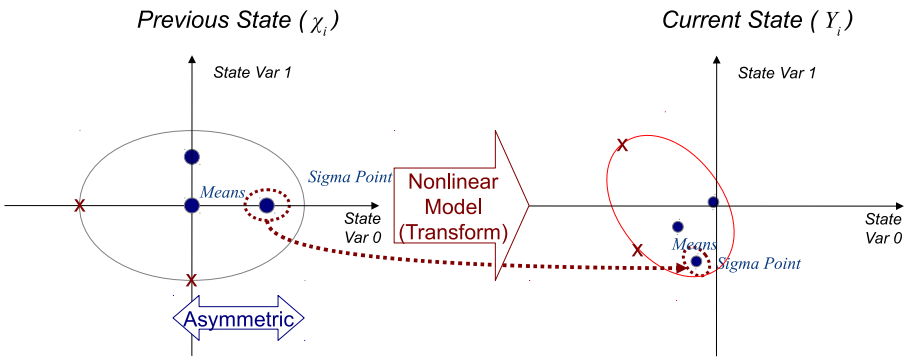


Figure 2.51: The Unscented Transform, Reduced Sigma Set UKF

$$\mathbf{P}_{yy} = \sum_{i=0}^{n} \bar{Wm}_i \left(\tilde{\chi}_i(t+1) - \bar{\mathbf{y}}(t+1)\right) \left(\tilde{\chi}_i(t+1) - \bar{\mathbf{y}}(t+1)\right)^T, \tag{2.141}$$

and

$$\mathbf{P}_{xy} = \sum_{i=0}^{n} \bar{Wm}_i \left(\chi_i(t) - \bar{\mathbf{x}}(t+1)\right) \left(\tilde{\chi}_i(t+1) - \bar{\mathbf{y}}(t+1)\right)^T \tag{2.142}$$

where the weightings $\bar{W}_i$ are calculated as the diagonal values of the matrix equation

$$\begin{bmatrix} \bar{Wm}_1 & \dots & \sqrt{\bar{Wm}_1}\sqrt{\bar{Wm}_N} \\ \dots & \dots & \dots \\ \sqrt{\bar{Wm}_1}\sqrt{\bar{Wm}_N} & \dots & \bar{Wm}_N \end{bmatrix} = Wm_0 * \frac{1 - Wm_0}{N} * \mathbf{W}^{-1} * 1_N * (\mathbf{W}^T)^{-1}. \tag{2.143}$$

The caveat for using a smaller sigma set is that instability is much more likely for system models that do not have symmetric statistics. The assumptions implicit in the use of one sigma point include that the mean and standard deviation are consistent enough for two points to define a Gaussian distribution. Hence, for highly irregular systems, the UKF or CKF are better choices. The reduced-set UKF is illustrated graphically in Figure 2.51.

2.11.4 Cubature Kalman Filter

The recently-developed Cubature Kalman Filter (CKF) is a sigma-point filter that uses a spherical-radial cubature rule for numerical integration to achieve estimation performance very close to a true Bayesian estimate [Arasaratnam & Haykin 2009]. Unlike an Unscented Kalman Filter (UKF), only $2N$ sigma points are needed rather than $2N + 1$, and there is no ambiguity regarding where to place the sigma points (which is a point of common contention among sigma-point filters). Like a square-root UKF, Cholesky updating of covariance matrices can be used to avoid matrix square-root operations on each iteration for higher numerical efficiency. The sigma points (or cubature points) and weightings used for numerical integration have no central mean point, and replace Equations 2.116, 2.117, and 2.118 with the symmetric distribution of points

$$\chi_i(t) = \mathbf{x}(t) + \sqrt{N\mathbf{P}_i(t)} \qquad \forall i = 1, 2, \ldots, N \tag{2.144}$$

$$\chi_{i+N}(t) = \mathbf{x}(t) - \sqrt{N\mathbf{P}_i(t)} \qquad \forall i = 1, 2, \ldots, N. \tag{2.145}$$

Although the derivation of the filter is significantly different, the main functional difference reduces to a sigma set of $2N$ points scaled only by N and no other parameters. The CKF uses the same numerical process as the UKF, but with a total of $2N$ sigma points used to calculate the means and covariances instead of $2N + 1$, and the use of the cubature rule replaces Equations 2.123, 2.126, and 2.127 with

$$\mathbf{P}_{\mathbf{xx}}(t+1) = \frac{1}{2N}\sum_{i=1}^{2N}\chi_i(t+1)\chi_i(t+1)^T - \bar{\mathbf{x}}(t+1)\bar{\mathbf{x}}(t+1)^T + \mathbf{Q} \tag{2.146}$$

$$\mathbf{P}_{\mathbf{yy}}(t+1) = \frac{1}{2N}\sum_{i=1}^{2N}\tilde{\chi}_i(t+1)\tilde{\chi}_i(t+1)^T - \bar{\mathbf{y}}(t+1)\bar{\mathbf{y}}(t+1)^T + \mathbf{R} \tag{2.147}$$

and

$$\mathbf{P}_{\mathbf{xy}}(t+1) = \frac{1}{2N}\sum_{i=1}^{2N}\chi_i(t+1)\tilde{\chi}_i(t+1)^T - \bar{\mathbf{x}}(t+1)\bar{\mathbf{y}}(t+1)^T. \tag{2.148}$$

The main advantages of the CKF are that the sigma set uses fewer parameters and is simpler to implement, and that the distribution approximation is closer to an ideal Bayesian filter, and is therefore better at handling highly nonlinear systems. Adaptive rules for the CKF have not been proven, and as there is no sigma point for the mean of the system, a reduced-set version of the CKF is impossible at present. However, the CKF shows great promise as a practical improvement over the basic UKF.

2.12 RTOS Survey for Flight Hardware

The Emdebian Linux OS on the ARM architecture has been used for most of the existing development work, and many other FOSS systems can be used as well. Customized Linux kernel patches and drivers that implement user-space support for the embedded OBC peripherals such as the RS-485 interfaces, SPI controllers, and I^2C sensors have been developed to provide hardware integration, and a set of dedicated libraries for 8-bit AVR microcontrollers has been created to assist in programming component applications for this system. The software for onboard systems has been developed entirely in C and C++ for efficiency and minimal code size. The operating system and software is stored in NAND flash memory on the component microcontrollers and on redundant NAND and NOR flash devices on the motherboard. NAND flash is the most common and provides large storage capacity, while NOR flash devices can store only a few megabytes but may be more resistant to radiation and corruption [Farokh Irom 2008].

While embedded Linux provides a convenient and powerful platform for robotic and spacecraft software and hardware development, it is ultimately considered neither real-time, nor highly-reliable. Software in aerospace and reliability-critical applications is generally chosen on the basis of real-time response, task timing repeatability, and architecture reliability, and the complexity of the basic Linux kernel provides excellent scheduling mechanisms, but still may make it a poor contender compared to dedicated embedded real-time operating system (RTOS) platforms [Brown & Martin 2010]. Two strategies for the operation of space hardware are being considered: first, the use of a dedicated open-source RTOS which is not Linux-compatible (but may be POSIX compatible), and second, the use of a real-time Linux patch that provides real-time task support and an additional layer of reliability to the kernel.

2.12.1 RTOS Candidates

Summaries of the RTOS candidates under consideration are as follows.

2.12.1.1 RTEMS

RTEMS is a mature RTOS originally designed for use in missiles with standards such as POSIX and μITRON, and is used in the Electra SDR on the Mars Reconnaissance Orbiter. It is technically a single-process multi-thread OS and does not provide (or need) multiprocess memory management. It is free and completely open source under a modified GPL2 license that allows linking with proprietary applications, with the exception of the embedded web server. Its primary strengths are that it has high development activity with many projects based on it, and supports most Linux tools for compiling and running programs due to POSIX compliance. It also performs well in benchmarks due to its simplicity, and includes a BSD network stack and some trace and debugging functionality built in, although there is no full-featured profiler. Due to POSIX real time compliance, its ability to run SSH and low level ROS functionality is likely, and the GDB utility can be used for debugging. However, the selection of drivers, ready-made applications, and BSPs available, as with most free projects, is limited.

2.12.1.2 FreeRTOS

FreeRTOS is a popular RTOS for microcontrollers and small projects and has flown on small satellites such as AAUSAT3. It is free and open-source, slightly smaller than RTEMS and also distributed under a modified GPL, although due to the number of third-party extensions available, it is somewhat more licensing-restricted as a whole. The main strength of FreeRTOS is that it has a large number of ports and board support packages available. It is well supported in the community. However, its small size restricts compatibility with complex applications. FreeRTOS does not support the POSIX API in general, though the IO extension uses a small subset of POSIX for file access. As a result, it has little compatibility with UNIX applications such as SSH and ROS. It is also not SMP-capable without running parallel instances of the RTOS with interprocess communication. GDB and OpenOCD can be used, but little performance monitoring is available without the proprietary FreeRTOS+Trace utility.

2.12.1.3 eCos

The eCos RTOS was originally developed by Cygnus, before being abandoned and given as free and open-source software to the community. It is quite mature, but development has not been as active as before, and many of the BSPs are for old platforms, with few up-to-date packages. It does have SMP capability and provides a subset of the POSIX and μITRON APIs for real-time use as well as BSD networking capabilities, but the network stack is out of date. There are also no recent applications such as ROS and Player, and SSH support is unlikely. It remains very customizable though, and with development work could support a platform well. No built-in trace and performance monitoring tools are available and there appear to be no current tools for monitoring, although virtual vectors are available within the RTOS for tracing and GDB can be used for debugging over serial.

2.12.1.4 uC/OS-II

The uC/OS-II RTOS is developed and maintained by Micrium, and while being open-source and free for non-commercial use, is technically proprietary. Consequently, many of the tools are for-cost, including trace and debugging tools, including μC/Probe and Lauterbach Trace32. The OS itself and packages to add networking and filesystem functionality are relatively cheap and come with source code. Only the filesystem component includes POSIX API calls. Dropbear will run on the OS, but there is no indication that ROS or Player would run without significant porting effort.

2.12.1.5 QNX Neutrino

QNX Neutrino is a commercial RTOS used in many applications including the Neptec LCS Camera, and has recently been acquired by Blackberry for use in personal mobile devices. It is free for non-commercial use and some components are also open-source (although no source has been seen yet), but any other use requires licensing. It is SMP capable and POSIX real time compliant and includes a network stack and full UNIX userspace environment. The size and

complexity of the OS is comparable to a small Linux distribution, and consequently functionality is good, but real-time performance is comparable to a RT-Linux kernel, although hard real-time support is better. The industrial value of QNX is largely in the number of certifications it has achieved, including IEC 62304 for medical devices, ISO/IEC 15408 EAL4 and the Safe Kernel to IEC 61508 SIL 3. QNX devices have also been military certified to DO-178B. QNX has been used in many robotics applications, and will run SSH, Player, and probably ROS with minimal porting effort. The only significant problems are the cost of support, and the fact that performance is likely to be comparable to a RT-Linux kernel due to complexity.

Tables 2.7 and 2.8 show a comparison of some of the important factors that were considered in choosing a dedicated RTOS for academic and space use for freely available and commercially sold RTOS candidates respectively. Due to its high level of functionality and POSIX compatibility, as well as efficiency and current availability as fully open-source, RTEMS was determined to be the most appropriate RTOS for use on the μrover at present.

Table 2.7: Table of Freely-Available RTOS Candidates

RTOS:	RTEMS	FreeRTOS	eCos	uC/OS-II	QNX Neutrino
Criteria					
Vendor	OAR	Real Time Engineers Ltd.	Cygnus (originally)	Micrium	QNX
Price	Free	Free	Free	$80, Free for edu. and NC	Free for non-com. only
Licensing type	GPL2+open linking	Modified GPL (noncomm.)	GPL-Compatible eCos	Open-source proprietary	Limited OSS some parts
Kernel/Rel. Version	v4.11	7.4.0	v3	OS-III now available	6.5.1
Real time perf. mon.	monitor,context,stack,usage	FreeRTOS+Trace ($395/3mo)	None built in	μC/Probe ($1000)	QNX Momentics Tool Suite
Interactive debugger	GDB (symbolic)	GDB/OpenOCD/Eclipse	GDB over serial/ethernet	Lauterbach Trace32	IDE uses GDB
Strace or truss	capture engine	FreeRTOS+Trace ($395/3mo)	virtual vector monitoring	KA Plug-In for IAR's C-SPY	System Profiler
Community support	http://wiki.rtems.org/	Forum on SourceForge	Mailing Lists/Bugzilla	Commercial Support	Commercial Support
C/C++ toolchain	Yes, gcc	Yes, gcc	Yes, gcc	Yes, gcc	Yes, gcc and Dinkumware
POSIX compliance	POSIX 1003.13-2003 P52	No, FreeRTOS+IO "POSIX-like"	Compatibility Layer	Only partly in μC/FS	
SMP capable	Yes	No, runs AMP (RTOS/Linux)	Yes, but limited	Yes, Call	Yes
Supp. ARMv7	Yes, EABI version 5	Yes	Yes	Yes	Yes
Supp. NEON/VFP	Yes, gcc compiler support	Yes, gcc compiler support	Yes, gcc compiler	Yes, gcc compiler	Yes, gcc compiler support
Can run JAVA	Kaffe port, maybe GNU GCJ	Maybe, no packaged solution	Possibly Wonka or Kaffe	Some work done	Commercial(J9, JamaicaVM)
SSH, ROS, etc.	SSH Likely, ROS maybe	No (requires POSIX)	Dropbear Yes, ROS No	Dropbear Yes, ROS No	SSH/Player yes, ROS likely
Network stack	Yes, Based on FreeBSD	FreeRTOS+UDP only	Yes, Based on FreeBSD	μC/TCP-IP ($80)	Yes
Used in space apps.	Yes (Electra SDR on MRO)	Yes on smallsats (AAUSAT3)	No known missions	Ground testing only	Yes (Neptec LCS Camera)
Video processing	OpenCV build required	OpenCV build required	OpenCV build required	Unlikely to run OpenCV	Has been done previously
Numerical processing	Yes	Yes	Yes	Yes	Yes
Interrupt servicing	Yes	Probably	Yes	Probably	Yes
Communications	With driver port	If drivers available	With driver port	If drivers available	Yes, with BSP
File system and bus	With driver port	If drivers available	With driver port	If drivers available	Yes, with BSP
GNU Toolchain	GNU RTEMS toolchain	GNU toolchain	GNU toolchain	GNU toolchain	GNU toolchain
Gentoo Toolchain	Yes, arm-rtems-gcc	Yes, arm-elf-gcc	Yes, arm-elf-gcc	Yes, arm-elf-gcc	Yes, arm-elf-gcc

Table 2.8: Table of Commercially Sold RTOS Candidates

RTOS:	RTX	VxWorks	Integrity	Nucleus	LynxOS
Criteria					
Vendor	Keil (ARM)	Wind River	Green Hills	Mentor Graphics	LynuxWorks
Price	Free, tools limited to 32k code	Lic. $17k, Src. $120k	Royalty Free, $?	RTOS $12495	$7k, or MSRP based
Licensing type	BSD (OS only, tools are extra)	Closed-source	Closed-source	Closed-source	Closed-source
Kernel/Rel. Version					4.x
Real time perf. mon.	None built in	None built in, third party	MULTI Profiler ($6000)	Nucleus Trace	SpyKer Pro ($?)
Interactive debugger	MDK-ARM uVision4 ($10000)	Wind River Workbench ($?)	TimeMachine ($17900)	EDGE $2995/seat	Luminosity IDE ($?)
Strace or truss.	MDK-ARM uVision4 ($10000)	Wind River Workbench ($?)	SuperTrace (hardware)	Nucleus Trace	SpyKer Pro ($?)
Community support	Commercial Support	Commercial Support	Commercial Support	Commercial Support	Commercial Support
C/C++ toolchain	Yes, armcc compiler	Tornado VxWorks IDE (gcc)	MULTI C/C++ or gcc	EDGE C++	Yes, gcc
POSIX compliance	No	POSIX 1003.13-2003 PSE52	POSIX 1003.13-2003	by POSIX API	Yes + Linux ABI
SMP capable	Yes	Yes	Yes	Yes	Yes
Supp. ARMv7	Yes	Yes	Yes	Yes	Yes
Supp. NEON/VFP	Yes	Yes	Yes	Yes	Yes
Can run JAVA	Oracle Java ME Embedded	Jworks Jeode JVM	IS2T RT MicroJVM, etc.	NewMonics PERC JVM	Aphelion Java Toolkit
SSH, ROS, etc.	None verified to run	SSH Yes, ROS/player maybe	SSH Yes, ROS/player maybe	None verified to run	Likely, given ABI
Network stack	TCP/IP networking suite	Yes, Based on FreeBSD	GHNet TCP/IP Stack	Nucleus NET	Yes, Based on FreeBSD
Used in space apps.	Minimal (AAU-Cubesat)	Many, MRO, MER, MSL, etc.	Iridium, GPS III satellites	None found	None found
Video processing	Unlikely to run OpenCV	OpenCV build required	Has been done previously	OpenCV build required	OpenCV build required
Numerical processing	Yes	Yes	Yes	Yes	Yes
Interrupt servicing	Probably	Yes	Yes	Yes	Yes
Communications	With driver port	Yes, with BSP	Yes, with BSP	Yes, with BSP	If drivers available
File system and bus	With driver port	If drivers available	Yes, with BSP	Yes, with BSP	If drivers available
GNU Toolchain	Proprietary Toolchain	GNU toolchain	MULTI toolchain	Proprietary Toolchain	GNU toolchain
Gentoo Toolchain	No	No	No	No	No

2.12.2 RT-Linux Candidates

Summaries for the RT-Linux candidates under consideration are as follows.

2.12.2.1 The PREEMPT_RT Patch

There are fewer contenders for an RT-Linux platform. The simplest way to provide real-time capabilities to Linux is to apply the official PREEMPT_RT patch (documented at rt.wiki.kernel.org), which consists of thousands of lines of patch code designed to add real-time preemption capabilities to the kernel. The patch makes almost every part of the kernel preemptable by using high-resolution timing and allowing a process with real-time priority to take control at reliable intervals for arbitrary amounts of time. While this does greatly improve the responsiveness of high-priority processes, it does so at the cost of kernel performance for other processes. The PREEMPT kernel option already provides kernel preemption that often reduces latencies into the millisecond range at the (potential) expense of kernel throughput, and PREEMPT_RT is required for latencies less than that. In both cases, very little modification of userspace programs is needed to gain the benefits of near-realtime performance (as the kernel scheduler handles the hard work), but it is not considered a "hard" real-time kernel in the embedded sense, as the performance of real-time (or simply high-priority) processes is not guaranteed at the kernel level. Performance is very much dependent on how well programmed the pertainant drivers and interrupt handlers are, and jitter specifically is noticeably higher than in "harder" real-time systems.

2.12.2.2 ChronOS

ChronOS is a Linux distribution from Virginia Tech that is designed to provide a Linux kernel testbed for real-time scheduling and resource management research on multicore platforms. The main difference is that the ChronOS Linux kernel scheduler has been extended to support additional single-processor and multiprocessor scheduling algorithms to improve multi-thread performance. ChronOS also provides a middleware API for explicit scheduling of tasks, using methods such as partitioned scheduling, global scheduling, etc. over multiple processors. Al-

though the extra functionality allows much greater flexibility and potentially higher performance than the basic PREEMPT_RT patch, the underlying kernel mechanisms are essentially the same, and the degree of real-time reliability is still less than dedicated real-time kernels.

2.12.2.3 RTLinux

The original RTLinux was developed as a commercial product at FSMLabs, and then acquired by Wind River Systems. The RTLinux architecture is important chiefly because it is a true real-time kernel that runs the traditional Linux kernel as a fully preemptable sub-process, allowing processes to run with noticeably lower latency and jitter compared to PREEMPT_RT kernels. However, it was officially abandoned in 2011, and operated only on the 2.4 series kernels at that time. While it provides full real time POSIX compatibility and an aerospace-qualified release known as FlightLinux, the age of the supported kernels and the lack of any official support or updates makes use of RTLinux itself impractical, but both RTAI and Xenomai are based on the original RTLinux kernel and are very popular and modern systems.

2.12.2.4 RTAI

RTAI (Real Time Application Interface) is the first of two forks of the original RTLinux code, focuses on maximizing performance, and is free and open source. Like RTLinux, it is a real-time kernel that runs Linux as a separate non-realtime task. To free the code from proprietary FSMLabs code, a hardware abstraction layer called ADEOS (Adaptive Domain Environment for Operating Systems) was developed to act as a hypervisor between the real-time code and userspace. ADEOS is interesting because it can be inserted at runtime as a kernel module in the Linux operating system to allow interfacing with the real-time kernel while remaining separate from it. RTAI also implements most of the POSIX RT functionality for hard-real time processes. RTAI has slightly better interrupt latency performance than Xenomai due to higher isolation of the real-time kernel using ADEOS, where interrupts are passed directly to the operating system domain handling them (such as the RT kernel) rather than all going through the standard Linux interrupt handling mechanisms. However, this makes porting of the RTAI code more difficult, and

RTAI supports fewer platforms and Linux versions than Xenomai (though ARM is included), and the development community is noticeably smaller.

2.12.2.5 Xenomai

Xenomai is the second fork of the original RTLinux code, focuses on maximizing portability and maintainability, and is also free and open source as well as having a larger following than RTAI. It also uses the real-time kernel approach, but focuses on providing pervasive, interface-agnostic, hard real-time capability seamlessly integrated into the GNU/Linux environment. To make the RTOS as universally compatible as possible, a hypervisor is used based on ADEOS/I-pipe, but Xenomai only exports a set of generic RTOS services on top of which RTOS personalities (called "skins") are built. POSIX, uITRON, VRTX, VxWorks, and pSOS+ skins are available in addition to the native Xenomai API. The DENX ELDK uses Xenomai, and has a space heritage in TacSat-2. As the focus is on maintainability and usability, interrupts are not dealt with directly in the real-time kernel as in RTAI, but performance is not significantly affected, and the system still operates as hard real time. Also, Xenomai is intended to support and co-operate with the PREEMPT_RT patch in future so that the userspace API and skins can run on either a dual-kernel or single-kernel system without any noticeable changes in interface. This makes Xenomai likely to be well-supported and ubiquitous in the future.

Table 2.9 shows a comparison of RT-Linux candidates for academic and space use. As it is the most actively maintained and current distribution of RT-Linux with the widest level of compatibility, Xenomai was determined to be the most appropriate RT-Linux candidate for use on the μrover.

Table 2.9: Table of RT-Linux Candidates

Linux/RT:	PREEMPT_RT Patch	ChronOS	RTLinux	RTAI/Adeos	Xenomai
Criteria	(preemption level)	(preemption level)	(RT kernel)	(RT kernel)	(RT kernel)
Licensing type	GPL2	GPL2	GPL/Paid, Similar to QT	GPL/LGPL2	LGPL
Kernel/Release Version	3.4-rt	3.0.24	Last was about 2.4.19	3.9.1	3.5.3
Real time perf. mon.	Yes, toolset	Yes, toolset	Yes, toolset	Yes, toolset	Yes, toolset
Interactive debugger	Yes, GDB	Yes, GDB	Yes, GDB	Yes, GDB	Yes, GDB
Strace or truss	strace	strace	strace	strace	strace
Community support	Yes	Wiki Only	No Longer Supported	Mailing List	Mailing List
C/C++ toolchain	Yes	Yes	Yes	Yes	Yes
POSIX compliance	Yes, Non-RT	Yes, Non-RT	POSIX 1003.13	Subset of POSIX RT	POSIX RT skin
SMP capable	Yes	As Linux	Yes	Yes	Yes
Supports ARMv7	Yes	Yes	Yes	Yes	Yes
Supports NEON/VFP	In Kernel	In Kernel	In Kernel	In Kernel	In Kernel
Can run JAVA	Yes	Yes	Yes	Yes	Yes
SSH, dropbear, ROS, etc.	Yes	Yes	Yes	Yes	Yes
Network stack	Yes	Yes	Yes	Yes	Yes
Used in space apps.	In various forms	None Found	Yes, as FlightLinux	None Found	Yes (TacSat-2)
Video processing	Yes	Yes	Yes	Yes	Yes
Numerical processing	Yes	Yes	Yes	Yes	Yes
Interrupt servicing	Yes	Yes	Yes	Yes	Yes
Communications	Yes	Yes	Yes	Yes	Yes
File system and bus	Yes	Yes	Yes	Yes	Yes
GNU Toolchain	Yes	Yes	Yes	Yes	Yes
Gentoo Toolchain	Yes	Yes	Yes	Yes	Yes

If the complexity of Linux itself is not required, then to minimize cost and maximize flexibility and performance RTEMS would be the most appropriate candidate, as it provides POSIX real time support, SMP capability, and a good feature set. A secondary possibility would be eCos, although it is not as well-maintained. FreeRTOS, while popular for many small-scale projects, lacks important features such as a POSIX API and SMP support. QNX, while popular and reliable, is not known to be significantly better in robotics applications than an RT-Linux variant. The use of an RT-Linux would be suitable as long as complexity, size, and reliability are acceptable, as all RT-Linux systems can be expected to interoperate well with basic Linux applications and support at least close to the full range of POSIX and UNIX functions. While use of the basic PREEMPT_RT patch works with almost any Linux kernel and requires little programming effort, real-time performance can be expected to be correspondingly low in comparison to a hard RTOS, which may be of less value overall. While ChronOS provides a good range of programmability, it operates more as a research platform than a hard real-time OS and is not generally used in real-time applications as a result. To overcome these limitations, it makes more sense to use an RT-Linux with a dedicated real-time kernel (like RTLinux). However, RTLinux is outdated and unmaintained, making it a bad choice. This leaves RTAI and Xenomai, both of which would make good choices for comparison with an RTOS. The tradeoff is effectively that RTAI may provide on the order of 10% better latency and jitter performance, while Xenomai may be longer lived, better supported, and easier to port to other applications. Given that other platforms and applications will undoubtably be used in the future, it makes more sense to choose Xenomai as an RT-Linux candidate, as the performance difference is relatively small and so that greater benefit and better interoperability can be achieved in the future.

2.12.3 Real-Time Software Testing

There are many benchmarks for computer systems, including the Whetstone (floating-point), Dhrystone (integer), Dhampstone (combined float and int), Hartstone (scheduling), HINT (problem-independent), and Rhealstone (timing) metrics, and benchmarking suites such as Mibench and Thread-Metric. Rhealstone metrics are specifically used for RT systems, and in-

corporate the six operations of task switch time, preemption time, interrupt latency, semaphore-shuffle time, deadlock-break time and intertask message latency [Pantano & Timmerman 2012]. In general, the most critical for external control and high-speed interfacing are latency and jitter, representing the total time and time variation observed when responding to interrupts or performing time-sensitive processing. Total task time and task-switch time are also useful comparative metrics between operating systems of different complexities. For use in space systems, it is important that an operating system provide both fast response in terms of interrupt latency and task completion time, and and consistent timing of real-time tasks. To obtain an idea of the relative performance of real-time operating systems, we will focus on measurement of latency, jitter and task time.

2.12.3.1 Latency

An event in a RTOS could be an interrupt request by hardware or a task scheduled by the operating system. The hardware interrupt latency is the time elapsed from the interrupt request generation to the execution of the first line of ISR code. System interrupt latency is the time elapsed from the generation of the signal task to the execution of the first instruction of the task. Interrupt processing time is the combined time to get the ISR, perform the ISR, and go through the scheduler and determine which thread now should run and then enable that thread to run, restoring its context if necessary. This provides a measure of how fast a platform can respond to events. The less time taken for operations that need to occur between the initiation of the interrupt and the actual execution determines the latency. Platforms with more efficient or timely preemption structures will generally have less latency.

2.12.3.2 Jitter

Jitter is a measure of how much the execution timing of a task differs over subsequent iterations. Real-time operating systems are optimized at low levels to minimize jitter. This provides a measure of the consistency of a platform's timing. Platforms with better real-time task management and "harder" task deadlines will have lower jitter.

2.12.3.3 Task Switch/Task Run Time

Task switching overhead refers to the time that the RTOS takes to change the execution context from one thread to a higher priority thread, which preempts the executing thread. Task run time refers to the total time required to complete a particular task, including preemption and latency. Hardware interrupts will be measured as a task switch, while processing and communications metrics will be measured as a run time. This provides a relative measure of the execution efficiency of each platform. Platforms with fewer internal operations to perform due to the task will take less time.

In addition, the process of monitoring a real-time system can be divided into the software, hardware, and hybrid approaches, summarized as follows [El Shobaki 2001]:

2.12.3.4 Software Monitoring Systems

With software monitoring, instrumentation code is run on the target system directly. Triggering is accomplished by executing the inserted code, and the generated events are stored in an execution history buffer in target system memory or on a separate system. An advantage of software monitoring is the flexibility and the applicability to many systems. The drawbacks of instrumentation are that it utilizes target resources such as memory space and execution time from the CPUs. Also, when the instrumentation is removed, the system may change behaviour due to timing differences. This problem, commonly referred to as the probe effect, causes many timing and synchronization related errors in multiprocess real-time systems. However, if only a single process is measured at a time, the possibility of timing errors is minimal.

2.12.3.5 Hardware Monitoring Systems

Here, the monitoring system uses only hardware to monitor the system. A hardware device is connected to buses of the target system to passively detect and collect the events of interest. Hardware monitoring is typically found in conjunction with in-circuit emulators and logic analyzers. The advantage of hardware techniques is that they do not interfere with execution of

the target system. The disadvantages are higher cost, complexity, and dependency on a specific hardware architecture. Moreover, it is challenging to obtain good signal quality at today's system speeds.

2.12.3.6 Hybrid Monitoring

Hybrid monitoring uses a combination of both software and hardware monitoring and is typically used to reduce the impact of software instrumentation alone. We apply this approach to the timing analysis of the software on the μrover by reading an external counter from a software monitoring framework.

Chapter 3

Autonomy

For any vehicle to operate independently of human control, it is necessary to have some degree of autonomous operating ability. This generally includes capabilities such as basic navigation, environmental awareness, fault detection, automated planning, and problem solving. While most traditional autonomous robots use either pre-coded state machines and fuzzy systems based on expert knowledge or online learning systems such as neural networks, the autonomy systems on the μrover are based on probabilistic theory, which allows them to handle uncertainties and inconsistencies to a greater degree. This chapter will introduce the probabilistic systems and programming frameworks used for navigation, decision-making, and learning as well as the systems implemented for vision and environmental awareness. The key novel technologies involve the use of embedded fixed-point Bayesian networks for probabilistic robot programming, decision-making, and behavioural control with the added capability of probabilistic learning and updating of knowledge, as well as the use of optical flow and structure from motion techniques together and the application of structure-from-motion to sequential images for monocular SLAM.

3.1 Probabilistic Systems

The fundamental concept of a *probability* of a certain *event* happening includes several assumptions. A probability p is generally a real positive number between zero and 1 (denoting 100%) that is used to assign quantitative measures of confidence to a given *result* or set of results of that event. To define a result, we first define a *result space* of all possible results Ω, which can be viewed as a discrete set of measurable outcomes of the event. For example, in the classic example of rolling a die, the event is the roll, and the result space is $\Omega = \{1, 2, 3, 4, 5, 6\}$. We also define a set of measurable events, or occurrences O to which we are willing to assign probabilities. A specific event $o \in O$ is also a subset of the result space Ω, such that each desired result can be assigned a probability, but all possible results and combinations need not be considered. This also implies that the empty event 0 (no event at all) and the trivial event Ω (all possible events at once) can be part of O. Also, O is closed under union ($\because o_1, o_2 \in O \therefore o_1 \cup o_2 \in O$) and complementation ($\because o \in O \therefore \Omega - o \in O$), implying that it is also closed under the other Boolean operations as well [Koller & Friedman 2009].

3.1.1 Probability Distributions

These principles are formalized in the concept of a *probability distribution* P, defined over the spaces Ω and S that maps from the events $o \in O$ to real probability values p. In numerical terms, this means that for a countable number of events $o_1 \dots o_N$, the distribution P must at least provide N mappings $p_1 \dots p_N$. Each mapping is described using functional notation by $p = \mathrm{P}(o)$, where the rule of positive probabilities states that

$$\forall o \in O,\ \mathrm{P}(o) \geq 0. \tag{3.1}$$

For P to be a complete probability distribution, it must also satisfy the rule of total probability across the trivial event spanning all of Ω as

$$\mathrm{P}(\Omega) = 1. \tag{3.2}$$

Finally, probability distributions must obey the sum rule, which is stated as

$$\forall(o_1, o_2 | o_1 \cap o_2 = 0), \mathrm{P}(o_1 \cup o_2) = \mathrm{P}(o_1) + \mathrm{P}(o_2) \tag{3.3}$$

where $o_1 \cup o_2$ is the union of events o_1 and o_2. In simple terms, these three rules simply state that probabilities must be all positive, the total probability of all possible results is 1 (because every possible result *must* be in the complete space Ω), and the probability of two completely unrelated (or "mutually disjoint") results is just the sum of the two probabilities. For the example of rolling a die where all results are assumed to be independent, this means that $\mathrm{P}(1 \cup 2) = \mathrm{P}(1) + \mathrm{P}(2)$, and $\mathrm{P}(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6) = 1$, which makes intuitive sense.

3.1.2 Random Variables

While individual probabilities of events are suitable for simple stochastic systems with Boolean events, most real systems are defined by events or outcomes with multiple possible values, and even continuous result distributions. Although multiple events could be defined, one for each possible outcome of a distribution in Ω, this would unnecessarily complicate the math and actually limit the kinds of distributions that could be modelled. To describe these kinds of multivalued systems and indeed any real-valued system, in the place of certain events we use *random variables*. A random variable is a functional that assigns a value to each outcome $o \in \Omega$ that gives meaning to that outcome, and for the purpose of practical implementation on computers, we will focus on *discrete* random variables with countable distributions. We denote random variables with an uppercase letter such as X, while a specific value that the random variable can take is denoted in lowercase such as x like any other scalar. Hence, the statement $X = x$ refers to a specific value x within the variable X, or expressed mathematically [Koller & Friedman 2009]

$$\{o \in \Omega : f_X(o) = x\} \tag{3.4}$$

where f_X just refers to the mapping that the random variable X performs on values from Ω. The complete set of values associated with a random variable can be obtained using the value operator

$\mathrm{V}(X)$. In practical terms, while a probability distribution over an event $\mathrm{P}(o)$ is a simple probability value, a probability distribution $\mathrm{P}(X)$ over a random variable is a vector of probabilities called a *multinomial* corresponding to the possible values that the variable can take. A random variable can take the place of a simple Boolean event by taking only two values $\mathrm{V}(X) = \{true, false\}$, and the distribution of such a random variable is known as a Bernoulli distribution. Random variables can also be continuous functions, where any real-numbered value x is essentially an event from O and can be assigned a probability. These kinds of random variables are usually associated with a well-known distribution such as a Gaussian function, and $\mathrm{P}(X \in \mathbb{R})$ is called a *probability distribution function*, or PDF. In most algorithmic implementations of continuous random variables, the distribution is either directly mapped to a real value by f_X or "binned" to map to the closest stored probability. In all cases, the rules for probability distributions from Equations 3.1, 3.2, and 3.3 must still apply, and the rule of total probability takes the form

$$\mathrm{P}(x) = \sum_{\forall x \in \mathrm{V}(X)} \mathrm{P}(X = x) = 1. \tag{3.5}$$

The probability distribution $\mathrm{P}(X = x)$ over a single random variable X or a subset of random variables is also known as the *marginal distribution* over the variables in question, or more specifically over all of the possible values contained within them. A distribution over X with three events $X = \{x_1, x_2, x_3\}$ with $\{x_1, x_2, x_3\} \in O$ contains three probabilities, $\mathrm{P}(X = x_1)$, $\mathrm{P}(X = x_2)$, and $\mathrm{P}(X = x_3)$, which sum to 1 as stated in Equation 3.5 as the three are mutually exclusive (i.e. the random variable cannot take more than one value at a time). Just as a probability distribution can be defined over several discrete events, one can be defined in the same way over several random variables, so that the probability of two events occurring can be quantified. The term $\mathrm{P}((X = x \cap Y = y)$ describes a distribution over random variables X and Y, where specific values of the random variables occur, for example $\mathrm{P}(X = x_2 \cap Y = y_1)$. The intersection of events $X = x_2 \cap Y = y_1$ is used to specify that we want the event of both $X = x_2$ and $Y = y_1$ occurring, out of all the different combinations of values of X and Y, and is called a *conjunction*. The probability distribution over two or more random variables is called a *joint distribution*, and conjunctions of

variables in joint distributions are an essential component of Bayesian inference. While combinatoric solutions are often used to determine the probability of a certain combination of events, generally when the likelihood of events are equal (such as throwing n dice and predicting the sum total), defining the probability of a specific combination of random variables is very powerful because we can set the probability of each value in each random variable separately (that is, each die can be "loaded"). Hence, a joint distribution contains a probability for *every* combination of values in the relevant random variables, as all the possible outcomes of the random variables in the canonical outcome space must have an associated probability. Each one of these probabilities is called an *atomic outcome*, and it is easy to see that even a small set of random variables can result in a very large outcome space.

As random variables are used frequently and in many contexts, shorthand notation is common to simplify expressions that use them. In this writing, we we refer to a specific value $\mathrm{P}(X = x)$ with the simple form $\mathrm{P}(x)$, since reference to X should be redundant due to choice of letter (though in unusual cases, the random variable is specified). If the probability distribution itself (and not the probability of just one value) is meant, then simply $\mathrm{P}(X)$ is used. Also, since we use conjunctions very frequently in the process of inference over joint distributions, the probability distribution over a conjunction of X and Y random variables $\mathrm{P}\left((X = x) \cap (Y = y)\right)$ is generally written as an implicit conjunction when dealing with several random variables $\mathrm{P}(X = x, Y = y)$, or incorporating the aforementioned implicit random variables, just $\mathrm{P}(x, y)$. The latter is the most popular format for describing inference using conjunctions. Note that with these simplifications, a probability distribution over random variables can be functionally identical to a simple probability distribution over a number of discrete Boolean events, but in a more compact and powerful form, and following an introduction to the basic concepts, random variables are assumed to be used with regard to all probability distributions.

3.1.3 Statistical Parameters

For the purpose of characterizing and understanding the behaviour of random variables, several standard measures are in use. The most common of these is the *expected value* of a random

variable, more commonly known as the *mean* and represented by E. It is generally defined very simply as the sum of a set of numerical values divided by the number of values in the sum, but this definition assumes equal probability for all values, which is not sufficient for most random variables. A more general definition for a the expected value $\mathrm{E}(X)$ of a random variable X includes the probability $\mathrm{P}(X = x)$ for each value x to become

$$\mathrm{E}(X) = \sum_{\forall x \in \mathrm{V}(X)} x\mathrm{P}(X = x). \tag{3.6}$$

The meaning of expected value is quite literal: it is the value that one would *expect* the random variable to take considering its probability distribution, or in other terms, the probability-weighted average of all possible values the random variable can take. It is easy to see that the expected value is a linear function, so scaling a random variable by some scalar a or adding a constant b will scale or add to its expectation, and the sum of two random variables X and Y is the sum of the expectation

$$\mathrm{E}(aX + b) = a\mathrm{E}(X) + b \tag{3.7}$$

$$\mathrm{E}(X + Y) = \mathrm{E}(X) + \mathrm{E}(Y). \tag{3.8}$$

In order to determine how values of a random variable vary between each other, we can also define the *variance* σ^2 of a random variable X, which is defined in terms of expected values as

$$\sigma^2(X) = \mathrm{E}\left((X - \mathrm{E}(X))^2\right) = \mathrm{E}(X^2) - (\mathrm{E}(X))^2 . \tag{3.9}$$

The variance provides a measure of how "spread out" the values of the random variable are taking their probabilities into account. It is linear with addition like the expected value, but scales as a quadratic function of X

$$\sigma^2(aX) = a^2\sigma^2(X). \tag{3.10}$$

Because of this nonlinearity, a more commonly-used form is the *standard deviation* $\sigma(X)$ which as the notation indicates is simply the square root of the variance $\sigma(X) = \sqrt{\sigma^2(X)}$ and therefore scales linearly.

3.1.4 Conditional Probabilities

Conditional probabilities take these concepts one step further, by effectively creating a tensor that maps related events back to probabilities that map to yet other events. Given sufficient prior information, it is easy to calculate conjunctions of events $o_1 \cap o_2$. For example, in the situation of a navigational sensor on the μrover detecting an obstacle, we can define the Boolean events o_d := "*An obstacle is detected*" and o_p := "*An obstacle is actually present*", so that $o_d \cap o_p$ represents the event that an obstacle is detected correctly. In probabilistic systems, it is typically necessary to present some assumptions regarding the state of the system in order to be able to extract useful information, such as that obstacles are expected to be detected in the area of interest about 30% of the time, which implies an estimate of $\mathrm{P}(o_c) = 0.3$. Estimates of likelihoods such as $\mathrm{P}(o_c)$ that are presented "before" the present time without consideration of experience (i.e. *a priori* knowledge) in probabilistic terms are known as *priors*, and accurate determination of these priors is the single most critical factor for making probability distributions practically valid and useful.

In most cases, we want to be able to determine whether a sensor measurement is actually useable by obtaining the likelihood of $o_d \cap o_p$ occurring. This probability is not necessarily 1 because sensors such as infrared or laser range sensors can be affected by noise, environmental factors such as sunlight, and the surface or material of the target. Rather than attempt to estimate the effects of all of these factors on the set $o_d \cap o_p$, we can estimate the likelihood of successful detection based on experience, which in simplest form could be simply the results of a series of N functional tests to obtain for the i'th outcome, a set $(o_{d,i}, o_{p,i})$. Assuming independent Bernoulli trials for the case of this example (which is a weak assumption in the general case), with the total number of correct detections being

$$n_c = \sum_{i=1}^{N} o_{d,i} \cap o_{p,i} \tag{3.11}$$

so the likelihood of the sensor giving a correct outcome in the case of N Bernoulli trials can then be considered to be

$$\mathrm{P}(o_d \cap o_p) = \frac{n_c}{N} = \frac{\sum_{i=1}^{N} o_{d,i} \cap o_{p,i}}{N}. \tag{3.12}$$

An example would be to state "out of 100 trials, in only 20 cases the sensor both detected an obstacle, and an obstacle was actually present", for which case $\mathrm{P}(o_d \cap o_p) = 0.2$. While this is an informative result, it is relatively useless as it is because it does not contain any context as to how many sensor readings are *expected* for the actual sensor and environment. To obtain a measure of how "good" the measurement is, we need to update our estimate of $\mathrm{P}(o_p)$ based on our knowledge of the detection event o_d. This relationship is defined by the *conditional probability* distribution

$$\mathrm{P}(o_p|o_d) = \frac{\mathrm{P}(o_d \cap o_p)}{\mathrm{P}(o_d)}. \tag{3.13}$$

The conditional probability $\mathrm{P}(o_p|o_d)$ gives the relative proportion of measurements with the object presence event o_p that occur within the set of measurements with the object detection event o_d. If $\mathrm{P}(o_d) = 1$, all measurements will trivially include a detected object and $\mathrm{P}(o_p|o_d)$ reduces to $\mathrm{P}(o_d \cap o_p)$. Similarly, $\mathrm{P}(o_p|o_d)$ is undefined for $\mathrm{P}(o_d) = 0$ because if there is no chance of detecting an object in the first place, the conditional probability has no meaning. As $\mathrm{P}(o_p|o_d)$ itself is a probability distribution, it must satisfy Equations 3.1, 3.2, and 3.3 as well. Also, just as with marginal and joint distributions, the conditional probability of random variables $\mathrm{P}(X = x|Y = y)$ can be used, with the only difference being that the random variables can contain multiple values, and the associated outcome space of each combination must be considered. This is usually abbreviated to $\mathrm{P}(x|y)$.

3.1.5 Conditional Independence

A conditional probability $P(o_p|o_d)$ generally implies that o_d affects o_p somehow, our example being that the event of detection is somehow related to the event of object presence (which is, in fact, the whole point of a sensor!). This is not always true for any set of events or random variables, though. In the general case of a conditional probability, it is possible that $P(o_1|o_2) = P(o_1)$, which implies that the condition o_2 has no effect on the probability of o_1 occurring. A similar conclusion is reached if $P(o_2) = 0$, implying that o_2 will never occur. In this case, events o_1 and o_2 are considered to be *independent* under P. Independence is denoted for events by $o_1 \perp o_2$, and independence is symmetric, so that $o_1 \perp o_2 \implies o_2 \perp o_1$. It can also be proved that P satisfies $o_1 \perp o_2$ if and only if $P(o_1 \cap o_2) = P(o_1)P(o_2)$, which is an additional consequence [Koller & Friedman 2009]. An independence statement over random variables is true for all possible values of those random variables. If two random variables X and Y are independent, we can also write the product of expected values as

$$E(XY) = E(X)E(Y). \tag{3.14}$$

In many cases, rather than two events being completely independent of each other, they are both dependent on a third event that affects them both, even though they are not affected by each other. That is, $o_1 \perp o_2$, and $P(o_1|o_2 \cap o_3) = P(o_1|o_3)$, This is an example of *conditional independence*, and we can say that o_1 is conditionally independent of o_2 given o_3. For random variables in a distribution P that have $X = x \perp Y = y|Z = z$ for all values in X, Y, and Z, we can say that X is conditionally independent of Y given Z and that the values of Z are *observed*. Also in this case, $X \perp Y|Z \iff P(X = x \cap Y = y|Z = z) = P(X = x|Z = z)P(Y = y|Z = z)$. In the case that Z is empty and has no values (the empty event $\emptyset$), X and Y are said to be *marginally independent*.

3.2 Bayesian Inference

The concept of conditional probability is very powerful, and can be considered to be the "transistor" of probabilistic mathematics since it adjusts one distribution based on another. Rearranging Equation 3.13 with general terms gives an alternate definition of $P(o_2 \cap o_1) = P(o_1)P(o_2|o_1)$, which resembles a cancellation of variables (with the contextual addition of "$\cap o_1''$ on the left-hand side). In fact, conditional probabilities can be seen to operate similarly to ratios $a : b$ and differentials da/db in that distributions can be "chained" together (though as with differentials, saying they "cancel" would not be mathematically correct). The *chain rule* for conditional probabilities generalizes this expression to incorporate n events as a product of conditional distributions

$$P(o_1 \cap o_2 \cap o_3 \cap \ldots \cap o_N) = P(o_1)P(o_2|o_1)P(o_3|o_1 \cap o_2)\ldots P(o_N|o_1 \cap \ldots \cap o_{n-1}). \tag{3.15}$$

Or, for random variables

$$P(X_1 \cap X_2 \cap \ldots \cap X_N) = P(X_1)P(X_2|X_1)\ldots P(X_N|X_1 \cap \ldots \cap X_{n-1}). \tag{3.16}$$

This allows any number of events to be related in terms of other events, and is a very important result for inference systems. The order or numbering of the events is not important because the conjunction operator $\cap$ is commutative, meaning that the chain can be re-formulated to better suit the set of known priors. For most robotic systems, the desired result is an expression for a conditional probability that provides a degree of confidence in a certain event with respect to other known events, such as the sensor example already given. Again, because the conjunction operator is commutative, we can replace $P(o_d \cap o_p)$ with $P(o_p \cap o_d) = P(o_d|o_p)P(o_p)$ in Equation 3.13 to express the conditional distribution entirely in terms of event distributions and conditionals. This is the form of *Bayes' Rule*, which is the foundation of all Bayesian inference systems, including Markov chains and Kalman filtering [Engelbrecht 2002].

$$\mathrm{P}(o_p|o_d) = \frac{\mathrm{P}(o_d|o_p)\mathrm{P}(o_p)}{\mathrm{P}(o_d)} = \frac{(likelihood) \cdot (prior)}{(evidence)}. \tag{3.17}$$

Or, for random variables

$$\mathrm{P}(Y|X) = \frac{\mathrm{P}(X|Y)\mathrm{P}(Y)}{\mathrm{P}(X)} = \frac{(likelihood) \cdot (prior)}{(evidence)}. \tag{3.18}$$

We can better understand the consequences of Bayes' rule by thinking of the distribution $\mathrm{P}(o_d|o_p)$ as the *likelihood* of getting a sensor reading if an object to be sensed is present, and $\mathrm{P}(o_p)$ as the probability of an object being present *prior* to the reading being made. The sensor reading itself is the *evidence* given that points to object presence, and becomes a certainty of $\mathrm{P}(o_d) = 1$ after a reading is made, but is generally estimated as $0 < \mathrm{P}(o_d) < 1$ in the absence of a positive sensor reading. In this way, the process of finding $\mathrm{P}(o_p|o_d)$ is known as *Bayesian Inference*, as we are "inferring" the likelihood of an obstacle being present given the statistical characterization of a sensor reading, rather than simply "assuming" that the reading is either correct or not. The term "likelihood" is often used interchangeably with "probability", but in statistical usage "likelihood" denotes an inference based on an existing outcome, while "probability" denotes an independent measure of confidence. In real systems, of course, additional variables will affect the likelihood of getting correct sensor readings, and the more that is included about these variables, the better the result of an inference will reflect the real system. For example, if the event of sunlight o_s affects the accuracy of an optical sensor, it can be included as a background event in a joint distribution by using the conjunction with o_s for conditional dependencies and the conditional on o_s for other distributions. An example would be

$$\mathrm{P}(o_p|o_d \cap o_s) = \frac{\mathrm{P}(o_d|o_p \cap o_s)\mathrm{P}(o_p|o_s)}{\mathrm{P}(o_d|o_s)}. \tag{3.19}$$

3.2.1 Probability Queries

Once a probability distribution is defined, it is necessary to have a consistent method of extracting information from it. The most basic kind of query is the *probability query*, where the goal is to

obtain an estimate of the probability of a given event or occurrence. Formally, we need to obtain the probability distribution associated with a given random variable X given a certain amount of related knowledge or "evidence" Y. The expression that provides the distribution of all values of X given a known value of Y is a conditional probability

$$\mathrm{P}(X|Y = y). \tag{3.20}$$

This is known as the *posterior probability distribution* over X given the condition y, that is obtained "after" the experience of the condition occurs (i.e. *a posteriori* knowledge), and can also be considered to be the marginal over X, but different from X because of the knowledge of y. In most cases, rather than knowing the entire probability distribution, we only are interested in a specific probability of a single value or set of values, usually the highest probability. This is referred to as a *Maximum A Posteriori* (MAP) query, or alternately, "Most Probable Explanation" (MPE). This refers to the most likely values of all variables that are not used as evidence (because evidence is already known and has certain probability), and is described using $\arg\max_x \mathrm{P}(x)$ (the value of x for which $\mathrm{P}(x)$ is maximal) as

$$\mathrm{MAP}(X|Y = y) = \arg\max_x \mathrm{P}(x \cap y). \tag{3.21}$$

A very important point to note is that while the MAP query for a single variable $\mathrm{MAP}(X|Y = y)$ is equivalent to just finding the highest probability of the single-variable distribution $\mathrm{P}(X)$, it is *not* the same as finding all the maximum values in a joint conditional distribution $\mathrm{P}(X \cap Z)$, because the underlying probabilities in this joint distribution depend on both values of X and Z. Rather, a MAP query over a joint distribution finds the most likely complete set of values (x, z) as each combination of values has a different probability. This leads to an important generalization of the MAP query, in which we use a smaller subset of X to find a less likely set of events by independently searching a joint, conditional distribution of the subset. This is called a *marginal MAP query*, and is used frequently in Bayesian inference engines.

$$\mathrm{MAP}(X|Y = y) = \arg\max_{x} \sum_{Z} \mathrm{P}(X \cap Z|Y) \tag{3.22}$$

3.3 Probabilistic Structures

The use of joint probability distributions as a basis for modelling relationships and making predictions is a very powerful concept. Until recently, though, using joint and conditional distributions for all but the simplest systems was an intractable problem. Consider the storage and computation required: a joint distribution $\mathrm{P}(X_1, \ldots, X_L)$ for random variables with N values requires $N^L - 1$ separate probability values to store all N^L combinations of the values in all variables (using the convention of commas in place of the $\cap$ conjunction operator). Additionally, assigning all of the related probabilities from expert knowledge is a daunting task, and automated learning methods are usually used as a result. However, a very large amount of sample data is needed for effectively learning so many probabilities. A small joint distribution of only 8 Bernoulli random variables would require $2^8 = 256$ separate probabilities to be calculated. The reason such a large number of values is necessary is that the joint distribution of all variables at once must represent *every possible combination* of values from those random variables. Clearly, a better solution is required.

The key to solving this problem is the use of *independence* between random variables. Rather than evaluating every possible combination of random variable values, we can parameterize the distribution into a set of specific outcomes. For example, if we let X_m be the result of a coin toss out of M coin tosses with two possible values for x_m being $X = x_{heads}$ and $X = x_{tails}$, we can define the parameterized probability p_m as the probability that the m'th coin toss results in $heads$. Using the probabilities $\mathrm{P}(X = x_{heads}) = 0.5 = p_m$ and $\mathrm{P}(X = x_{tails}) = 1 - 0.5 = 0.5$, we can now represent the distribution using only the M values of p_m rather than all 2^M possible combinations of random variable values. The critical assumption is that *each random variable is at least marginally independent* of the other variables, which allows us to write the distribution over M coin tosses as

$$\mathrm{P}(X = x_1, X = x_2, \ldots, X = x_M) = \mathrm{P}(X = x_1)\mathrm{P}(X = x_2) \ldots \mathrm{P}(X = x_M) = \prod_{m=1}^{M} p_m. \tag{3.23}$$

Note that this gives a general (and intuitive) probability of any combination of independent probabilities, and holds even if the probability p_m changes between coin tosses. Also note that by parameterizing the random variable values, the space of joint distributions becomes an n-dimensional manifold in the original space of $\mathbb{R}^{2^n}$. This is usually desirable because in a given probability distribution, we usually seek the answers to specific questions regarding the probability of certain states occurring. By parameterizing the values of independent (or at least marginally independent) random variables, we can significantly decrease the number of values that must be stored in a joint distribution.

3.3.1 Naive Bayesian Modelling

Parameterization of joint distributions provides an effective way to make storage and calculation more tractable. However, obtaining useful probability estimations for use in a joint distribution is not trivial. Most commonly, probability estimates for values of random variables such as X and Y are obtained through averaging of repeated trial measurements of the stochastic processes they represent. We have already stated that the probability values in a joint distribution $\mathrm{P}(X, Y)$ can be completely different from those in the separate marginal distributions $\mathrm{P}(X)$ and $\mathrm{P}(Y)$. Given a large enough database of experience, it may be possible to estimate the joint distribution completely, but there is little information that can be re-used and every joint distribution would require a separate dataset. It is much more practical to use the chain rule of Equation 3.16 to *factor* the distribution into a conditional probability

$$\mathrm{P}(X, Y) = \mathrm{P}(X)\mathrm{P}(Y|X). \tag{3.24}$$

We can now represent the joint distribution with a marginal distribution $\mathrm{P}(X)$ and a conditional distribution $\mathrm{P}(Y|X)$. This makes much more sense for real systems, as X can now represent a "cause" and Y an associated "effect". Returning to the collision-detection sensor example from the previous section, we can define X as the random variable of object detection with $\mathrm{P}(X = x_d)$ as the parameterized probability that an object is detected, and Y as the random variable of object actual presence with $\mathrm{P}(Y = y_p)$ as the parameterized probability that an object is actually present.

Rather than having to measure the specific distribution $\mathrm{P}(X, Y)$ directly, only the probability of object detection and the accuracy of the sensor are needed.

Of course, this example only considers a single relationship, and probabilistic models of much more complicated systems are needed, with multiple probability dependencies. For example, a vision system operating in concert with the obstacle sensor may be able to detect and triangulate features on nearby objects, and we can use it to increase the accuracy of our estimation of object presence. Let W be the random variable of features being detected within the range of the obstacle sensor, and let $\mathrm{P}(W = w_f)$ be the parameterized probability that enough features are detected to constitute an obstacle. For the moment, we need not consider the complexities of determining whether the features in question constitute an obstacle, as this complexity can be effectively "hidden" by the use of the distribution over W, as we will clarify later. It is natural to think of the information from these two sensors as being related (and hence the variables W and X), since they are effectively observing the same set of obstacles. We can write the conditional joint distribution over these two variables similarly to Equation 3.24 as

$$\mathrm{P}(W, X, Y) = \mathrm{P}(W, X|Y)\mathrm{P}(Y). \tag{3.25}$$

However, from a probabilistic standpoint, it is vital to view the sensors' outputs as being conditionally dependent on a third variable Y, the actual probability of an object being present and the variable of real interest when navigation of a rover is the goal. Knowing this, W and X become merely a means to obtain Y, and as they do not offer any additional information with respect to each other besides the common dependency Y, W and X become marginally independent. For marginal independence of W and X with respect to Y, we can assume formally that for our distribution P

$$\mathrm{P} \vDash (W \perp X|Y). \tag{3.26}$$

The assumption in Equation 3.26 implies that

$$\mathrm{P}(W, X|Y) = \mathrm{P}(W|Y)\mathrm{P}(X|Y). \tag{3.27}$$

Substituting Equation 3.27 into Equation 3.25 gives us the joint distribution over all three random variables as a product of separate conditional and marginal distributions, which as we have already stated is an ideal form for the description of how separate probabilistic processes interact

$$\mathrm{P}(W, X, Y) = \mathrm{P}(W|Y)\mathrm{P}(X|Y)\mathrm{P}(Y). \tag{3.28}$$

Using the chain rule from Equation 3.15 and Equation 3.26 in this way, we can generalize the *factorization* of a joint distribution of M variables $X_1 \ldots X_M$ that are marginally independent but dependent on a condition Z to

$$\mathrm{P}(X_1, \ldots, X_M, Z) = \mathrm{P}(Z) \prod_{m=1}^{M} \mathrm{P}(X_m|Z). \tag{3.29}$$

This principle of factorizing parameterized joint distributions is extremely useful. Besides being able to "split up" a joint distribution into more easily-obtainable conditional factors, if we assume N values for each random variable we can again reduce the number of actual probabilities required across M random variables from the order of $N^M + 1$ to the order of $N \times M + 1$ by means of parameterization, as well as correspondingly reducing the amount of calculation required to a set of multiplications. Note the similarity of Equation 3.28 to the right side of Bayes' rule in Equation 3.18. If $\mathrm{P}(W|Y)$ and $\mathrm{P}(X|Y)$ are considered to be "likelihoods" and $\mathrm{P}(Y)$ is a "prior", then only "evidence" is missing from this expression, indicating that for our example we are only describing a probability of general object detection and the actual reading of the sensor is missing. If we divide both sides of Equation 3.28 by $\mathrm{P}(X)$ and apply the chain rule so that $\mathrm{P}(W, X, Y)/\mathrm{P}(X) = \mathrm{P}(W, Y|X)$, we return to a conditional expression that has a change of dependency on the left hand side

$$P(W, Y|X) = \frac{P(W|Y)P(X|Y)P(Y)}{P(X)}. \tag{3.30}$$

The sensory model we describe here is consequently known as a *naive Bayes Model*, or *idiot Bayes Model*, which is often used for classification of observed features in sensory systems. For these systems, it is assumed that the conditional variable Y contains a set of "classes" that are responsible for the "features" represented by variables $X_1 \dots X_M$. If we want to compute a measure of confidence in whether a class y_1 or y_2 better fit the observed features, we can compare two distributions directly by taking the ratio of their posterior probability distributions

$$\frac{P(X_1 = x_1, \dots, X_M = x_m, Y = y_1)}{P(X_1 = x_1, \dots, X_M = x_m, Y = y_2)} = \frac{P(Y = y_1)}{P(Y = y_2)} \prod_{m=1}^{M} P\frac{(X_m|Y = y_1)}{(X_m|Y = y_2)}. \tag{3.31}$$

3.3.2 Bayesian Networks

For us to be able to properly organize and represent a large set of joint distributions using factorization in this way, we need a method of clearly associating random variables that are dependent on each other, in the case of our sensor example the association of W and X with Y. A directed graph can be constructed, with nodes that represent random variables connected by edges that show the direction of dependence of one random variable on another. For the naive Bayes model in Equation 3.28, such a graph would look like Figure 3.1. A graph or "network" of probability distributions over random variables, with one independent random variable representing a node and directed edges showing its dependencies, is called a *Bayesian Network* (BN), and forms the primary representation of probabilistic relationships in this research.

When discussing Bayesian networks, we will use conventional terminology for describing graphs such as a "node" being an individual element (in this case, defined by a random variable) and an "edge" being a line connection between nodes (in this case, denoting a directed conditional relationship between random variables). A "parent" is a node with one or more edges directed away from it, and a "child" is a node that has edges directed to it from a parent node. Similarly, an "ancestor" is a node that has some sequence of edges leading to the node in question through

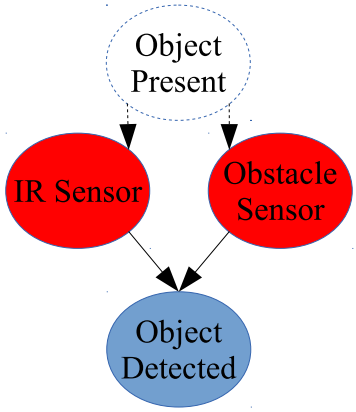


Figure 3.1: Bayesian Network of naive Bayes sensor model

its parents, their parents and so on. A "descendant" is a node that can be reached from the node in question through some sequence of edges through its children, their children, and so on. Note that most nodes are both child and parent; the node that is the focus of the current context forms the reference for the terms, and can have both "parents" and "children" with respect to itself.

A Bayesian network provides a compact way to encode both the structure of a conditional factorization of a joint distribution, and also (equivalently) the independence assumptions about the random variables included in the distribution. Strong independence of a random variable makes a node that has no parents that it is dependent on, such as in the case of Y, while marginally dependent or conditional random variables are nodes with parents connected as in the case of $(W \perp X|Y)$. Hence, the Bayesian network can serve as a framework for formalizing expert knowledge about how the world works. Note that since dependencies are one-way, all edges in a Bayesian network must be *directed*. Also, for closed-form calculation of probabilities, random variables cannot depend on themselves. Therefore, there can be no *cycles* in the graph, where edges leading from a node can eventually create a path back to that node. A Bayesian network structure is therefore generally known as a *directed acyclic graph* or DAG.

The formalization of a Bayesian network is as follows: The structure is formed of a DAG

with nodes corresponding to all random variables X in the graph set $\wp$. Let $\mathrm{Pa}(X)$ denote the set of parent nodes of X, $\mathrm{Ch}(X)$ denote the set of children of X, and $(\wp - \mathrm{Ch}(X))$ refer to all the nodes in $\wp$ that are not children of X. We will use $\mathrm{An}(X)$ and $\mathrm{De}(X)$ for the full sets of ancestors and descendants, respectively. The set of conditional independence assumptions of the variables $X \in \wp$ are then [Koller & Friedman 2009]

$$\forall X : (X \perp (\wp - \mathrm{Ch}(X))|\mathrm{Pa}(X)) \,. \tag{3.32}$$

The Markov blanket for a node X is the set of nodes $\partial X = X \cup \mathrm{Pa}(X) \cup \mathrm{Ch}(X) \cup \mathrm{Pa}(\mathrm{Ch}(X))$, that is the set of X's parents, children, and the parents of X's children. The Markov blanket of X completely determines the behaviour of X, and can be considered the set of nodes that must have known probability distributions in order to obtain a conditional distribution including X. The parents of the child nodes $\mathrm{Pa}(\mathrm{Ch}(X))$ must be included as they can be used to "explain away" the node X. This leads to the network version of the Markov property, which states that X depends only on the values of ∂X regardless of the value of another node $Y \notin \partial X$ as

$$\mathrm{P}(X|\partial X, Y) = \mathrm{P}(X|\partial X). \tag{3.33}$$

The structure of the network we are using is known as naive Bayes, as it is based on the naive Bayes model. Applying the chain rule to all nodes in a Bayesian network, the probability distribution over a given network or subnetwork of nodes $\wp = \{X_1 \dots X_M\}$ can be said to factorize over $\wp$ according to the dependencies in the network if the distribution can be expressed as a product [Koller & Friedman 2009]

$$\mathrm{P}(\{X_1 \dots X_M\}) = \prod_{m=1}^{M} \mathrm{P}(X_m|Pa(X_m)). \tag{3.34}$$

This version of the chain rule is particularly useful for calculating the Conditional Probability Distribution (CPD) or *individual factor* of a given node in the network. Due to the dependency structure of the network, the conditional probability of a given node X depends on all its ancestors, so that $\mathrm{P}(X|\mathrm{Pa}(X))$ must be calculated recursively. A *query* for the posterior probability

distribution of X involves first querying all of its parent nodes $Y \in \text{Pa}(X)$ to determine their probability distributions, then multiplying them by the probability distributions of each parent node Y such that by a simplification of the chain rule

$$\text{P}(X = x) = \sum_{Y \in \text{Pa}(X)} \text{P}(X = x|Y = y)\text{P}(Y = y). \tag{3.35}$$

For discrete random variables, which we will be primarily using, it is important to note that this is effectively a *matrix multiplication*. Representing probability distributions by tables, the distribution $\text{P}(X|Y)$ will be an $L + 1$-dimensional matrix for L parents, with the major dimension of size N for N random variable values in $\text{V}(X)$. Each parent Y is assumed to have a distribution of size M (although each could be different with no change in the concept), so that the distribution $\text{P}(Y)$ is a $N \times 1$ matrix. To calculate the conditional distribution for X, the size $M \times N$ plane of the matrix representing the values from the parent versus the values from the child node, which is the distribution $\text{P}(X|Y)$ is multiplied with the $N \times 1$ distribution $\text{P}(Y)$ just as in Equation 3.35. This results in an $N \times 1$ matrix that is effectively a temporary posterior distribution estimate for $\text{P}(X)$ which avoids frequent recalculation for determining the conditional distributions of children $\text{Ch}(X)$ while traversing the network. This process is graphically illustrated in Figure 3.2. The CPD $\text{P}(X)$ represents a *local probabilistic model* within the network, which for each node represents a local summary of probability for its own random variables.

In this way, any node in a Bayesian network can be queried to obtain a probability distribution over its values. While exact inference as described into a Bayesian network is widely understood to be an NP-hard problem [Dagum & Luby 1993] [Wu & Butz 2005], it is still much more efficient than the raw computation of a joint distribution, and provides an intuitive, graphical method of representing dependencies and independences. It is easy to see how any system that can be described as a set of independent but conditional random variables can be abstracted into a Bayesian network, and that a wealth of information regarding the probability distributions therein can be extracted relatively efficiently. This forms our basic methodology for probabilistic modelling.

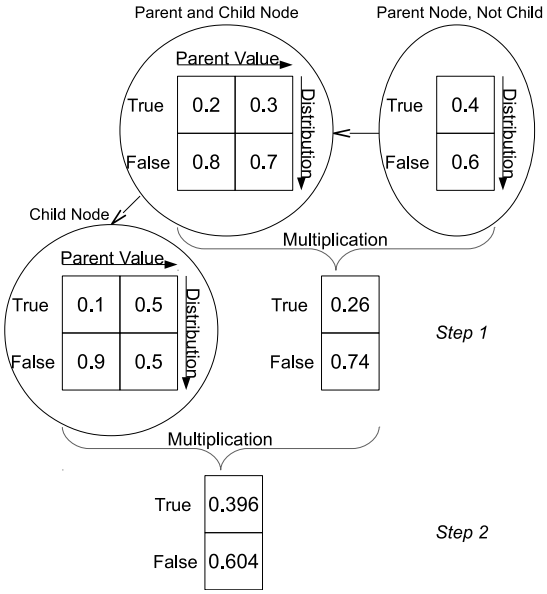


Figure 3.2: Matrix calculation for querying a discrete random variable

3.4 Bayesian Programming

As we now have methods for building probabilistic relationships and making probabilistic inferences, we can make use of this framework to allow a μrover to make intelligent decisions and respond appropriately to uncertain situations. The basis for our approach is inference using a Bayesian network as an abstraction of expert knowledge regarding the rover itself and assumptions about its environment, such as obstacles and hazards. We approach the problem of probabilistic robotics using the Bayesian Robot Programming (BRP) methodology developed by Lebeltel, Bessiere et al [Bessiere *et al.* 2000] [Lebeltel *et al.* 2000] [Lebeltel *et al.* 2004], which provides a quite comprehensive framework for robotic decision-making using inference and learning from experience. Despite having considerable promise and providing a novel solution for reasoning under uncertainty, BRP has not been developed significantly since the initial publications during 2000-2004. We add to this method by formally using Bayesian networks as a knowledge representation structure for programming, and by constructing these networks dynamically with implicit information obtained from the μrover bus and from a store of mission information. There are several advantages to this approach, which include clarity of representation, a practical structure for constructing joint distributions dynamically, and reliance on a proven probabilistic methodology. Also, the use of recursive inference in a Bayesian network avoids the need to manually partition and decompose large joint distributions, which greatly simplifies the programming process.

3.4.1 Logical Propositions

Inference in Bayesian robot programming relies on the concept of *logical propositions* [Bessiere *et al.* 2000]. A logical proposition is essentially a formalization of the state of a probabilistic system, which by our probabilistic framework is represented by a value assignment for a discrete random variable $X = x$ or several discrete random variables. Conjunctions ($X = x \cap Y = y$) and disjunctions ($X = x \cup Y = y$) are then also, by extension, propositions. To distinguish the use of random variable sets from the use of propositions (although logically,

sets and propositions are isomorphic to each other) we will use the propositional logic operators $\wedge$ for conjunction and $\vee$ for disjunction, as well as the shorthand notation for random variables. Hence, propositions will take the form (x), $(x \wedge y)$, and $(x \vee y)$. We add to this the concept of the *negation* of a proposition $\neg x$, which represents the complement of the value x. For a probability assignment $\mathrm{P}(X = x)$, this represents the complementary probability $1 - \mathrm{P}(X = x)$, or informally, the probability of a given event *not* happening.

As with Bayesian networks, the process of inference is used to extract information from propositions, based on conditional probability distributions over a set of random variable dependencies. It is assumed that most propositions are conditional on the same basic set of prior knowledge (or in graph parlance, the random variables most queried are child nodes of the same set of parent nodes). The conjunction of the set of random variables that are considered "prior knowledge" for a given proposition are often shortened to the proposition π for brevity. Based on the rules used for probabilistic inference, only two basic rules are needed for reasoning [Robinson 1965]. The Conjunction Rule

$$\mathrm{P}(X = x \wedge Y = y|\pi) = \mathrm{P}(X = x|\pi)\mathrm{P}(Y = y|X = x \wedge \pi) = \mathrm{P}(Y = y|\pi)\mathrm{P}(X = x|y \wedge \pi) \tag{3.36}$$

and the Normalization Rule

$$\mathrm{P}(X = x|\pi) + \mathrm{P}(\neg X = x|\pi) = 1. \tag{3.37}$$

We assume that the random variables used such as X are discrete with a countable number of values, and that a logical proposition of random variables $[X = x_i]$ is mutually exclusive such that $\forall i \neq j, \neg(X = x_i \wedge X = x_j)$ and exhaustive such that $\exists X, (X = x_i)$. The probability distribution over a conjunction of two such variables using shorthand notation is then defined as the set $P(X, Y) \equiv P(X = x_i \wedge Y = y_i)$. Using this concept of propositions, the Conjunction Rule becomes [Bessiere *et al.* 2000]

$$\forall x_i \in X, \forall y_j \in Y : \mathrm{P}(x_i \wedge y_j|\pi) = \mathrm{P}(x_i, y_j|\pi) = \mathrm{P}(x_i|\pi)\mathrm{P}(y_j|x_i, \pi) = \mathrm{P}(y_j|\pi)\mathrm{P}(x_i|y_j, \pi) \tag{3.38}$$

and an equivalent disjunction rule can be stated as

$$\forall x_i \in X, \forall y_j \in Y : \mathrm{P}(x_i \vee y_j|\pi) = \mathrm{P}(x_i|\pi) + \mathrm{P}(y_j|\pi) - \mathrm{P}(y_j|\pi)\mathrm{P}(x_i, y_j|\pi) \tag{3.39}$$

while the Normalization Rule becomes

$$\forall y_j \in Y : \sum_{\forall x_i \in X} \mathrm{P}(x_i|\pi) = 1. \tag{3.40}$$

The marginalization rule may then be derived for propositions as

$$\forall y_j \in Y : \sum_{\forall x_i \in X} \mathrm{P}(x_i, y_j|\pi) = \mathrm{P}(y_j|\pi). \tag{3.41}$$

For clarity, when applying these rules to distributions over random variables, we do not necessarily have to state the individual propositions (or values). As with common random variable notation, the conjunction rule can be stated without loss of generality as

$$\mathrm{P}(X, Y|\pi) = \mathrm{P}(X|\pi)\mathrm{P}(Y|X, \pi), \tag{3.42}$$

the normalization rule as

$$\sum_X \mathrm{P}(X|\pi) = 1, \tag{3.43}$$

and the marginalization rule as

$$\sum_X \mathrm{P}(X, Y|\pi) = \mathrm{P}(Y|\pi). \tag{3.44}$$

It is assumed that all propositions represented by the random variable follow these rules.

3.4.2 Bayesian Programming

A *Bayesian program* has been defined by Lebeltel et al. as a group of probability distributions selected so as to allow control of a robot to perform tasks related to those distributions. A "Program" is constructed from a "Question" that is posed to a "Description". The "Description" in turn includes both "Data" represented by δ, and "Preliminary Knowledge represented by π. This "Preliminary Knowledge π consists of the pertinent random variables, their joint decomposition by the chain rule, and "Forms" representing the actual form of the distribution over a specific random variable, which can either be parametric forms such as Gaussian distributions with a given mean and standard deviation, or programs for obtaining the distribution based on inputs [Lebeltel *et al.* 2000].

Rather than unstructured groups of variables, we apply these concepts to a Bayesian network of M random variables $\wp = X_1, X_2, \ldots, X_N \in \pi, \delta$, from which an arbitrary joint distribution can be computed using conjunctions. It is assumed that any conditional independence of random variables in π and δ (which must exist, though it was not explicitly mentioned by Lebeltel et al.) is represented appropriately by the Bayesian network, thus significantly simplifying the process of factorization for joint distributions. The general process we use for Bayesian programming, including changes from the original BRP, is as follows:

1. **Define the set of relevant variables.** This involves identifying the random variables that are directly relevant to the program desired. In a Bayesian network, this is implicit in the edges between nodes that represent dependencies. For example, for a collision-avoidance program, the relevant variables include those from the nodes associated with the obstacle sensors and vision system, and are generally easy to identify due to the structure of the network. Usually, a single child node is queried to include information from all related nodes.

2. **Decompose the joint distribution.** The original BRP methodology explicitly partitioned a joint distribution of M variables $\mathrm{P}(X_1, \ldots, X_M|\delta, \pi)$ into subsets, each one a conjunction, and then used the product of the factorization of each subset, a process called *decomposition*

[Lebeltel *et al.* 2004]. To achieve a simpler and easier to automate method, we make use of the properties of the Bayesian network for implicitly including information in parent nodes when queried. A question such as a MAP query of any given node involves knowing the distributions for the parents of that node, and so on recursively until a node with no parents is reached. So only the appropriate child node in the network must be included. We can then apply the factorization rules for joint distributions in Equation 3.29 to reduce $P(X_1, \ldots, X_M|\delta, \pi)$ to a product of conditional distributions which are queried recursively

$$P(X) \prod_{m=1}^{M} P(X_m|\delta, \pi). \tag{3.45}$$

3. **Define the forms.** For actual computations, the joint and dependent distributions must be numerically defined. A distribution over a boolean random variable can have two values that sum to 1, a distribution over a discrete random variable can have several values summing to 1, and a continuous distribution is essentially a function with area underneath equal to 1 that is parametrically defined. The most common function to be used, and the function used for continuous distributions in this work, is the Gaussian distribution with parameters mean $\bar{x}$ and standard deviation σ that define the shape of the distribution, commonly formulated as

$$P(X = x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}. \tag{3.46}$$

A uniform distribution can also be set with $P(X = x) = k$, having the same probability regardless of value. Finally, because continuous distributions are essentially function calls, a distribution that is a function into some arbitrary external process such as a map can be used. In our Bayesian network representation, a node can be discrete, parametric, or functional in nature, and the inference process operates in the same way with only the values changed. The discrete probability values or functional parameters are defined when the network is constructed, but can be changed later during the process of learning.

4. **Formulate the question.** Queries into a BRP system traditionally involve partitioning the

random variables in $\wp$ into three sets: a "searched" set $Se \subset \wp$ for variables that contain information we want to determine, a "known" set $Kn \subset \wp$ for variables that contain an observed state so that $\mathrm{P}(X = x) = 1$ for $X \in Kn$, and "unknown" variables $Un \subset \wp$ that are only stochastically estimated. Under these sets, a "question" is formulated as the posterior probability distribution $\mathrm{P}(Se|Kn, \pi)$, which makes intuitive sense because in any query we want to obtain the probability distribution over a given random variable based on all the distributions that affect it. Although many methodologies can be used to determine the "final" answer to the question, namely a specific probability value in this distribution, we will generally use marginal MAP queries to obtain the value of highest probability taking into account the dependencies in Un, making the form of the question

$$\arg\max_{Se} \sum_{Un} \mathrm{P}(Se, Un|Kn, \delta, \pi). \tag{3.47}$$

The use of a Bayesian network formalizes the relationships of these sets, so that a query into a common child node of Se incorporates information from parents Kn and Un of Se. Se is typically a single node at the "bottom" of the network representing a random variable for action or decision, while Kn typically includes nodes from the "top" where defined parameters and sensory variables that affect the rest of the network are placed, and Un typically includes nodes between the two that act as intermediaries. It is important to note that a "question" is functionally *just another conditional distribution*, and therefore operates in the same way as an additional node in the Bayesian network. Obtaining the conditional distribution for that additional node effectively asks the question regarding the state of the node, which is done for nodes associated with results and actuators.

5. **Perform Bayesian inference.** To perform inference into the joint distribution $\mathrm{P}(X_1, \ldots, X_M|\delta, \pi)$, the "Question" that has been formulated as a conjunction of the three sets Searched (Se), Known (Kn), and Unknown (Un) is posed to the system and solved as a Bayesian inference that includes all relevant information to the set Se. For our Bayesian network implementation. The "Answer" is obtained as a probability distribution, or in the

case of a MAP query, a value from the set Se. For our Bayesian network implementation, with the random variables in Se, Kn, and Un internally linked together as nodes and edges. Nodes associated with actions to be taken typically have conditional distributions that act as "questions" regarding their operational state. If for example, the question is asked "what should the right-side motors do?", the network nodes related to obstacle presence and mapping, and in turn, prior sensor data, will have to be traversed to obtain the "Answer", which is a posterior probability distribution that is used to select a particular value of motor speed. During operation, the components that use node outputs for their values (e.g., motors and control nodes) perform this inference operation periodically, and this allows the system to operate continuously based on the current state of the network.

3.4.3 Bayesian Inference

The last step in Bayesian programming is the actual inference operation used to determine the probability distribution for the variable or set of variables in question. Obtaining the joint distribution $\mathrm{P}(Se|Kn, \pi)$ is the goal, and requires information from all related random variables in $\{Kn, Un, \pi\}$, which in the Bayesian network are visualized as parents of Se. This distribution can always be obtained using the following inference method [Lebeltel & Bessière 2008]. The marginalization rule from Equation 3.44 first allows the inclusion of Un, as

$$\mathrm{P}(Se|Kn, \delta, \pi) = \sum_{Un} \mathrm{P}(Se, Un|Kn, \delta, \pi). \tag{3.48}$$

By the conjunction rule from Equation 3.42, this can be stated as

$$\mathrm{P}(Se|Kn, \delta, \pi) = \frac{\sum_{Un} \mathrm{P}(Se, Un, Kn|\delta, \pi)}{\mathrm{P}(Kn|\delta, \pi)}. \tag{3.49}$$

Applying the marginalization rule again to sum the denominator over both Se and Un, we have

$$\mathrm{P}(Se|Kn, \delta, \pi) = \frac{\sum_{Un} \mathrm{P}(Se, Un, Kn|\delta, \pi)}{\sum_{\{Se,Un\}} \mathrm{P}(Se, Un, Kn|\delta, \pi)}. \tag{3.50}$$

The denominator of Equation 3.50 acts as a normalization term, and for simplicity will be replaced with the constant $\Sigma = \sum_{\{Se,Un\}} \mathrm{P}(Se, Un, Kn|\delta, \pi)$, giving

$$\mathrm{P}(Se|Kn, \delta, \pi) = \frac{1}{\Sigma} \sum_{Un} \mathrm{P}(Se, Un, Kn|\delta, \pi). \tag{3.51}$$

To complete the inference calculation, we only need to reduce the distribution $\sum_{Un} \mathrm{P}(Se, Un, Kn|\delta, \pi)$ into factors that can be determined. To do this, we must assume that these factors are at least marginally independent. While BRP originally reduced these factors into marginally independent subsets, we can assume that independence is denoted by the structure of the Bayesian network, so we only need be concerned with the ancestors of Se. Using only the ancestors of a given node removes the need to scale by Σ. Given that inference into a Bayesian network typically involves querying a single node, we will assume that Se is the singleton $Se = \{X\}$. This can also be accomplished if Se is larger by making X a parent of all nodes in Se. Applying the chain rule again to Bayesian networks, the probability distribution over Se directly depends on the distributions of its parents, which for the moment we will assume are known to be unconditional random variables for clarity. We can factorize the immediate vicinity of $Se = \{X\}$ as

$$\mathrm{P}(Se|Kn, \delta, \pi) = \sum_{Un} \prod_{Y \in \{X, \mathrm{Pa}(X)\}} \mathrm{P}(X|Y)\mathrm{P}(Y). \tag{3.52}$$

This gives us a factorization for a single node. Of course, we cannot assume that the parents of X have no dependencies, and in general should be assumed to have some other dependencies Z so that we have $\mathrm{P}(Y|Z)$. In this case we must consider the parent nodes of the node being queried $\mathrm{Pa}(X)$, the parents of the parent nodes $\mathrm{Pa}(\mathrm{Pa}(X))$, and so on recursively until we have spanned the complete set of ancestors Y with $Y \in \mathrm{An}(X)$. From a purely algorithmic perspective, we can walk the Bayesian network backwards through the directed edges from X, determining the conditional distribution of each node from its parents as we go, and therefore breaking down the determination of the joint distribution into smaller, separate calculations. Considering Z to be the parents of each ancestor node Y and following the method of Equations 3.34, Equation 3.35, and

Equation 3.52, a general expression for the factorization of $\mathrm{P}(Se|Kn, \delta, \pi)$ through the Bayesian network is

$$\mathrm{P}(Se|Kn, \delta, \pi) = \sum_{Y \in \{X, \mathrm{An}(X)} \left[\prod_{Z \in \mathrm{Pa}(Y)} \mathrm{P}(Y|Z)\mathrm{P}(Z) \right]. \tag{3.53}$$

This is a recursive calculation, as we must first obtain the conditional distributions $P(Z|Y)$ for the ancestors Z furthest from the node X before the closer ancestors and parents of X (as in depth-first traversal of the branches of a dependency tree). To save calculations, the temporary estimate $\mathrm{P}(Y) = \mathrm{P}(Y|Z)\mathrm{P}(Z)$ is saved for each node Y for use when calculating $\mathrm{P}(\mathrm{Ch}(Y))$ for the children of Y.

3.4.4 Bayesian Network Representation

To construct an appropriate machine representation of a Bayesian network, it is necessary to consider both the numerical properties of a Bayesian node (or random variable), and the underlying requirements of the hardware and software that support the information contained in the network. As Bayesian nodes are essentially random variables associated with other random variables by way of a joint distribution, an object-oriented approach is taken to describing them using C structures. Unlike most Bayesian network implementations, our implementation is unique in that it uses fixed-point math for storage and calculation and is programmed in C for better portability and calculation efficiency on small-scale embedded systems.

The fundamental properties of a Bayesian Node are the probability distribution of the random variable, and the way that distribution is affected by the knowledge of other nodes nearby. For numerical compactness and code efficiency, the values of a node are represented as $M \times N$ distribution matrices, where each row represents the probability distribution of the random variable, and each column represents the effects of linked nodes. Total probability requires that all values in each row m sum to 1. The joint distribution P of probability values associated with a given random variable is the content actually stored, as well as a vector of labels for each of the local values in the random variable itself.

At minimum, a random variable with N possible values will have a $1 \times N$ distribution matrix. A parent node will increase the number of distributions that must be accounted for in the child node, causing at most M possible probability distributions for each one of its M possible variable values. If two or more parent nodes are present, the total number of combinations of affecting values must be considered in the child node. Hence, a parent with 2 values and a parent with 3 values will together contribute $2 \times 3 = 6$ possible probability distributions, and if the node itself has 4 possible values, a total of $4 \times 2 \times 3 = 24$ probability values must be stored in total. In general, if each parent has a distribution N_l values in size, and there are L parents, then the number of distributions M possible in the child node are

$$M = \prod_{l=1}^{L} N_l. \tag{3.54}$$

As each child node must have an $N \times M$ matrix, assuming that parent nodes have similar numbers of values, the storage size of the node scales roughly as N^L. This can be mitigated by designing a deeper graph with more nodes and less parents per node, as the simplifying properties of the Bayesian network will decrease the total storage required. A parent node with an $M_l \times N_l$ distribution matrix, regardless of the number of parents and the size of M_l, will still only contribute N_l values to its child nodes, making the speed of storage size increase dependent on the size of the probability distributions in question. A given node X will then have to store a table of size $|\mathrm{V}(X \cup \mathrm{Pa}(X))|$.

The actual method of storing the distributions is not trivial. Because the dimensionality of the distribution matrix effectively increases with each parent (adding a new set of combinations of variables), fixed-dimension matrices are not practical for nodes where new parents may have to be added dynamically. Many languages use nested template classes and other object-oriented methods for implementing N-dimensional storage. However, for speed and compactness of storage, we use a single C array for storage of the distribution, and index it with a linear index that is a function of the parent numbers of the node. To create the index, when addressing an array as an $L + 1$-dimensional matrix for L parents we use an extension of the conventional mapping to a linear array index i for a row-major-order matrix, which for row (m) and column (n) indices is formulated as $n + m * columns$. By recognizing that each additional dimension must be indexed by multiplying past the sizes of all preceding dimensions, we can consistently index into a linear array at location i using matrix indices m_1 for dimension 1 of size M_1 (we choose columns here for consistency), m_2 for dimension 2 of size M_1(we choose rows here for consistency), and $m_3, m_4, \ldots$ and above for additional matrix dimensions of size $M_3, M_4, \ldots$ respectively, obtaining

$$i = m_1 + m_2 M_1 + m_3 M_2 M_1 + \ldots + m_{L+1} \prod_{l=1}^{L} M_l = \sum_{n=1}^{L+1} \left(m_n \prod_{l=1}^{n-1} M_l \right). \tag{3.55}$$

This $O(L)$ complexity operation must be done for every index into the array, although short-

cuts can be taken when traversing a dimensional axis, such as incrementing by m_1 for traversing rows, $m_2 M_1$ for columns, etc.

A set of functions for creating Bayesian networks have been implemented in C utilizing the fixed-point math system. The functions were specifically targeted at making the system reliable, efficient and small for use on embedded processors. Nodes of the network are stored as structures indexable in a static array to ensure that all nodes can be searched easily and no memory leakage occurs. The linked nodes and probability distribution in each node are also dynamically allocated. Each time a dynamic element is accessed, the pointer to it is tested for validity. This lowers the chance of segmentation faults and corrupted data. Currently, the network is built from both hardware data and XMLBIF files that contain the network structure and probability distributions.

In a Bayesian network representation, everything the rover "knows" is represented as linked random variables. The "knowledge" (priors, etc.) is initially provided by the rover's "self-aware" devices. "Abstractions" that need to be inferred are provided by the mission plan. Using the communications system detailed above, known values are obtained directly from hardware via the system bus, with models, capabilities, and links between the nodes provided by devices themselves. Abstractions such as the definitions of obstacles and mapped areas of interest are expert knowledge that is typically included in the mission planning data. Figure 3.3 shows a simple example of a Bayesian network constructed in this manner.

3.4.5 Node and Variable Types

While all nodes in the Bayesian network we construct represent random variables, the variables represent a variety of different real abstractions, and are consequently constructed using different data sources and roles within the Bayesian network. All nodes are kept as similar as possible in terms of data representation and programming interface, and all effectively function as probability distributions over random variables, although data nodes in particular usually call external sources to calculate appropriate responses to queries in real time.

Ability or (*A*) nodes provide the abstraction of functions, and include the sensor models and movement models used to convert physically measurable quantities into concepts that are suitable

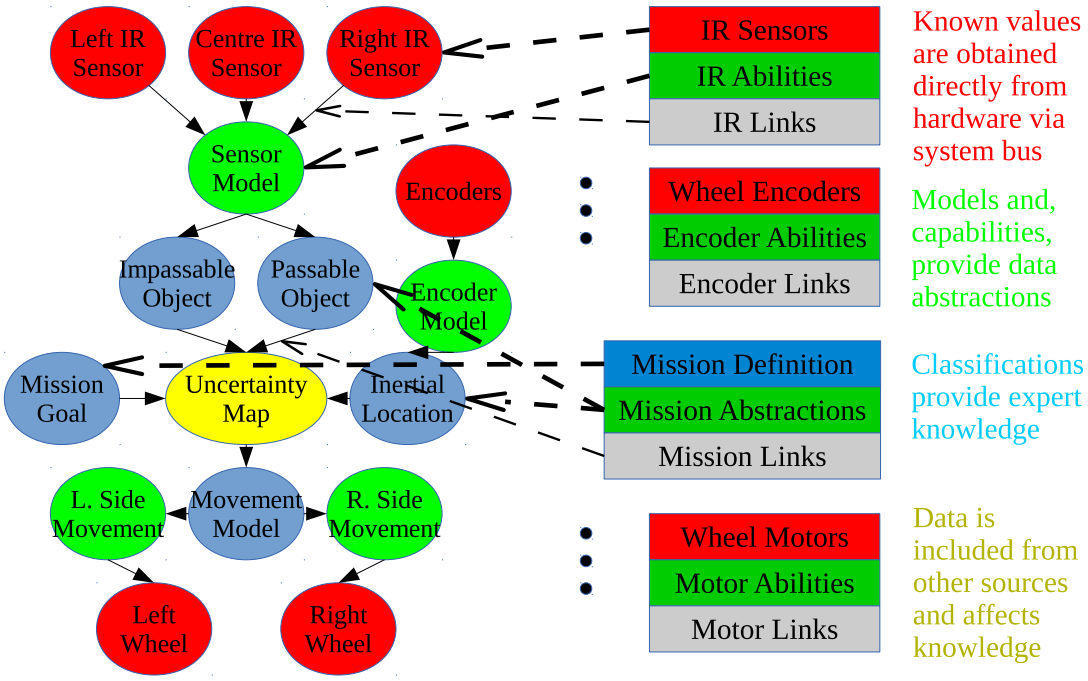


Figure 3.3: Bayesian network built using knowledge of rover structure

for inference and reasoning such as “high”, “medium”, and “low”. Ability nodes operate similar to fuzzifier/defuzzifier rules in a fuzzy logic system, but can also be implemented as probabilistic algebraic functions, such as the sensor model for the infrared range sensors. They are usually a parent or child of a bus node. Ability nodes for sensors are typically included in the set of knowns Kn and ability nodes for actuators are typically searched in the set Se to make decisions about actuator movement and operational functions.

Bus or (B) nodes include hardware devices connected to the system bus such as the obstacle sensors, inertial sensors, environmental sensors, and actuator drivers. Sensor nodes usually are parent nodes only that provide actual measurement values to ability nodes, while actuator nodes are usually child nodes only that are dependent on ability nodes representing their actual function, and have joint distributions representing a “question” regarding the actuator state. A bus node acts as a “terminal” or “endpoint” in the network that allows interaction with the rover itself and the outside world, and generally is associated with nodes in Se or Kn.

Classification or (C) nodes are used to interpret and translate information within the network. For example, an inertial sensor can only give information about probability of body orientation and an obstacle sensor about the probability of the presence of an obstacle, but determining the likelihood of the rover tipping over or the likelihood of a collision are conditional judgments based on inference. Classification nodes act as drivers in determining behaviours to respond to external or internal events, and are most similar to the BRP concept of "Parametric Forms". Classification nodes are also generally included in the set of unknowns Un.

Data or (D) nodes act as an interface to additional information outside the network. These include probability maps built as two or three-dimensional distributions, and mission information databases used to build probability distributions dynamically based on external instructions. Data nodes typically use function pointers to refer queries to functions that provide the appropriate information based on the system state, and are similar to the BRP concept of "Program Forms". As they provide data, they are usually included in the set of knowns Kn.

In Figure 3.3 and others, ability nodes are coloured green, bus nodes are coloured red, classification nodes are colored blue, and data nodes are colored yellow. Learned probability data is stored and shared between μrovers in XML format. There have been several different proposed Bayesian Network Interchange Formats (BNIF) developed, such as BIF, XMLBIF, and XBN [Cover 1999]. Despite a lack of current information and limited scope of adoption, it appears that the XBN standard is the most current, although the earlier XMLBIF format appears more efficient to parse. Both are based on XML, but no current DTD template for either is available. Therefore the format used is a loose interpretation of the XMLBIF standard with additional tags to support complete storage of the node structure such as node type $\{A, B, C, D\}$.

3.4.6 Behaviours

Conventional robotic programming often involves the use of a state machine or fuzzy system to determine a specific procedure or set of goals that determine how the robot "behaves". This simplifies programming significantly by discretizing the operation of a robot into individual, manageable states that can correspond to steps of mission goals, and a behavioural system can sometimes fulfill complex mission goals without significant planning or mapping [Serdar Guzel & Bicker 2012]. When implemented as discrete states, some method of identifying when to transition between behaviours and which behaviour to transition to is needed. In a probabilistic system such as BRP, the current state is determined probabilistically by a state transition table that functions much like a probabilistic Finite State Machine (FSM) [Hy *et al.* 2004]. Finite state machines are the most common formal method for behaviour implementation as they provide a clear and simple abstraction of behavioural mechanisms. However, for all but the simplest behavioural systems, FSM implementations suffer from exponential complexity growth in a similar fashion to dependent joint distributions. The number of potential behaviours and transitions grows too large to be completely accounted for. To overcome this problem, we can make use of the simplifying properties of a Bayesian network with marginally independent nodes to drive behavioural changes [Lazkano *et al.* 2007]. Hierarchical behaviour systems that function as a Bayesian network have been used similarly to BRP [Mansard *et al.* 2004], but unless necessary, we will focus on a simple, complementary implementation of the concepts posed thus far. Behavioural modification can be performed in a probabilistic system in three main ways [Bessiere *et al.* 2000]:

1. by altering the observed random variables that have a probability of 1 or close to it due to measurement results from a sensor or other observation system.

2. by changing the distributions of the random variables used to compute the joint distribution in the Bayesian network, that is, changing the context of the "question" that is asked when determining the next appropriate action.

3. by modifying the "question" or conditional joint distribution $P(Se|Un, Kn, \delta, \pi)$ used to

determine a course of action, thus directly changing which random variables are queried without changing the known and unknown data

Observing the random variables so that the distribution changes to a certain value is simply a process of observing sensors or state variables that control the outcomes of queries into the network, so changing the observed variables should drive the content of a behaviour rather than the selection of the behaviour. The distributions within the random variables of the network reflect prior knowledge of the system and are changed by learning processes, so that queries into the network can reflect experience. Therefore, the most appropriate and flexible method for reflecting different behaviours is to represent a behaviour by the "Questions" that are asked of the network. This is analogous to formulating a question based on "what is the information I need for this behaviour" rather than "what will the information say if I am in this behaviour", which would be the case if the Bayesian network were altered based on the behavioural state. The concept of incorporating behavioural elements into the nodes of the network itself also has the problem of circular reasoning; if the Bayesian network also drives behaviour selection, probabilistic loops can occur which would break the DAG structure and make accurate closed-form evaluation of queries difficult. In BRP, each "question" does include behavioural prior data in the form $\pi_{behaviour}$, making the form $\mathrm{P}(Se|Un, Kn, \pi_{behaviour})$ [Lebeltel *et al.* 2000]. This essentially means changing the joint distribution by changing the variables present in $\pi_{behaviour}$, or for a Bayesian network, changing which nodes are linked to the "question", which could be considered to be a dedicated child node queried as well. Using this concept, the current behaviour is determined by a joint distribution over a random variable Be, where $\mathrm{V}(Be) = b_1, b_2, \ldots b_m$, which is used to determine which of m behaviours is chosen, and the parents $\mathrm{Pa}(Be)$ are the variables that are of interest in determining the current behaviour b_m. Since obtained data δ is assumed to be not involved in changing behaviour, this makes the probability distribution over behaviours

$$\mathrm{P}(Be = b|Un, Kn, \pi). \tag{3.56}$$

The issue then becomes, how does the current behaviour affect the operation of the system? Considering the choice of b_m to be a simple state machine, Be can be implemented as a programming construct such that the searched node Se or joint distribution used as a "question" can change based on the state. We can describe this with the joint distribution $\mathrm{P}(Se_{behaviour}|Un, Kn, \pi)$, in which we change the variable queried rather than the preliminary knowledge $\pi_{behaviour}$. However, to maintain our focus on building systems that can be completely represented by a Bayesian network, we will instead define Be as a Classification (C) type node that for maximum generality, depends on all nodes that determine behaviour and is depended on by all nodes that change their distribution in response to the selected behaviour. We prevent cycles from forming in the DAG by requiring that the set of ancestors of Be and the set of descendants of Be are disjoint, or

$$\mathrm{An}(Be) \cap \mathrm{De}(Be) = \emptyset. \tag{3.57}$$

Typically, members of $\mathrm{De}(Be)$ are ability (A) or bus (B) nodes that directly control the responses of the system to behavioural stimuli, and therefore are generally nodes that are queried periodically to determine actions, and should have probability distributions that fundamentally change the "question" based on the selected behaviour. Figure 3.4 illustrates the use of a behavioural node in this way.

3.4.7 Learning

With a functioning Bayesian inference system, the μrovers need to be able to feed back learned data to the network in a practical manner in order to improve the results of future inferences. Probabilistic systems are well-suited to learning methods, as priors are often obtained directly from statistical measurements of uncertain processes. Dynamic Bayesian Networks (DBNs) are Bayesian networks that are built by determining the structure of a given system from dependence and independence relationships determined from analysis of available data [Yehezkel & Lerner 2006]. In a dynamic Bayesian network, extra probabilities p that control the likelihood of a given outcome are assumed to be available which can be estimated from data (system identification), usually by maximum-likelihood criteria over N data sequences such as

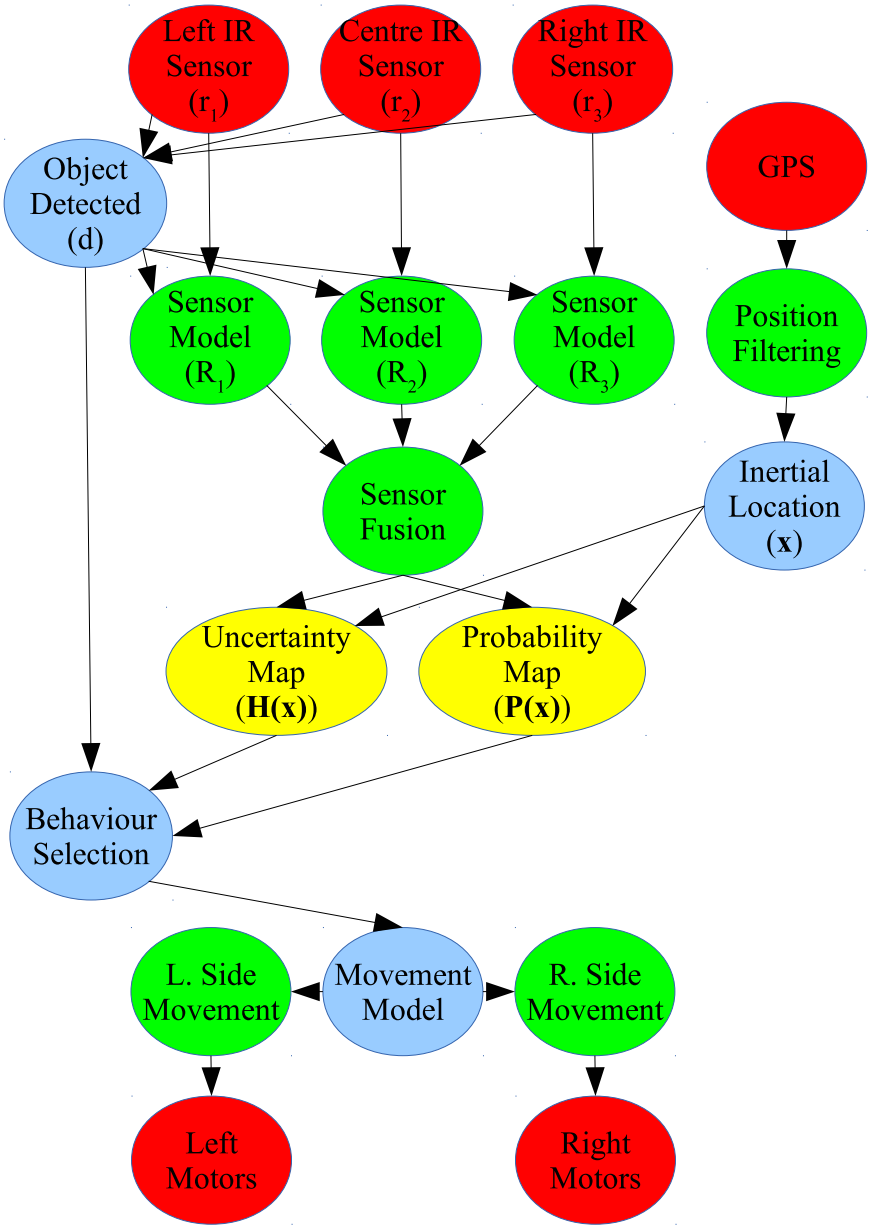


Figure 3.4: A small Bayesian network with behaviours

$$p_{ml} = \arg\max_{p} \mathrm{P}(y|p) = \arg\max_{p} \sum_{n=1}^{N} \log\left(\mathrm{P}(y^n(T)|p)\right). \tag{3.58}$$

However, for the case of a robotic system with well-defined relationships and capabilities, attempting to automatically determine the dependence of variables that have well-defined roles already only provides an opportunity for additional uncertainty to occur. Therefore DBNs are better suited for analysis of raw data such as the analysis of gathered scientific data, while deterministically-structured BNs are appropriate for intuitive programming of known systems, such as μrover control. We instead focus on refining the performance of the existing Bayesian network by updating the probability distributions over each random variable in the network.

While in traditional programming methods and fuzzy systems one would specify a certain set of conditions that trigger a variable value, using probability distributions has the advantage of being able to specify a given probability distribution that is associated with variable values during operation. Effectively, this means that instead of explicitly programming the effect of each condition in the network, the effect is determined under the hypothesis that we know the condition. This methodology is known as *inverse programming* [Hy *et al.* 2004], and is especially useful for classifying sets of sensory data or determining what behaviour is currently appropriate.

The limitations of maximum-likelihood criteria are that they do not include any prior knowledge or experimental information about the distribution in question. If the expected value is not achieved experimentally (as would occur if an incomplete set of data or incorrect model are used) then a value could be "learned" that does not accurately reflect the statistics behind real mechanisms. It is therefore more appropriate to use Bayesian methods to update Bayesian priors [Bessiere *et al.* 2000]. Over a set of N data sequences, if the random variable Y takes on values $y(1), \ldots, y(N)$, the probability distribution over a given probability variable p can be written as

$$\mathrm{P}(p|y(1), \ldots, y(N)) = \frac{\mathrm{P}(y(1), \ldots, y(N)|p)\mathrm{P}(p)}{\mathrm{P}(y(1), \ldots, y(N))} \tag{3.59}$$

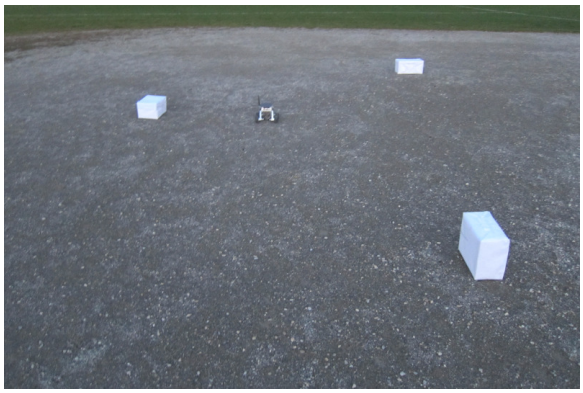


Figure 3.5: Beaver Prototype Testing in Sand and in Outdoor Test Area with Obstacles

3.5 Bayesian Mapping

To make use of the probabilistic systems we have described for the basic task of environmental mapping and navigation, we apply Bayesian probabilistic models to the sensor and mapping hardware on the μrover and allow it to map a small area of outdoor space with known obstacles using only the infrared range sensors for obstacle detection. The area mapped by the μrover is shown in Figure 3.5. It is open except for three boxes, which due to the low sensitivity of the infrared sensors are covered with white paper for high reflectivity. In this test, time-averaged GPS is used for coarse position sensing, as the infrared sensors contain far too little information for both obstacle detection and localization. A full SLAM implementation using the monocular camera for feature detection is detailed later in this work.

Being able to infer the likelihood of success or failure given uncertain data is important for maximizing results in a time and resource-limited mission scenario. We use simple single-output sensors with a statistical model to monitor the environment in front of the micro-rover. To avoid necessitating the involvement of human operators, the system for data collection and decision making has to be implemented on the embedded micro-rover hardware itself. We make use of a small Bayesian network to perform the obstacle detection and navigation task. Due to the limitations present on the micro-rover platform used in this study, all numerical algorithms are being implemented in fixed-point arithmetic, with the necessary scaling and normalizing applied.

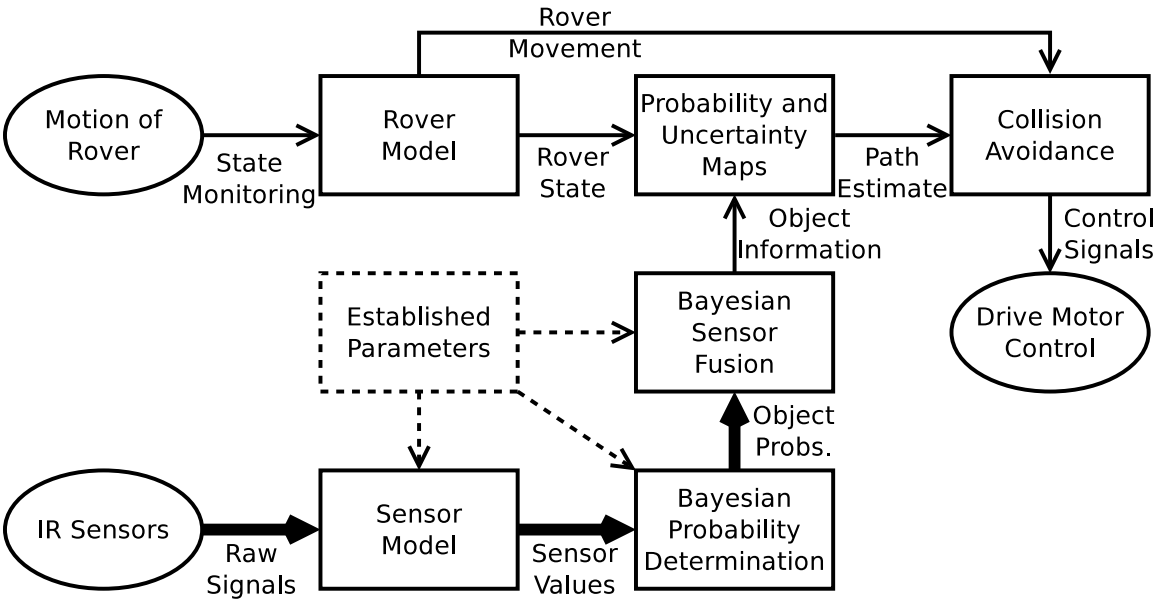


Figure 3.6: Bayesian Mapping Operational Flowchart

This kind of Bayesian system has not been applied and tested in an actual planetary micro-rover of this type. As a goal for a simple Bayesian sensing and control system, we construct an uncertainty map of the rover's surroundings that can be used to identify prominent features reliably and avoid collisions during forward motion. We do not attempt to localize the rover itself using this information due to the limited information provided by the range sensors, which would require too much movement during measurements to perform simultaneous localization and mapping. Figure 3.6 shows the flow of information through the system. A sensor model for simple infrared range sensors is used with a directional sensor fusion model to improve the accuracy of the sensors, and a Bayesian method is used to infer the likelihood of object presence at a given location on the map, structured as a Bayesian network to simplify design and calculation. In addition, the uncertainty in the measurement made is estimated, mapped, and used to drive the search methodology. A set of behaviours is then applied to the likelihood map to implement obstacle avoidance. To evaluate performance in a real-world scenario, this system is implemented on a small micro-rover prototype and tested in an outdoor area with obstacles present.

3.5.1 Bayesian Network

A Bayesian network provides a compact way to encode both the structure of a conditional factorization of a joint distribution, as well as the independence assumptions about the random variables included in the distribution [Koller & Friedman 2009]. We use a small Bayesian network to relate the sensors, model, mapping, and actuators by means of probabilistic inference into the network.

The Bayesian network structure used to relate the various aspects of μrover operation for the mapping task is shown in Figure 3.7. This network is the structure used specifically for probabilistic mapping of a given area using IR range sensors. Each sensor or actuator, represented by a red "bus" node, is connected to a model, represented by a green "ability" node, which provides an abstraction of its capabilities. These are utilized in turn by blue "classification" nodes that are used to determine abstract information such as "collision on the left" versus "collision on the right" and yellow "data" nodes that provide information such as a probability map. The connections between nodes indicate dependencies when calculating a joint distribution over a given node. Each node stores a probability distribution that can be indexed by its child nodes, either as a discrete set of probability values or a continuous function that returns a probability value. Hence both discrete and continuous distributions can be stored in the Bayesian network. It also provides a formal method of obtaining factorizations over nodes based on the dependencies encoded with the Bayesian network [Bessiere *et al.* 2000].

3.5.2 Sensor Model

For the purposes of this study, the rover is tasked to observe all objects encountered and statistically identify obstacles. The μrover's infrared range sensors are used to detect objects based on infrared light reflected from their surfaces, and each provide a range observation measurement r. Each sensor model is considered to be of a Bernoulli type with a probability of correct object detection β_r, a fourth-order polynomial function of the range sensor state, modelled as a random variable R, where [Hussein & Stipanovic 2007]

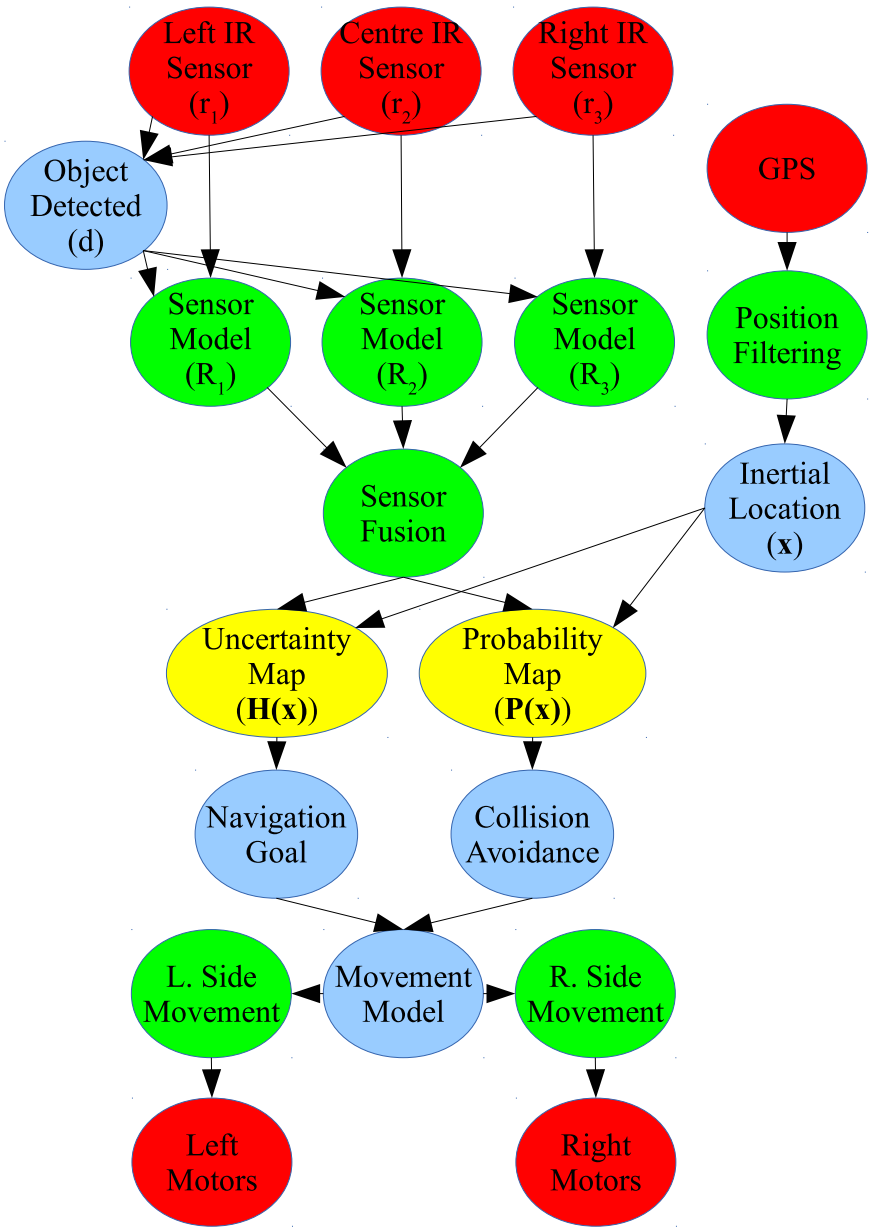


Figure 3.7: A Bayesian network used for mapping with IR sensors

$$R = \|\mathbf{x} - \bar{\mathbf{x}}\| \in \mathbb{R}^+ \tag{3.60}$$

with $\bar{\mathbf{x}}$ being the rover's estimated current location and $\mathbf{x}$ being the location of a sensed object. We can quantify the the probability β_r that the range sensor state R is correct by defining it as

$$\beta_r = \begin{cases} \beta_b + \frac{1-\beta_b}{{r_{max}}^4}({r_{max}}^2 - R^2)^2, & \text{if} \quad R \leq r_{max} \\ \beta_b, & \text{if} \quad R > r_{max} \end{cases} \tag{3.61}$$

where β_b is the base likelihood of correct object detection, assumed to be $\beta_b = 0.5$ so that an even likelihood of correctness is assumed if the object is outside the sensor's range since no actual information of object presence will be provided to the sensor. r_{max} is the sensor's maximum range of approximately $2m$, beyond which correct and incorrect object detection are equally likely. The peak value of β_r if the object being observed is located at $\bar{\mathbf{x}}$ is 1 (certainty) and occurs closest to the sensor, which generally has higher accuracy when closer to an object. This provides a model by which the assumption of object presence at R can be made based on the actual measurement of r.

A probabilistic method is used to update the probability of object presence given a range observation r. We define the likelihood of an object being present at range R from the current location and time $t + 1$ given the range observations r as $\mathrm{P}(R|r, t + 1)$ and taking into account the reliability of the sensor. Although we have no other reference for object detection besides the range sensors, we can increase accuracy by including any knowledge we have already regarding object presence at a given map location $\mathbf{x}$ with the variable $o = 0, 1$ with 1 defining an object being present and 0 defining an object not being present. This can be written in the form $\mathrm{P}(R|r, o, t + 1)$, which can be solved for by applying Bayes' rule as [Wang & Hussein 2011a]

$$\mathrm{P}(R|r, o, t + 1) = \frac{\mathrm{P}(r|R, t)\mathrm{P}(R, t)}{\mathrm{P}(r, t)} = \begin{cases} \frac{\beta_r \mathrm{P}(R,t)}{2\beta_r \mathrm{P}(R,t) - \beta_r - \mathrm{P}(R,t) + 1}, & \text{if} \quad o = 1 \\ \frac{(1-\beta_r)\mathrm{P}(R,t)}{-2\beta_r \mathrm{P}(R,t) + \beta_r + \mathrm{P}(R,t)}, & \text{if} \quad o = 0 \end{cases} \tag{3.62}$$

where we use the law of total probability and the fact that $\mathrm{P}(r|R, t)$ is given by β_r. Equation 3.62

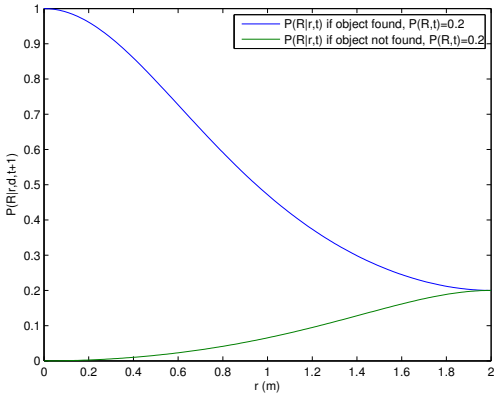


Figure 3.8: Probability distributions $\mathrm{P}(R|r, d, t+1)$ for range sensor model

represents the "Sensor Model" nodes in the Bayesian network of Figure 3.7. This probability distribution is shown for the case where the probability of object detection by itself is constant as $\mathrm{P}(R, t) = \mathrm{P}(R) = 0.2$ in Figure 3.8. Of note is the fact that regardless of object detection o, the distribution converges to $\mathrm{P}(R, t)$ at $r = r_{max}$. To determine whether an object has been contacted, we need to make sure the output of the range sensor is above the noise level of the device, for which we make sure at least two samples in a row indicate that contact well above the RMS error σ_r is made.

$$o = \begin{cases} 1, & \text{if} \quad r_{raw}(t) > 2\sigma_r \wedge \quad r_{raw}(t-1) > 2\sigma_r \\ 0, & \text{otherwise} \end{cases} \tag{3.63}$$

3.5.3 Sensor Fusion

Three directional infrared sensors are placed on the front of the rover for use in obstacle detection, with the side sensors angled 30° out from the central sensor. To improve the detection reliability of objects in the probability map, the sensor data is combined using a linear opinion pool. Each sensor set at an angle θ_r is assumed to have a sensor angle of view w_r and a Gaussian horizontal detection likelihood, so the detection probability incorporating angular deviation for each sensor is estimated as

$$\alpha_s = \frac{1}{w_r^2\sqrt{2\pi}} * e^{-(\frac{\theta_r}{w_r})^2}. \tag{3.64}$$

To combine probability information in the Bayesian network, we can create a node that depends on other nodes above it in the structure. To combine the information from the three sensors together, we use the "Sensor Fusion" node, which provides an opinion pool for N sensors at angles of $\theta_n, n = 1 \ldots N$ with respect to a primary direction by

$$\mathrm{P}(R|r, o, t, \theta_1, \theta_2, \cdots \theta_N) = \sum_{n=1}^{N} \mathrm{P}(R|r, o, t) * \frac{1}{w_r^2\sqrt{2\pi}} * e^{-(\frac{\theta_n}{w_r})^2}. \tag{3.65}$$

3.5.4 Map Updates

Mapping of object probabilities is done at each point $\mathbf{x}$ within the sensor range r_{max} of $\bar{\mathbf{x}}$ at the angle θ_r. The rover's map is structured as an occupancy grid spanning the area in question, and functioning as a probability distribution over obstacle presence O. Initially, every grid element in the map is initialized to $\mathrm{P}(R)$, the estimated probability of encountering an obstacle in this environment, and the map is updated at the locations pointed to by the sensors. We consider map updates to be a Bayesian parameter estimation problem using a joint probabilistic model to obtain the probability $\mathrm{P}(O|R, \mathbf{x}, t)$ of an obstacle being present at $\mathbf{x}$. We use the estimate of the probability of a range measurement given an obstacle at $\mathbf{x}$, the probability of an obstacle $\mathrm{P}(O, \mathbf{x}, t)$, and the prior for any obstacle detection with Bayes' rule to form

$$\mathrm{P}(O|R, \mathbf{x}, t+1) = \frac{\mathrm{P}(R|O, \mathbf{x}, t)\mathrm{P}(O, \mathbf{x}, t)}{\mathrm{P}(R, \mathbf{x}, t)}. \tag{3.66}$$

A two-dimensional Gaussian function can be used to estimate $\mathrm{P}(O|R, \mathbf{x}, t+1)$ on the map to capture the uncertainty in positional measurement.. While the main goal is statistical identification of obstacles, it is also important to know how much certainty is present at each point in the map. The uncertainty in a measurement can be modelled as the informational entropy present [Wang & Hussein 2011b]. The information entropy at $\mathbf{x}$ and time t for a Bernoulli distribution P_s=$\{P_s, 1 - P_s\}$ are calculated as

$$\mathrm{H}(O|R, \mathbf{x}, t+1) = \min_{t}(-\mathrm{P}(R, \mathbf{x}, t)\ln(\mathrm{P}(R, \mathbf{x}, t)) - (1 - \mathrm{P}(R, \mathbf{x}, t))\ln(1 - \mathrm{P}(R, \mathbf{x}, t))). \quad (3.67)$$

The minimum of all measurements over t taken at a mapped point $\mathbf{x}$ is used to reinforce that uncertainty decreases over time as more data is gathered. The use of a probabilistic sensor model allows the rover's view of the world to be "fuzzy", so that rather than assuming a precise location for obstacles and landmarks, the rover can choose the path that is least likely to contain obstacles, and also consider the locations on the map that contain the least information to be the least reliable. This makes the system more robust to position errors and sensor inaccuracies, as it will attempt to choose the best solution while considering the presence of statistical errors such as Gaussian noise.

3.6 Navigation Algorithms

The rover maintains two maps of its operating area, one for detected object probability $\mathrm{P}(O|R, r, \mathbf{x})$ and one for accumulated entropy $\mathrm{H}(O|R, r, \mathbf{x})$, which are constant over t unless sensory data is added at any point $\mathbf{x}$. As the rover is currently only intended to travel several meters to carry out a mission goal, this is sufficient for short-range travel. The "Probability Map" node provides an interface from the occupancy grid to the Bayesian network by representing the map as a two-dimensional discrete probability distribution. As a probability distribution, the map can be part of a query to obtain the probability of obstacles at a specific location, or updated as part of a learning process with the original map serving as the prior distribution. The main difference in considering the map a probability distribution is that each row or column must be normalized to the size of the map for queries to be accurately carried out.

3.6.1 Mapping Methodology

Assuming the rover is present at the centroid of a grid element $\bar{\mathbf{x}}$ then a sensor reading at range r will affect positions $\mathrm{P}(\mathbf{x}_x + r\cos(\theta_r), \mathbf{x}_y + r\sin(\theta_r))$ and any adjacent locations within the distri-

bution spread of the sensor. Entropy is mapped in much the same way, as $\mathrm{H}(\mathbf{x}_x + r\cos(\theta_r), \mathbf{x}_y + r\sin(\theta_r))$ for orthogonal Cartesian components $\mathbf{x}_x$ and $\mathbf{x}_y$ for each θ_r. Before any data is gathered, the probability map is initialized to $\mathrm{P}(R)$, while the entropy map is normally initialized to 1 at every point. As the search process within the map is not generally randomized, this leads to the same pattern being repeated initially. To evaluate the impact of varying initial uncertainty on the search pattern, the entropy map was also initialized using a pseudo-random value $\delta_h < 1$ as $1 - \delta_h < \mathrm{H}(\mathbf{x}) < 1$ in a separate set of tests. For efficiency, the rover is assumed to only evaluate a local area of radius d_{max} in its stored map at any given time t. Mapping continues until there are no remaining points with uncertainty exceeding a given threshold, $(\forall \mathbf{x}, \mathrm{H}(\mathbf{x}) < H_{desired})$.

Considering the set Δ_s as all points within this radius, where $\{\forall \mathbf{x} \in \Delta_s, \|\mathbf{x} - \bar{\mathbf{x}}\| < d_{max}\}$, the target location $\hat{\mathbf{x}}$ is generally chosen to be the point with maximum uncertainty:

$$\hat{\mathbf{x}'} = \arg\max_{\Delta_s}(\mathrm{H}(O|R, r, \mathbf{x})). \tag{3.68}$$

However, as this typically results in mapping behaviour that follows the numerical map search algorithm, it is desirable to provide a more optimal search metric. We choose the map location with maximum uncertainty within the margin δ_h and minimum distance from the rover and use a logical OR with Equation 3.68 in the algorithm as

$$\hat{\mathbf{x}} = \hat{\mathbf{x}'} \vee [\arg\max_{\Delta_s}(\mathrm{H}(O|R, r, \mathbf{x}) - \delta_h) \wedge \arg\min_{\Delta_s}\|\mathbf{x} - \bar{\mathbf{x}}\|]. \tag{3.69}$$

To make the target destination available as a probability distribution, we use a maximum likelihood estimation process to create a Gaussian distribution over a target location random variable T with the mean at the horizontal angle θ_t aiming toward $\hat{\mathbf{x}}$ and with a standard deviation of π radians so that at least a 180° arc has probability of reaching the target.

$$\mathrm{P}(T|R, \mathbf{d}, \theta_t) = \frac{1}{\frac{\pi}{2}\sqrt{2\pi}} e^{-\frac{(x-\theta)^2}{\pi^2/4}} \tag{3.70}$$

The node "Navigation Goal" in Figure 3.7 encapsulates this so that queries can be performed.

The spread of the Gaussian distribution allows a wide variety of choices for steering angle even if obstacles are present between the rover and the target.

3.6.2 Navigational Decisions

For forward navigation, we would like to travel to the target location by the shortest route possible while minimizing the risk of a collision. From a naive Bayes standpoint, what we need is a probability distribution that includes both the probability of collision across the possible steering angles of the rover, and the distance to potential obstacles so that closer obstacles count as more dangerous than distant ones. This can accomplished by first obtaining the vector $\mathbf{d} = \hat{\mathbf{x}}' - \bar{\mathbf{x}}$ from the rover to the target point, and considering the area of the map occupying the solid angle θ_d from this vector centred on the rover, such that the angles $\theta \in [-\theta_d \dots \theta_d]$ with respect to $\mathbf{d}$ are considered. A set of M discrete angles $\theta_m, m = 1 \dots M$ can then be evaluated by summing the normalized total probability of encountering an obstacle over all locations along the length d_{max} vector $\mathbf{x}_\theta$ to form the probability distribution

$$\mathrm{P}(O|R, \mathbf{d}, \theta_m) = \frac{\sum_{\mathbf{x}_\theta} \mathrm{P}(O|R, \mathbf{x}_\theta)}{d_{max}}. \tag{3.71}$$

This provides a metric for the likelihood of encountering an obstacle in the direction θ and allows existing map data to help plan routes. To incorporate the concept of distance, the sum of the distribution is weighted by the distance $|\mathbf{x}_\theta - \bar{\mathbf{x}}|$, effectively increasing the likelihood of encountering closer obstacles.

$$\mathrm{P}(O|R, \mathbf{d}, \theta_m) = \frac{\sum_{\mathbf{x}_\theta} (d_{max} - |\mathbf{x}_\theta - \bar{\mathbf{x}}|) \mathrm{P}(O|R, r, \mathbf{x}_\theta)}{d_{max}} \tag{3.72}$$

The probability distribution $\mathrm{P}(O|R, r, d, \theta_m)$ is implemented in the "Collision Avoidance" node in Figure 3.7, and is used together with the target location to drive the "Movement Model" node, which calculates a distribution $\mathrm{P}(M|O, R, r, \mathbf{d})$ over a random variable of movement direction M to prioritize the target point, but avoid areas with high obstacle likelihood.

$$\mathrm{P}(M|O,R,\mathbf{d}) = \mathrm{P}(T|R,\mathbf{d},\theta_t) + (1 - \mathrm{P}(O|R,\mathbf{d},\theta_m)) \tag{3.73}$$

The query $\arg\max_M \mathrm{P}(M|O,R,r,\mathbf{d})$ is then used to determine the best choice of direction for forward movement.

3.7 Visual Odometry

An essential part of navigation for any mobile robot is the detection of its environment, which is often accomplished through visual means for reasons already stated. In addition to environmental visual mapping, which is a relatively long-distance application, it is also beneficial to obtain short-distance visual measurements for motion estimation and odometry. This is the domain of optical flow algorithms. While optical flow can be accomplished by simply following the frame-by-frame movements of feature points like those described for use in the ORB algorithm, accurate odometry of a so-called sparse optical flow algorithm in three dimensions is limited because of the relative scarcity of feature points and the lack of symmetry in most sparse feature distributions. A dense optical flow algorithm is preferred if the computational power is available due to higher overall accuracy and more even estimation of the flow field.

3.7.1 Optical Flow from Optical Mice

Perhaps the simplest way of implementing an optical flow algorithm is by using optical mouse hardware with refocused lenses for ground tracking. Optical mice are by far the most common hardware optical flow devices, and the technology has become mature enough that the optical tracking provided by commercial ASICs is quite robust [Ng 2003][Sekimori & Miyazaki 2007]. Most optical mice use red LED illumination of the tracking surface as CMOS-based image sensors are more sensitive to near-infrared wavelengths, but laser-based optical mice are now available that offer higher tracking resolutions. Despite this, relatively few applications of optical mouse technology to the planetary rover odometry problem are found because the COTS hardware associated with them is purpose-built for tracking while in contact with perfectly flat, smooth surfaces. On these surfaces, conventional mouse optics allow for up to $0.5m/s$ of speed, but only about $\pm 1mm$ of vertical movement before tracking is lost, with some dependence on the directionality of the surface texture pattern [Minoni & Signorini 2006]. Accuracy of $0.8mm$ is possible on many surfaces, but combined displacements across different axes and distortion of the ground under the sensor can also cause increased error. The sensor should be placed at

a greater distance from the ground to achieve better tracking, and an array of sensors should be used if possible [Palacin *et al.* 2006]. To track relative motion on arbitrary surfaces at larger distances (often on the order of centimetres for a rover with suspension) different optics must be used to increase the collimation of the optical system. Using customized focusing lenses, optical mouse hardware has been tested successfully for useable odometry even from airborne vehicles.

Optical mice typically use Digital Image Correlation (DIC) for comparison of monochrome images of the tracking surface, a widely-applicable technique that has been in use since the 1970s [Keating *et al.* 1975]. The basic algorithm for DIC is based on iterative maximization of a correlation coefficient derived from comparing subsets of the pixel intensities across two or more images. For a point (x_i, y_j) in an untransformed image that is transformed to a point (x'_i, y'_j) in a subsequent image, the correlation coefficient $r_{i,j}$ is a function of the displacement components v_x and v_y for the center of a given sub-image in the x and y directions respectively and the displacement gradients relating v_x and v_y with x and y as follows [Konecny & Pape 1981]

$$r_{i,j}(v_x, v_y, \frac{\partial v_x}{\partial x}, \frac{\partial v_x}{\partial y}, \frac{\partial v_y}{\partial x}, \frac{\partial v_y}{\partial y}) = 1 - \frac{\sum_i \sum_j [\mathbf{F}(x_i, y_j) - \bar{\mathbf{F}}][\mathbf{F}'(x'_i, y'_j) - \bar{\mathbf{F}}']}{\sqrt{\sum_i \sum_j [\mathbf{F}(x_i, y_j) - \bar{\mathbf{F}}]^2 \sum_i \sum_j [\mathbf{F}'(x'_i, y'_j) - \bar{\mathbf{F}}']^2}} \tag{3.74}$$

where $\mathbf{F}$ and $\mathbf{F}'$ are intensity matrices at the sub-images where the points (x_i, y_j) and (x'_i, y'_j) are located, respectively. To simplify the estimation of v_x and v_y, any motion can be assumed to occur only in the plane perpendicular to the camera's imaging axis (which is implicitly assumed for optical mouse movement and is considered an approximation for rover movement). This makes the transformation between $\mathbf{F}$ and $\mathbf{F}'$ an affine transformation, which can be approximated as

$$x' = x + v_x + \frac{\partial v_x}{\partial x}\Delta x + \frac{\partial v_x}{\partial y}\Delta y \tag{3.75}$$

$$y' = y + v_y + \frac{\partial v_y}{\partial x}\Delta x + \frac{\partial v_y}{\partial y}\Delta y. \tag{3.76}$$

For odometry applications, deformation of the image is not considered, and only an estimate of overall translation from the displacement components u and v is propagated. However, for planar surfaces this allows a high degree of accuracy, limited only by the camera resolution and the quality of the optics.

For planar movement, the basic kinematic equation of motion for a rover with heading angle θ and turn rate ω can be considered as

$$\theta(t+1) = \theta(t) + \omega \tag{3.77}$$

$$\mathbf{X}(t+1) = \mathbf{X}(t) + \mathbf{R}\theta(t+1)\mathbf{t} \tag{3.78}$$

where $\mathbf{X}(t)$ is the rover's position at time step t, $\mathbf{R}\theta(t+1)$is the rotation matrix about the z axis for heading angle θ and $\mathbf{t}$ is the translation vector of forward-backward movement. For complete odometry of a rover, it is desirable to find both $\mathbf{t}$ and ω from optical measurements, but a single optical sensor only senses planar translation $\mathbf{t}$. Thus, two or more sensors at known position vectors $\mathbf{M}_i$ relative to the coordinate origin of the rover must be used. As the optical sensors are functionally omni-directional but operate with predefined body axes, it is implicitly assumed that the x forward axis and y sideways axis are aligned with the x and y body axes of the rover. Figure 3.9 shows the location and orientation of the optical sensors on the μrover with respect to the body coordinates.

For calibration of the sensors, the angular offset $\Delta\theta_i$ can be obtained by driving the rover directly forward for a distance and using the x and y distances Δx_i and Δy_i measured by the i'th optical sensor as

$$\Delta\theta_i = \text{atan}\frac{\sum_{t=1}^{n}\Delta y_i(t)}{\sum_{t=1}^{n}\Delta x_i(t)}. \tag{3.79}$$

The position coordinates $m_{x,i}$ and $m_{y,i}$ for the i'th sensor can also be estimated from a rotation about angle ω as

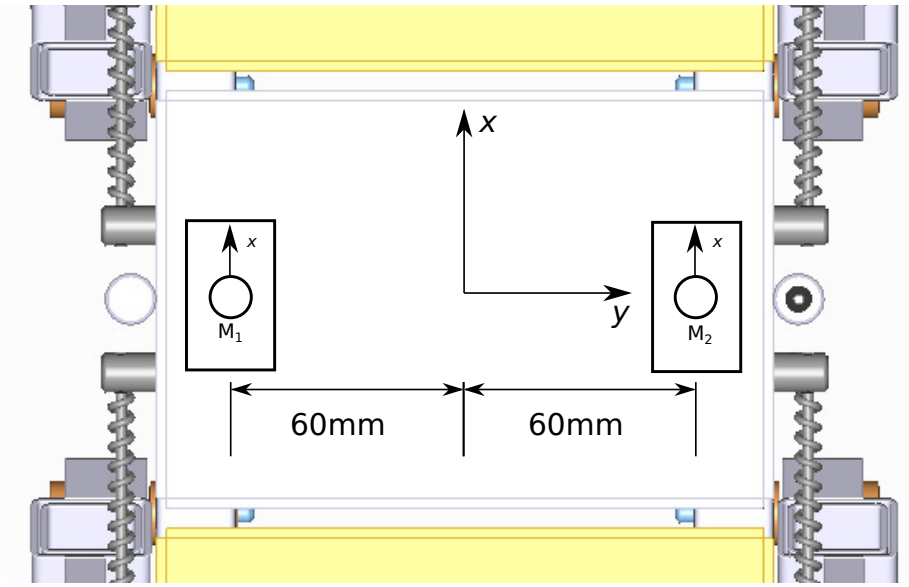


Figure 3.9: Optical flow sensor locations on body

$$m_{x,i} = \frac{1}{\omega}\sum_{t=1}^{n}(\sin\Delta\theta_i\Delta x_i(t) + \cos\Delta\theta_i\Delta y_i(t)) \tag{3.80}$$

$$m_{y,i} = -\frac{1}{\omega}\sum_{t=1}^{n}(\cos\Delta\theta_i\Delta x_i(t) - \sin\Delta\theta_i\Delta y_i(t)). \tag{3.81}$$

Because of this position offset of $(\Delta x_i, \Delta y_i)$, the position of the optical sensors changes as the orientation of the body changes, as

$$\mathbf{M}_i(t+1) = \mathbf{M}_i(t) + \mathbf{R}_\theta \begin{pmatrix} \Delta x_i \\ \Delta y_i \end{pmatrix}. \tag{3.82}$$

To find a common transformation in terms of $\mathbf{t}$ and ω between $\mathbf{M}_i(t)$ and $\mathbf{M}_i(t+1)$, a quadratic criterion can be used for a total of k sensors [Mudrova *et al.* 2011]

$$E(\mathbf{t},\omega) = \sum_{i=1}^{k} |\mathbf{R}_\omega\mathbf{M}_i + \mathbf{t} - \mathbf{M}_i(t+1)|^2. \tag{3.83}$$

Fortuitously, this criterion has the same form as that used in the Iterative Closest Point algo-

rithm [Lu & Milios 1994]. Consequently, the same solution can be used in analytical form

$$\omega = \text{atan}\frac{S_{xy'} - S_{yx'}}{S_{xx'} - S_{yy'}} \tag{3.84}$$

$$\mathbf{t} = \begin{pmatrix} \bar{x}' \\ \bar{y}' \end{pmatrix} - \mathbf{R}_\omega \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} \tag{3.85}$$

where $\bar{x}$, $\bar{y}$, $\bar{x}'$, and $\bar{y}'$ are the mean values of $x_i(t)$, $y_i(t)$, $x_i(t+1)$, $y_i(t+1)$ respectively over all k sensors and S represents the corresponding cross-covariances between the current and next values of $x_i(t)$, $y_i(t)$, $x_i(t+1)$, $y_i(t+1)$. The estimated $\mathbf{t}$ and ω can then be used to determine $\mathbf{X}(t+1)$ using Equation 3.78.

3.7.2 Optical Flow from Camera Vision

Using dedicated sensors is sufficient for estimating two-dimensional movement from optical flow. To estimate three-dimensional movement, the information available from a higher-resolution camera with a deep field of view is typically required. A variety of optical flow methods in computer vision are available [Prazdny 1980][Negahdaripour & Lee 1992][Dille *et al.* 2009][Wang *et al.* 2009]. One very promising algorithm was created by Gunnar Farneback for field odometry measurements using the movement of segments within images between two frames. Its main advantage is that it can be solved linearly and can perform faster than a typical optical flow algorithm that looks for motion in every pixel of the image. Using filtering over time for accuracy, the vector field can be used to estimate the motion of the camera between frames. An efficient implementation of this algorithm has been done in OpenCV.

The Farneback optical flow algorithm makes use of three-dimensional orientation tensors, which are represented as a positive semidefinite matrix $\mathbf{T}_{3x3}$. The in-frame two-dimensional velocity vector (v_x, v_y) is extended to a three-dimensional spaciotemporal vector $\mathbf{v}$ with two image dimensions and one dimension of time (successive frames) by [Farnebäck 2000].

$$\mathbf{v} = \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix} \textit{ and } \hat{\mathbf{v}} = \frac{\mathbf{v}}{|\mathbf{v}|}. \tag{3.86}$$

To estimate orientation tensors, the weighted pixel region surrounding a grid point chosen for segmentation is projected onto a second-degree polynomial, in this case assuming affine motion for image coordinates x and y and using one equation for each component of two-dimensional velocity

$$v_x(x, y) = ax + by + c \tag{3.87}$$

$$v_y(x, y) = dx + ey + f. \tag{3.88}$$

This can be written in terms of the spatiotemporal vector as

$$\mathbf{v} = \mathbf{Sp}. \tag{3.89}$$

where

$$\mathbf{S} = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.90}$$

and

$$\mathbf{p} = \begin{pmatrix} a & b & c & d & e & f & 1. \end{pmatrix}^T \tag{3.91}$$

To estimate the parameters a, b, c, d, e, f from parameter vector $\mathbf{p}$, the matrix product $\mathbf{v}^T\mathbf{Tv}$ should be minimized [Farnebäck 2000]. The minimum value in this expression is given by λ_3, the smallest eigenvalue. With exact translation and no noise, $\hat{\mathbf{v}}^T\mathbf{T}\hat{\mathbf{v}} = 0$, but practical constraints prevent this. As a minimization, the use of $\mathbf{v}^T\mathbf{Tv}$ requires less computation than the normalized

version, but the isotropic component $\lambda_3\mathbf{I}$ should be removed from $\mathbf{T}$ to avoid bias against large velocities. Applying the spatiotemporal model, the function to be minimized sums over the image area [Farneback 2001]

$$d_{total}(\mathbf{p}) = \sum_i \mathbf{v}_i^T \mathbf{T}_i \mathbf{v}_i = \sum_i \mathbf{p}^T \mathbf{S}_i^T \mathbf{T}_i \mathbf{S}_i \mathbf{p}. \tag{3.92}$$

To ensure that the last element of $\mathbf{p}$ remains 1, $\mathbf{p}$ and the intermediate product $\mathbf{Q} = \sum_i \mathbf{S}_i^T \mathbf{T}_i \mathbf{S}_i$ should be partitioned with the parameter α as

$$\mathbf{p} = \begin{pmatrix} \mathbf{p}_p \\ 1 \end{pmatrix}, \tag{3.93}$$

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_p & \mathbf{q} \\ \mathbf{q} & \alpha \end{pmatrix} \tag{3.94}$$

such that Equation 3.92 becomes

$$d_{total}(\mathbf{p}) = \mathbf{p}_p^T \mathbf{Q}_p \mathbf{p}_p + \mathbf{p}_p^T \mathbf{q} + \mathbf{q}^T \mathbf{p}_p + \alpha \tag{3.95}$$

which is minimized by $\mathbf{p}_p = -\mathbf{Q}_p^{-1}\mathbf{q}$. For purposes of movement estimation, this is known as *input flow*. An example optical flow field obtained using this technique is shown in Figure 3.10.

3.7.3 Egomotion from Optical Flow

From the perspective of an autonomous vehicle, the most important quantities obtained from optical flow processes are the estimates of translational and rotational motion. Assuming that the observer is responsible for the only motion in the scene and given a sufficiently large set of flow vectors, a planar or three-dimensional movement transformation can be reconstructed from the perspective of the observer (known as egomotion)[Zhuang *et al.* 1988][Scharer *et al.* 2004][Campbell *et al.* 2005]. The egomotion problem is not new, having been studied by Longuet-Higgins, Prazdny and others

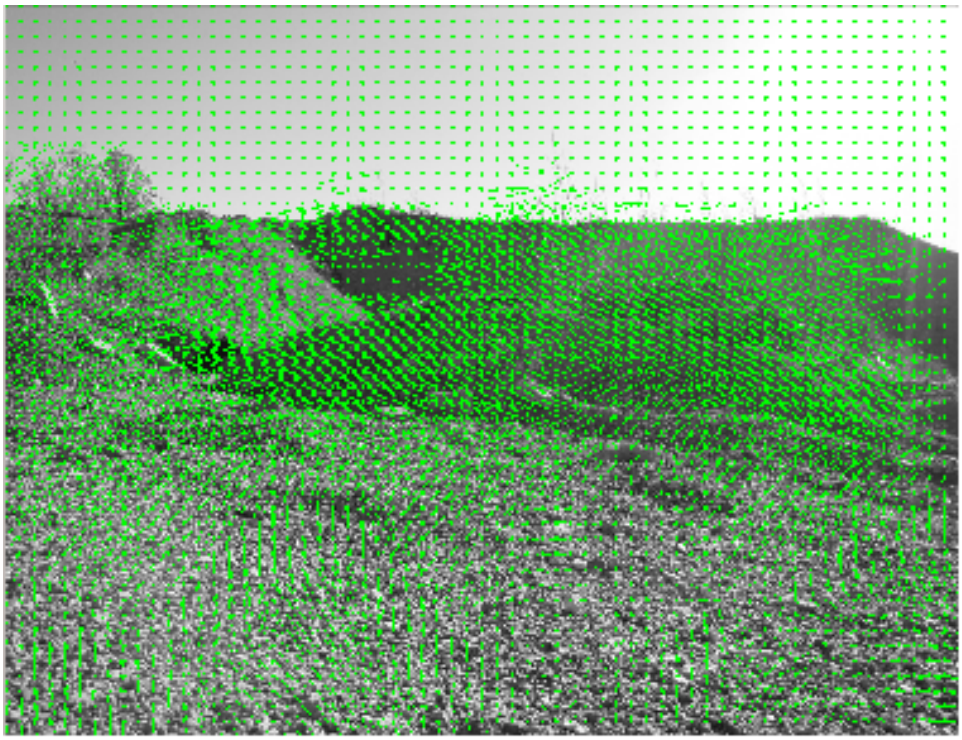


Figure 3.10: Example of optical flow field

[Longuet-Higgins & Prazdny 1980] [Prazdny 1980][Prazdny 1981][Festl *et al.* 2012] with respect to the human vision system. It is also possible to detect objects and obstacles using optical flow [Dur 2009], though these techniques are more sensitive to noise and require much more complex computation. In general, egomotion is assumed to occur in an egocentric (centred on the observer) frame of reference and cause changes in the velocity field opposite to the observer's motion. Complete estimation of three-dimensional motion is difficult because the calculated optical flow depends on the distance to the point in the scene as well as the six possible motion parameters. Some constraints on motion or point selection are usually applied to overcome this [Irani *et al.* 1994]. Better accuracy can also be achieved by using a wider field-of-view or multiple overlapping camera view fields [Tsao *et al.* 1997].

The goal is to represent every step of motion by a unique translation $\mathbf{t}$ and rotation $\mathbf{R}$, so that given an initial point $\mathbf{x}$ every observable point has a relative movement $\Delta\mathbf{x} = \mathbf{t} + \mathbf{R} \times \mathbf{x}$. Traditionally, this requires a set of nonlinear equations to be solved iteratively, and a unique solution is not

guaranteed depending on the consistency of the optical flow field [Negahdaripour & Lee 1992]. Most optimization methods used in this way also have a statistical bias that results in systematic errors over time from isotropic noisy input [MacLean 1999], and in most cases, the optimization is not computationally efficient. For these reasons, we will concentrate on linearized methods that are free of statistical bias.

Because optical flow does not provide accurate tracking of large displacements, it is important to ensure that images are taken close enough together that only a small amount of movement with a minimum of deviation occurs in between frames. At low speeds, this is practical for planetary rovers. If simple planar ($\mathbf{R} = \mathbf{I}_{3x3}$) flow is creating the optical flow field, it can be estimated through simple affine transformation. For a given point location starting at (u, v) that is transformed to a new point location (u', v') through a vector distance $(\Delta u, \Delta v)$

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} \Delta u \\ \Delta v. \end{pmatrix} \tag{3.96}$$

To find the corresponding coordinates in three-dimensional space that this flow corresponds to, we assume that the total movement of the observer is small enough that a homography between the two images can be approximated by an affine transformation, written as

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}. \tag{3.97}$$

Calculating the affine coefficients, denoted by a, b, c, d, e, f, requires at least three points and the assumption that all optical flows in the image should be coplanar. This requires that the planar flow $\Delta x = u' - u$ and $\Delta y = v' - v$. In practice, the points used for calculation may not match, and an approximation is required. After this, Δx and Δy can be calculated by [Wang *et al.* 2009]

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} + \begin{pmatrix} u \\ v \end{pmatrix}. \tag{3.98}$$

For introducing the effects of nonlinear projection, such as in the case of the human eye, a

pinhole camera model with focal length f results in a three-dimensional (x, y, z) projection on the two-dimensional image plane of (u, v) calculated by $u = fx/z$ and $v = f * y/z$. Under a three-dimensional translation vector $\mathbf{t} = (t_x, t_y, t_z)$ and rotation matrix $\mathbf{R}$, the three-dimensional instantaneous relative movement is

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = -\mathbf{t} - \mathbf{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{3.99}$$

The corresponding movements, or *model flow*, of a projected point in the image plane using the pinhole camera model is [Longuet-Higgins & Prazdny 1980]

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{z} \begin{pmatrix} -f & 0 & x \\ 0 & -f & y \end{pmatrix} \mathbf{t} + \frac{1}{f} \begin{pmatrix} xy & -(f^2 + x^2) & fy \\ (f^2 + y^2) & -xy & -fx \end{pmatrix} \mathbf{R}. \tag{3.100}$$

To find the optimal translation and rotation that best describe the observed optical flow, it is sufficient to optimize the squared Euclidean distance of the residual vector between the input flow and the model flow [Bruss & Horn 1983]. The bilinear optimization constraint used to obtain this can be calculated by inserting the optimized depth into Equation 3.100 for which the algebraic transformation is [Heeger & Jepson 1992]

$$0 = \mathbf{t} \left(\begin{pmatrix} fv \\ -fu \\ yu - xv \end{pmatrix} - \begin{pmatrix} -(f^2 + y^2) & xy & fx \\ xy & -(f^2 + x^2) & fy \\ fx & fy & -(x^2 + y^2) \end{pmatrix} \mathbf{R} \right) = \mathbf{t} \left(\mathbf{M} - \mathbf{HR} \right). \tag{3.101}$$

While subspace methods are often used for ego-motion recovery from optical flow fields [Heeger & Jepson 1992], there are similar alternatives. For linearized optimization of the estimated ego-motion and removal of statistical bias as described above, we follow the method of Raudies and Neumann, which is similar to the use of subspaces, but has shown better accuracy and computational efficiency [Raudies & Neumann 2009]. Computational efficiency is achieved

by using only the linearly independent upper triangle and diagonal part of the matrix **H** to form the base polynomial vector **e**. All constraints in **e** are used to obtain more robust estimates. The auxiliary variable vector **k** of all combinations of **t** and **R** components is used to obtain a linear version of Equation 3.101. These are defined as [Raudies & Neumann 2009]

$$\mathbf{e} = \begin{pmatrix} -(f^2 + y^2) \\ xy \\ fx \\ -(f^2 + x^2) \\ fy \\ -(x^2 + y^2) \end{pmatrix}, \ \mathbf{k} = \begin{pmatrix} t_x r_x \\ t_x r_y \\ t_x r_z \\ t_y r_y \\ t_y r_z \\ t_z r_z \end{pmatrix}. \tag{3.102}$$

The linear optimization problem then becomes a minimization of the integral over the entire image with flow vectors **v**

$$\int_{flows} (\mathbf{t}^t \mathbf{M} + \mathbf{k}^t \mathbf{e})^2 \, d\mathbf{v}. \tag{3.103}$$

Setting the partial derivatives of Equation 3.103 to 0 in the classic method of linear optimization gives a linear set of nine equations in nine variables if expanded

$$\int_{flows} (\mathbf{t}^t \mathbf{M} + \mathbf{k}^t \mathbf{e}) \cdot \mathbf{e}^t d\mathbf{v} = 0. \tag{3.104}$$

$$\int_{flows} (\mathbf{t}^t \mathbf{M} + \mathbf{k}^t \mathbf{e}) \cdot \left(\mathbf{M} \frac{\partial (\mathbf{k}^t \mathbf{e})}{\partial \mathbf{t}} \right)^t d\mathbf{v} = 0. \tag{3.105}$$

To reduce this, Equation 3.104 can be solved for **k**. The expression for **k** and the partial derivative $\partial(\mathbf{k}^t\mathbf{e})/\partial\mathbf{t}$ can be substituted into Equation 3.105 to obtain the homogenous linear system of equations

$$0 = \mathbf{t}^t \int_{flows} \mathbf{L}_i \mathbf{L}_j d\mathbf{v} \equiv \mathbf{t}^t \mathbf{C}, \ i, j = 1 \ldots 3 \tag{3.106}$$

with

$$\mathbf{L}_i \equiv \mathbf{M}_i - (\mathbf{De})^t \int_{flows} \mathbf{eM}_i d\mathbf{v},\ i = 1 \ldots 3 \tag{3.107}$$

$$\mathbf{D} \equiv \left(\int_{flows} \mathbf{ee}^t d\mathbf{v} \right)^{-1}. \tag{3.108}$$

For the system of Equations 3.106-3.108, a non-trivial solution can be found for $\mathbf{t}$ by simply finding the eigenvector corresponding to the smallest eigenvalue of the scatter matrix $\mathbf{C}$. The translation $\mathbf{t}$ can then be used to solve for the rotation matrix $\mathbf{R}$ in Equations 3.101, 3.102, and 3.103.

To calculate for and remove the statistical bias which is a result of the bilinear optimization and Equation 3.101, we can assume the noise is isotropic, and infer a statistical bias [Heeger & Jepson 1992]. Given the noise components n_u and n_v present in u and v respectively, the expected values follow $E(n_u) = E(n_v) = 0$ and the variances follow $E(n_u^2) = E(n_v^2) = \sigma^2$. Noise can then be conceptually added to $\mathbf{C}$ to give the noisy $\tilde{\mathbf{C}}$ as

$$E(\tilde{\mathbf{C}}) = E(\mathbf{C}) + \sigma^2 \begin{pmatrix} f & 0 & -fE(x) \\ 0 & f & -fE(y) \\ -fE(x) & -fE(y) & E(x^2 + y^2) \end{pmatrix} = E(\mathbf{C}) + \sigma^2 \mathbf{N}. \tag{3.109}$$

The bias term that should be removed is the matrix $\sigma^2\mathbf{N}$, for which several methods have been proposed. One of the conceptually simplest is the method of Kanatani, which involves estimation of σ^2 and simple subtraction of the bias term [Kanatani 1993]. For computational efficiency and higher accuracy, we will use the method of MacLean to transform the constraints such that the influence of noise is isotropic [MacLean 1999]. Using the eigenvector corresponding to the smallest eigenvalue of $\mathbf{C}$, as before, the noisy matrix $\tilde{\mathbf{C}}$ can be transformed in a method referred to as pre-withening as

$$\check{\mathbf{C}} = \mathbf{N}^{-\frac{1}{2}} \tilde{\mathbf{C}} \mathbf{N}^{-\frac{1}{2}}. \tag{3.110}$$

As an addition, due to the use of $\mathbf{N}^{-\frac{1}{2}}$ as a transformation, the influence of noise on the

transformed scatter matrix $\breve{\mathbf{C}}$ is now only $\sigma^2\mathbf{I}$, where $\mathbf{I}_{3x3}$ is the identity matrix. As $\breve{\mathbf{C}}$ has the same ordering of eigenvectors as $\mathbf{C}$, we use $\breve{\mathbf{C}}$ in place of $\mathbf{C}$ for finding the optimal solution to the system of Equations 3.106-3.108, and then simply scale the chosen eigenvector, multiplying by $\mathbf{N}^{-\frac{1}{2}}$ to obtain the unbiased solution. Care must be taken to ensure that $\mathbf{N}$ does not become negative, as small negative values sometimes appear in large, complex optical flow fields. For this reason, the absolute value followed by the square root, and only then the matrix inversion, are used for numerical calculation in practice.

3.8 Feature Detection

First, feature points must be obtained from a series of images. This is commonly done using the SIFT or SURF keypoint descriptor extractors. However, both SIFT and SURF are patent-encumbered and several other keypoint extractors are now available, most notably the BRISK (Binary Robust Invariant Scalable Keypoints), BRIEF (Binary Robust Independent Elementary Features), and FREAK (Fast Retina Keypoint) feature descriptor extractors. An even more recent development, the ORB (Oriented FAST and Rotated BRIEF) descriptor extractor, adds rotational invariance to BRIEF and is planned for use. ORB has the computational advantage of being based on integer arithmetic, and has proven to perform comparably to or better than the floating-point SIFT and SURF methods in many applications [Rublee *et al.* 2011].

3.8.1 Keypoint Detection

A method of keypoint detection must be used to get the keypoints in the first place. The FAST keypoint detector (Features from Accelerated Segment Test) is frequently used for keypoint detection due to its speed, and is used for quickly eliminating unsuitable matches in ORB. Starting with an image patch p of size $31x31$, each pixel is compared with a Bresenham circle centred on that point (built 45 degrees at a time by $x_{n+1}^2 = x_n^2 - 2y(n) - 1$). The radius of the surrounding circle of points is nominally 3, but is 9 for the ORB descriptor, which expands the patch size and number of points in the descriptor. If at least 75% of the pixels in the circle are contiguous and more than some threshold value above or below the pixel value, a feature is considered to be present [Rosten & Drummond 2005]. The ORB algorithm introduces an orientation measure to FAST by computing corner orientation by intensity centroid, defined as

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \quad where \quad m_{pq} = \sum_{x,y} x^p y^q I(x,y). \tag{3.111}$$

The patch orientation can then be found by $\theta = \text{atan2}(m_{01}, m_{10})$. Since the FAST detector does not produce multi-scale features, a Harris filtered scale pyramid is used to compare several scales of features.

3.8.2 Keypoint Description

The feature descriptor f_n provided by BRIEF is a bit string result of binary intensity tests τ, each of which is defined from the intensity $p(a)$ of a point at a relative to the intensity $p(b)$ at a point at b by [Rosten & Drummond 2005]

$$\tau(p; a, b) = \left\{ \begin{array}{ll} 1 & : p(a) < p(b) \\ 0 & : p(a) \geq p(b) \end{array} \right\} \tag{3.112}$$

and

$$f_n(p) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; a_i, b_i). \tag{3.113}$$

BRIEF descriptors can be referred to as BRIEF-k, where k is the number of bytes needed to store the descriptor. The descriptor is very sensitive to noise, so Gaussian smoothing is applied to the image patch that the descriptor acts on. The more smoothing, the more matches can be obtained. Also, the basic BRIEF descriptor falls in accuracy quickly with rotations of more than approximately 10 degrees. To make BRIEF invariant to in-plane rotation, it is steered according to the orientations computed for the FAST keypoints. The feature set of points (a_i, b_i) in $2xn$ matrix form is rotated by multiplication by the rotation matrix R_f corresponding to the patch orientation θ to obtain the rotated set F[Rublee *et al.* 2011].

$$F = R_f \left(\begin{array}{ccc} a_1 & \cdots & a_n \\ b_1 & \cdots & b_n \end{array} \right). \tag{3.114}$$

The steered BRIEF operator then becomes $g_n(p, \theta) = f_n(p) \vee (a_i, b_i) \in F$. A lookup table of steered BRIEF patterns is constructed from this to speed up computation of steered descriptors in subsequent points. The algorithm is already quite fast, but to additionally decrease the processing time if desired, the camera image can be lowered in resolution, or pixels can be under-sampled by choosing only every 2nd pixel or every 4th pixel in a staggered pattern over the image for feature matching [Ambrosch *et al.* 2009].

3.9 Depth and Structure from Feature Matching

To obtain depth in a 3-D scene, an initial baseline for 3-D projection is first required, which requires the calculation of the Fundamental Matrix **F**, which is a the general 3x4 transformation matrix that maps each point in a first image to another second image. It is generally preferable to use stereoscopic vision for point cloud reconstruction because the baseline can be obtained with two cameras a known distance apart at each location. As a result, the fundamental matrix is constant and can be calculated relatively easily. For monocular vision, the fundamental matrix must be estimated using homographies. In this work, we use a series of images captured with a single camera and some knowledge about the difference between them, but not enough to assume that the fundamental matrix is constant.

3.9.1 Matching

The first step is to match the keypoints with descriptors generated by BRIEF between two images taken from slightly different positions, attempting to find a corresponding keypoint a' in the second image that matches each point a in the first image. Brute-force matching of all combinations of points is the simplest method which generally involves an XOR operation between each descriptor and a population count to obtain the Hamming distance. This is an $O(n^2)$ algorithm, and takes relatively long to complete. However, The FLANN (Fast Library for Approximate Nearest Neighbor) search algorithm built into OpenCV is used in current work as it performs much faster while still providing good matches. [Muja & Lowe 2009].

The more features in common between these images, the more potentially good matches M_f can be found. However, it is vital that these matches be correct correspondences between the images, or a valid transformation between the two images cannot be found. The matches M_f are first coarsely pruned of bad pairings by finding the maximum distance between points d_{max} and then removing all matches that have a coordinate distance d_a of more than half the maximum distance between features.

$$M_g = M_f(a) | d_a < d_{max}/2. \tag{3.115}$$

3.9.2 The Fundamental Matrix

The set of "good" matches M_g is then used to obtain the fundamental matrix for the given scene. The fundamental matrix is the matrix $\mathbf{F}$ that maps every point on the first image to its corresponding location in the second image, based on the assumption of linear geometry between two viewpoints. Consequently, each keypoint a_i in the first image will map to a corresponding keypoint a_i' on the epipolar line (the line of intersection at a_i' of the second image plane with the camera baseline) in the second image by the relation [Luong & Faugeras 1995].

$$a_i'^T \mathbf{F} a_i = 0, \; i = 1, \ldots, n. \tag{3.116}$$

For three-dimensional space, the matrix F has nine unknown coefficients and Equation 3.116 is linear and homogeneous, so F can be uniquely solved for by using eight keypoints with the method of Longuet-Higgins [Longuet-Higgins 1987]. However, image noise and distortion inevitably cause variation in points that make it difficult to obtain a single "correct" $\mathbf{F}$ for all points. Therefore, for practical calculations, a linear estimation method such as linear least squares (i.e. $\min_F \sum_i (a_i'^T \mathbf{F} a_i)^2$) or RANSAC must be used. RANSAC (RANdom SAmple Consensus) is an efficient algorithm designed for robust model fitting that can handle large numbers of outliers, and is commonly used with OpenCV and other algorithms.

RANSAC is a probabilistic solution method that functions as follows [Fischler & Bolles 1981]: Given a fitting problem with a set of m obtained data points and parameters $\mathbf{x}$ for a model $f(\mathbf{x})$ that can be estimated from only $n < m$ data points, the probability of a given randomly-selected data point matching a "good" model is assumed to be $P(good)$ and the probability that the algorithm will finish without finding a good fit among the available data points is $P(fail) = (1 - P(good)^n)^l$, defined as the probability of the algorithm experiencing l consecutive fitting failures. Rearranging the equation for $P(fail)$, the number l can be obtained from

$$l = \frac{logP(fail))}{log1 - P(good)^n)}. \tag{3.117}$$

The algorithm selects n data points randomly from the available m points and estimates the parameters $\mathbf{x}$ from the model. The total set of l data points is then searched to find how many data points fit the model while using the calculated parameters $\mathbf{x}$ within a given tolerance of the maximum, here assumed to be 0.1. If enough data points are found, then the algorithm finishes and returns $\mathbf{x}$ as a valid solution (the $P(good)$ case). Otherwise, the algorithm is repeated up to $4l$ times. If after $4l$ repetitions no valid solution is found, then the algorithm fails (the $P(fail)$ case).

We use RANSAC for its speed to estimate $\mathbf{F}$ for all matches M_g and estimate the associated epipolar lines [Feng & Hung 2003]. Outliers (defined as being keypoints more than the tolerance 0.1 from the estimated epipolar line) are then removed from M_g to yield a final, reliable set of keypoint matches M_h. If no keypoint matches remain by this point, then there are too few features in common between the two images and no triangulation can be created.

3.9.3 The Essential Matrix

To perform a three-dimensional triangulation of points from two-dimensional feature planes and a transformation $\mathbf{F}$ between them, it is necessary to take into account any transformations and projective ambiguity caused by the cameras themselves. Each camera can have a different rotation and translation with respect to an image baseline, so a camera matrix is defined as $\mathbf{C} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$, being composed of the calibration matrix $\mathbf{K}$, the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$. We also need to locate the position of the second camera **C2** in real space with respect to the first camera **C1**. The cameras can be individually calibrated using a known pattern such as a checkerboard [Hartley 1997] [Wang *et al.* 2007], but fairly good results have been achieved by estimating the camera calibration matrix as

$$\mathbf{K} = \begin{pmatrix} s & 0 & w/2 \\ 0 & s & w/2 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.118}$$

For real-world point localization, we can use the essential matrix, defined as $\mathbf{E} = \mathbf{t} \times \mathbf{R}$. Just as $\mathbf{F}$ relates two matching points x and x' in the image plane, $\mathbf{E}$ relates two matching *normalized points* $\hat{x}$ and $\hat{x}'$ in the camera plane as [Hartley & Sturm 1997]

$$\hat{a}_i'^T \mathbf{E} \hat{a}_i = 0, \; i = 1, \ldots, n. \tag{3.119}$$

In this way, $\mathbf{E}$ includes the "essential" assumption of calibrated cameras [Shil 2012b] and consequently has only six unknown coefficients and five degrees of freedom. $\mathbf{F}$ is considered to be a generalization of $\mathbf{E}$, and $\mathbf{F} = \mathbf{E}$ when the camera matrix is the transformational identity matrix ($C = [I|0]$). This leads to a useful relationship between the fundamental and essential matrices, which is

$$\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K}. \tag{3.120}$$

3.9.4 Orientation

After calculating $\mathbf{E}$, we can find the location of the second camera **C2** by assuming for simplicity that the first camera is uncalibrated and located at the origin ($\mathbf{C1} = [I|0]$). We decompose $\mathbf{E} = \mathbf{t} \times \mathbf{R}$ into its component $\mathbf{R}$ and $\mathbf{t}$ matrices by using the singular value decomposition of $\mathbf{E}$ [Hartley & Zisserman 2004]. We start with the orthogonal matrix $\mathbf{W}$ and its transpose $\mathbf{W}^T$, where

$$\mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.121}$$

and the singular value decomposition of $\mathbf{E}$ is defined as

$$\text{SVD}(\mathbf{E}) = \mathbf{U} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{V}. \tag{3.122}$$

The matrix $\mathbf{W}$ does not directly depend on $\mathbf{E}$, but provides a means of factorization for $\mathbf{E}$.

Detailed proofs of the following can be found in [Hartley & Zisserman 2004] and are not reproduced here for brevity, but there are two possible factorizations of $\mathbf{R}$, namely $\mathbf{R} = \mathbf{U}\mathbf{W}^{\mathbf{T}}\mathbf{V}^{\mathbf{T}}$ and $\mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathbf{T}}$, and two possible choices for $\mathbf{t}$, namely $\mathbf{t} = \mathbf{U}(0,0,1)^T$ and $\mathbf{t} = -\mathbf{U}(0,0,1)^T$. Thus when determining the second camera matrix $\mathbf{C2} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$, we have four choices in total. This is because solutions are possible for either direction of the translation vector $\mathbf{t}$ between the cameras, or for a rotation of π radians about the vector $\mathbf{t}$. A given point will be in front of both cameras in only one of these solutions, so all four must be tried in triangulation to ensure the correct solution is found.

3.9.5 Triangulation

Given the essential matrix $\mathbf{E}$, and a pair of matched keypoints $[\mathbf{a} = (a_x, a_y), \mathbf{b} = (b_x, b_y)] \in M_h$, it is now possible to triangulate the original point positions in three dimensions. Just as in estimating the fundamental matrix, image noise and distortion generally prevent perfect solutions from being found [Shil 2012a], so a process of least-squares estimation is used. The algorithm described by Hartley and Sturm [Hartley & Sturm 1997] for iterative linear least-squares triangulation of a set of points is used as it is affine-invariant and performs quite well without excessive computation time. A point in three dimensions $\mathbf{x} = (x_x, x_y, x_z, 1)$ when written in the matrix equation form $\mathbf{A}\mathbf{x} = 0$ results in four linear nonhomogeneous equations in four unknowns for an appropriate choice of $\mathbf{A}_{4x4}$. To solve this, singular value decomposition can again be used, or the method of pseudo-inverses. An alternate method [Shil 2012a] is to simply use $\mathbf{x} = (x_x, x_y, x_z)$ and write the system as $\mathbf{A}\mathbf{x} = \mathbf{B}$, with $\mathbf{A}_{4x3}$ and $\mathbf{B}_{4x1}$ defined as

$$\mathbf{A} = \begin{pmatrix} a_x\mathbf{C1}_{2,0} - \mathbf{C1}_{0,0} & a_x\mathbf{C1}_{2,1} - \mathbf{C1}_{0,1} & a_x\mathbf{C1}_{2,2} - \mathbf{C1}_{0,2} \\ a_y\mathbf{C1}_{2,0} - \mathbf{C1}_{1,0} & a_y\mathbf{C1}_{2,1} - \mathbf{C1}_{1,1} & a_y\mathbf{C1}_{2,2} - \mathbf{C1}_{1,2} \\ b_x\mathbf{C2}_{2,0} - \mathbf{C2}_{0,0} & b_x\mathbf{C2}_{2,1} - \mathbf{C2}_{0,1} & b_x\mathbf{C2}_{2,2} - \mathbf{C2}_{0,2} \\ b_y\mathbf{C2}_{2,0} - \mathbf{C2}_{1,0} & b_y\mathbf{C2}_{2,1} - \mathbf{C2}_{1,1} & b_y\mathbf{C2}_{2,2} - \mathbf{C2}_{1,2} \end{pmatrix} \tag{3.123}$$

and

$$\mathbf{B} = \begin{pmatrix} -a_x\mathbf{C1}_{2,3} - \mathbf{C1}_{0,3} \\ -a_y\mathbf{C1}_{2,3} - \mathbf{C1}_{1,3} \\ -b_x\mathbf{C2}_{2,3} - \mathbf{C2}_{0,3} \\ -b_y\mathbf{C2}_{2,3} - \mathbf{C2}_{1,3}. \end{pmatrix} \tag{3.124}$$

Solution of the resulting system of equations (in this case, using singular value decomposition) yields $\mathbf{x}$, which can be transformed into undistorted "real" coordinates by $\hat{\mathbf{x}} = \mathbf{KC1x}$. This assumes that the point is neither at 0 nor at infinity, which may pose a problem with feature points that lie near infinity in projective reconstructions, but for most structure-from-motion applications it performs fairly well. Again, this triangulation must be performed *four* times, once for each possible combination of $\mathbf{R}$ and $\mathbf{t}$, and each resulting keypoint set checked to verify it lies in front of the camera. This test can be just a simple perspective transformation using $\mathbf{C1}$ and a test to ensure $\hat{\mathbf{x}}_z > 0$. Triangulation of all existing matches will result in an initial point cloud with points p_i.

3.9.6 Pose Estimation

The last step is to find the object pose from the 3D-2D point correspondences and consequently the egomotion of the camera relative to the feature points, commonly know as the Perspective & Point (PnP) problem [Bujnak *et al.* 2011]. Bundle adjustment can also be performed as an additional optimization step on the point cloud to improve the point correspondences once all triangulation is done. Bundle adjustment is not strictly necessary though, and as we desire a fast, probabilistic result, we instead focus on obtaining the projection of the point cloud in space. For this, we apply the OpenCV implementation of the EPnP algorithm [Moreno-Noguer *et al.* 2007] which provides $O(n)$ complexity with the number of point correspondences.

Four control points denoted as c_i are used to identify the world coordinate system of the given reference point cloud with n points $\mathbf{p}_1 \ldots \mathbf{p}_n$, chosen so that one is located at the centroid of the point cloud and the rest are oriented to form a basis with the principal directions of the data. Each reference point is described in world coordinates as a normalized, weighted sum of the control

points with weightings α_{ij} (the w superscript refers to world coordinates). As this coordinate system is consistent across linear transforms, they have the same weighted sum in the camera coordinate system (the c superscript refers to camera coordinates), effectively creating a separate basis [Moreno-Noguer *et al.* 2007]

$$\mathbf{p}_i^w = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^w, \ \mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c, \ \sum_{j=1}^{4} \alpha_{ij} = 1. \tag{3.125}$$

The known two-dimensional projections $\mathbf{u_i}$ of the reference points $\mathbf{p_i}$ can be linked to these weightings with the camera calibration matrix $\mathbf{K}$ considering that the projection involves scalar projective parameters w_i as

$$\mathbf{K}\mathbf{p}_i^c = w_i \begin{pmatrix} \mathbf{u}_i \\ 1 \end{pmatrix} = \mathbf{K} \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c. \tag{3.126}$$

The expansion of this equation has 12 unknown control points and n projective parameters. Two linear equations can be obtained for each reference point, and concatenated together to form a system of the form $\mathbf{Mx} = 0$, where the null space or kernel of the matrix $\mathbf{M_{2nx12}}$ gives the solution $\mathbf{x} = [\mathbf{c}_1^{cT}, \mathbf{c}_2^{cT}, \mathbf{c}_3^{cT}, \mathbf{c}_4^{cT}]$ to the system of equations, which can be expressed as

$$\mathbf{x} = \sum_{i=1}^{m} \beta_i \mathbf{v}_i \tag{3.127}$$

where the set $\mathbf{v_i}$ is composed of the null eigenvectors of the product $\mathbf{M}^T\mathbf{M}$ corresponding to m null singular values of $\mathbf{M}$. The method of solving for the coefficients $\beta_1 \ldots \beta_m$ depends on the size of m. Given perfect data from at least six reference points, m should be 1, but in practice, m can vary from 1 to 4 depending on the camera parameters, reference point locations with respect to the basis, and noise. Hence, four different methods are used in the literature [Moreno-Noguer *et al.* 2007] for practical solution, but the methods are complex and not summarized here.

3.10 Visual Mapping and Localization

The goal of obtaining clouds of feature points from a series of successive images and localizing them with respect to the rover is to obtain a clear picture of what obstacles and terrain lie nearby in the rover's environment, so that better navigation decisions can be made. For interpretation and decision-making based on this information, we will use probabilistic methods as we have in the other major components of this work.

The estimation of vehicle movement from a single camera is known as partially-observable SLAM, as opposed to fully-observable SLAM where a single measurement (e.g. from a stereo vision bench) is sufficient to gather both bearing and range information [Lemaire *et al.* 2007]. As two or more separate observations are necessary to obtain pose estimation, state estimation is delayed by at least one measurement unless the initial state can be estimated somehow, for example by a Gaussian sum filter [Kwok *et al.* 2005]. The bearings-only SLAM problem effectively reduces to a Structure-from-Motion (SfM) problem as images taken from multiple locations must be combined into a single cloud to obtain effective localization, hence the SfM approach to our vision system. The main difference is generally one of implementation trade-offs, as SfM methods are usually batch-based and designed for existing image collections that can be processed slowly and exhaustively (e.g. for photo-tourism [Brahmachari & Sarkar 2011]), while SLAM methods are designed to give up fidelity and scale for fast processing and accurate ranging to nearby obstacles. Generally, SfM methods are implemented to be purely geometric for high accuracy, while SLAM methods are purely probabilistic (usually a particle filter) to maximize speed, with the caveat being that a multiple data source or at least range-aware sensor system (such as LIDAR or a stereo vision processor) must be present [Katrin Pirker & Bischof 2011]. Our method combines aspects of both methodologies to enable SLAM by using only a single camera and minimal vehicle state information, and can operate in real time using modern DSP hardware.

Occupancy grids are frequently used for quantitative mapping applications, and represent one of the most intuitive ways of storing spatially-dependent probabilistic data. The main concerns with a full-sized occupancy map are the amount of storage required, which increases quickly

with spatial resolution and additional dimensions, and the amount of processing required for the entire map, which is proportional to the map size and must often be repeated to estimate position within the map. While many SLAM methods use particle filters for probabilistic comparisons to minimize the computation required for a reasonable estimate, the entire map or at least a large part of it must be used for estimating the vehicle state $\mathbf{x}(t)$ and the map itself $\mathbf{M}$, which can take considerable processing time even for a single estimate. The FastSLAM algorithm uses a Bayesian network approach with a Rao-Blackwellized particle filter to reduce the amount of processing required, but still requires a Kalman filter for each landmark location used in each particle [Montemerlo *et al.* 2002]. Making use of sub-mapping and Bayesian network methods allows the problem to be divided into much more tractable segments, but only provides part of the solution and relies on stochastic models for interpreting data. The geometric triangulation of visible features is relatively fast and accurate and provides local feature and pose identification, but is inherently uncertain and not well suited to repeatable positive identification of features. It is therefore desirable to identify features and relative poses using SfM techniques to generate sparse point clouds, and then use probabilistic techniques based on Bayesian network and BRP principles to map and navigate efficiently. Also, rather than attempting to identify the same features visually on subsequent visits to a location, which is nearly impossible to do reliably in practice without well-known geometry and very complex identification algorithms, probabilistic mapping can be used to approximate the vehicle's position with respect to known obstacles without actually having to re-identify the obstacles themselves. It is also possible to incorporate occlusion information into mapping of sparse point clouds to boost accuracy and identifiability further [Wu *et al.* 2008], and using sparse storage and efficient query methods such as octrees for recording the map can greatly reduce the memory and non-volatile storage required for large maps [Wurm *et al.* 2010], though these additions are not currently incorporated here.

The greatest challenge to be faced with respect to visual SLAM via structure-from-motion methods is that of computational time and memory use. For a complete three-dimensional map that can be traversed and revisited reliably, feature points from all n images previously taken must be accurately localized and cross-referenced with those from the remaining $n - 1$ images,

resulting in an algorithm with factorial complexity ($n!$) that is unreasonable for use in real time, as the more images are collected, the longer a single image will take to process. However, it is not necessary (or even logical) to exhaustively compare all images in question if some knowledge of the relationships between the images is known, such as the general location and bearing of the camera. Given the limited processing power and storage available on the μrover, the simplest possible assumption is the one currently used: that images are taken sequentially over movement, and the image immediately preceding the current one has the closest spatial relationship. This is effectively similar to having stereo vision with an unknown baseline between each pair of images, but as previously stated, the camera matrices can be obtained without prior knowledge of the baseline using SfM techniques..

3.10.1 Sequential Relative Localization

Using only an adjacent pair of images has the advantage that the observed optical flow between the images can serve as a usable localization increment. As baseline triangulation of feature points makes use of only a pair of images at a time, the choice of using an incremental pairwise flow of images makes practical sense. The estimation of ego-motion can be based on either optical flow or SfM methods, and if sufficient processing power is available, both can be used as well. Once the the affine transformation between the two images has been obtained, let the translation and rotation in world coordinates of the prior image be $\mathbf{t}_w(t-1)$ and $\mathbf{R}_w(t-1)$ respectively, and the rotation and rotation of the current (or most recent) image in world coordinates be $\mathbf{t}_w(t)$ and $\mathbf{R}_w(t)$ respectively, for which we need to find the current camera matrix in world coordinates $\mathbf{C}_w(t)$. From each pair of images, we obtain the current affine transformation between the camera positions as a relative translation $\mathbf{t}(t)$ and rotation $\mathbf{R}(t)$. Using the incremental approach, we transform the prior world translation and rotation to the current one with

$$\mathbf{C}_w(t) = [\mathbf{R}_w(t-1)\mathbf{R}(t)|\mathbf{R}(t)\,(\mathbf{t}(t) + \mathbf{t}_w(t-1))]\,. \tag{3.128}$$

Feature points can then be projected into their real locations on the map by the transformation

$$x' = (\mathbf{R}_w(t-1)\mathbf{R}(t))^T x + \mathbf{R}_w(t-1)\left(\mathbf{t}(t) + \mathbf{t}_w(t-1)\right). \tag{3.129}$$

As the μrover's navigation camera is fixed to the front of the body, the translation and rotation of the camera matrix in world coordinates $\mathbf{C}_w(t)$ can be considered to correspond to the pose of the body itself. This also assumes that origin of the camera world coordinate system corresponds to the origin or "start point" of the rover world coordinate system, which is reasonable as long as care is taken to start visual navigation either at the origin of the map or by applying an offset $\mathbf{o}_w$ of the estimated map position at the start of visual navigation to the initial camera matrix. The mapping system should retain this information and the vision system should only initialize itself from the map and then update the map position after each frame estimation. Updating the current position is a matter of converting the current camera frame $\mathbf{C}_w(t) = [\mathbf{R}_w(t)|\mathbf{t}_w(t)]$ into a positional offset, which is simply a scaling to the map coordinates $\mathbf{l}$ of the vector $\mathbf{t}_w$

$$\mathbf{l} = k\mathbf{t}_w(t) + \mathbf{o}_w \tag{3.130}$$

Each point in the point cloud is offset in the same way, so that the relationship is maintained. Orientation is obtained by finding a quaternion rotation $\mathbf{q}$ corresponding to the rotation matrix $\mathbf{R}_w$ by using the elements r_{ij} of $\mathbf{R}_w$.

$$\mathbf{q} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{1+r_{00}+r_{11}+r_{22}}}{2} \\ \frac{r_{21}-r_{12}}{2\sqrt{1+r_{00}+r_{11}+r_{22}}} \\ \frac{r_{02}-r_{20}}{2\sqrt{1+r_{00}+r_{11}+r_{22}}} \\ \frac{r_{10}-r_{01}}{2\sqrt{1+r_{00}+r_{11}+r_{22}}} \end{bmatrix} \tag{3.131}$$

We will denote the location and orientation obtained from optical flow techniques as $\mathbf{l_f}$ and $\mathbf{q_f}$ respectively, and the location and orientation obtained from point clouds as $\mathbf{L_p}$ and $\mathbf{q_p}$ respectively. As they should represent the same movement in three dimensions, both can be used in tandem for movement estimation.

3.10.2 Unusable Feature Sets

The situation of being unable to use a pair of images can occur during the navigation process, and is most commonly due to:

- Not enough feature points being matched (images do not contain enough similar features)
- Inaccurate estimates of rotation **R** and translation **t**. (triangulation of available matches has large error)
- Inaccuracy of the fundamental matrix **F**, preventing decomposition to **E**, **R**, and **t** (available matches do not represent a useable homography)

In most cases, a lack of accurately-matched feature points is the main problem, and is dependent largely on the complexity of the image, the sharpness of focus, and the consistency of lighting and white balance between images. In the event that a new image at time step t in the sequence cannot be used in combination with the prior image $t-1$, the next image in the sequence $t+1$ is used with the reference image at $t-1$, if it fails then the one at $t+2$ is used, and so on until a successful pairing is made. As priority is given to images at the current location, once a successful pairing is made, the current image is used as the prior image for time $t-1$ and the current image at time t is then used with it to continue the process. If a successful pairing cannot be made, then the camera matrix and point cloud are not stored.

3.10.3 Probabilistic Maps for Geometric Data

The basic method for describing an uncertain set of spatial relationships is the stochastic map, proposed early on by Smith, Self, and Cheeseman [Smith *et al.* 1990]. For representation of a map, spatial variables such as the vector $\mathbf{x} = (x_x, x_y, x_z)$ are used with a probability density function (PDF) $P(\mathbf{x})$ that assigns a probability to each combination of x_x, x_y, x_z, thus making the vector $\mathbf{x}$ uncertain but with well-defined statistics in terms of the mean $E(x) = \hat{\mathbf{x}} = (\hat{x_x}, \hat{x_y}, \hat{x_z})$ and covariance $C(x) = E([\mathbf{x} - \hat{\mathbf{x}}][\mathbf{x} - \hat{\mathbf{x}}]^T)$. Variants of this approach are used for most probabilistic SLAM models, but in many cases the relationship of the PDF to the actual sensor data

is only vaguely defined, as "landmarks" are assumed to be found with well-defined reliability by the sensor hardware [Hebert *et al.* 1996]. Most relevant to this work, Pirker et al. developed a methodology for incorporating geometrically-derived point clouds into a three-dimensional probabilistic map while considering a sensor model for point cloud structures and loop closure detection [Katrin Pirker & Bischof 2011].

Using the voxel model for occupancy grid building, each unit in the occupancy map contains a probability of occupancy for an obstacle $p(t)$ at a given time t. When referring to the current point in time, the time step argument t is omitted for clarity. A sensor model is necessary to map sensor readings to appropriate probability values. However, as the input to the probability function is merely a cloud of points with with approximately Gaussian error statistics in the depth estimation obtained by triangulation, we use a simplification of the model proposed by Andert to obtain the probability of occupancy p [Andert 2009]

$$P(z|y,\sigma_y) = p = \begin{bmatrix} e^{-\left(\frac{z-y}{\sqrt{2}\sigma_y}\right)^2} - \frac{1}{2} \end{bmatrix} \frac{k}{\sigma_y} + \frac{1}{2},\ 0 < z \leq y \\ \begin{bmatrix} e^{-\left(\frac{z-y}{\sqrt{2}\sigma_y}\right)^2} + \frac{1}{2} \end{bmatrix} \frac{k}{\sigma_y} + \frac{1}{2},\ z > y \tag{3.132}$$

This probability is stored in odd-log form and updated additively by subtracting the initial belief $o(0)$ as

$$o(t) = o(t-1) + \log(\frac{p(t)}{1-p(t)}) - o(0) \tag{3.133}$$

3.10.4 Filtering of Egomotion Estimates

Estimation of the vehicle's current position and orientation at a given time can be accomplished by using the optical flow and SfM ego-motion measurements for evaluating the posterior distribution $P(\mathbf{x}|\mathbf{l_f}, \mathbf{q_f}\mathbf{l_p}, \mathbf{q_p})$. For closed-form implementation, rather than an ideal Bayesian filter, an approximation generally must be used such as a Kalman filter. For filtering of position-only measurement, an unscented Kalman filter was implemented as described in Section 2.11.

In practice, when navigating in an environment with few unique features and many false fea-

ture point matches such as a rocky or desert area, triangulation of similar points between multiple frames actually decreases the accuracy of point triangulation due to the number of matched feature points that are incorrectly thought to represent the same point in space. Two-frame optical flow has performed relatively well in comparison. Therefore, ego-motion estimation from optical flow is primarily used to identify the camera motion between frames, while feature point triangulation is used to locate obstacles and build a map based on the ego-motion estimation.

Chapter 4

Results and Discussion

Testing was done with the μrover and many of its component parts. The results of testing the hardware, software, and algorithms used on the μrover are detailed in this chapter, including environmental testing, filtering, electronic calibration, and navigational tests. Key results include the selection and characterization of onboard sensors and power systems, the evaluation of fixed-point math operations and Kalman filtering, the performance of different real-time software platforms, the use of the Bayesian-network based control system for obstacle avoidance and navigation, and the operation of the sequential monocular vision system for simultaneous localization and mapping, as well as qualification of the design for space conditions..

Table 4.1: Beaver μrover performance specifications

Specification	Quantity	Unit
Maximum safe ground incline about roll axis	40	degrees
Maximum safe ground incline about pitch axis	45	degrees
Forward driving force (4 wheels)	320	N
Forward driving force (2 wheels)	160	N
Vertical lift force (4 wheels)	270	N
Vertical lift force (2 wheels)	100	N
Maximum ground clearance (raised)	15	cm
Minimum ground clearance (lowered)	4	cm
Maximum vertical obstacle climb height	5	cm
Maximum sudden cliff drop height	10	cm
Obstacle detection range (IR sensors)	20-200	cm
Obstacle detection range (vision)	30-1200	cm
Cliff detection range (vision)	30-100	cm

4.1 Mechanics

4.1.1 Performance Specifications

Based on the design of the μrover that has been described, a summary of performance specifications determined through testing is provided in Table 4.1.

4.1.2 Mass Budget

The mass budget for the μrover components is given in Table 4.2. A leftover mass of $1kg$ is allotted to each of the payloads front and back and their associated hardware. Contingency percentages are also used to allow for some variation in the final mass of each component. The total mass of the μrover is designed to be 6kg in total with payloads.

Table 4.2: Beaver μrover mass budget

Part Name	Num Units	Unit Mass (g)	Raw Mass (g)	Contingency (%)	Predicted Mass (g)
OBC Board	1	73	73	1.00%	73.73
Drive Board	1	160	160	1.00%	161.6
Sensor Board	1	68	68	1.00%	68.68
Camera	1	15	15	1.00%	15.15
Radio Module	1	45	45	5.00%	47.25
Antenna	1	29	29	5.00%	30.45
IR Sensors	6	12	72	1.00%	72.72
Batteries	6	24	144	5.00%	151.2
Main Solar Array	1	142	142	5.00%	149.1
Folding Solar Array	2	108	216	5.00%	226.8
Folding Array Support	2	14	28	10.00%	30.8
Electronics Enclosure	1	370	370	5.00%	388.5
Enclosure End	2	20	40	5.00%	42
Enclosure Hardware	1	20	20	10.00%	22
Chassis Frame	1	420	420	5.00%	441
Chassis Wiring	1	10	10	10.00%	11
Spring Dampers	4	27	108	10.00%	118.8
Deployment Spring	4	18	72	10.00%	79.2
Suspension Bearing	4	7	28	10.00%	30.8
Suspension Arm	4	86	344	10.00%	378.4
Suspension Hardware	1	28	28	10.00%	30.8
Drive Bearing	8	6	48	10.00%	52.8
Drive Motor	4	136	544	10.00%	598.4
Mitre Gear	8	5	40	10.00%	44
Wheel	4	120	480	10.00%	528
Payload Allowance	2	1000	2000	10.00%	2200
Total:		2963	5544		5993.18

4.2 Electronics

4.2.1 DC-DC Converter Selection

To determine the best choice of COTS voltage converter for use on the μrover, a survey of DC-DC converter hardware was conducted by obtaining samples of several ICs that fit required criteria and building the recommended circuit given on the datasheet for each of them to produce $3.3V$ (buck converters), $5V$ (boost converters), or $4.2V$ (battery chargers). Circuit testing was done on solderless protoboards, which are known to serve as a poor substitute for PCBs particularly at high switching frequencies, but this was considered to be a valid environmental limitation to prove that the ICs could still operate under highly adverse conditions. Most ICs tested were from Maxim and Linear Technology due to the high integration and low number of external components common to their parts, but ICs from other manufacturers were considered as well. The criteria for IC selection were as follows:

- Available in SOIC, SOT, or other hand-solderable SMT packages
- Able to produce the required voltage through internal external resistor dividers
- Low complexity in circuit, leading to lower chance of failure
- Minimal possibility of shoot-through or shorting
- External MOSFET switching preferred for better heatsinking
- Minimum number of additional external parts

As the tested parts performed very differently from each other, A qualitative comparison was considered to be sufficient for selection, given in Table 4.3. From this testing, the MAX1626 step-down converter IC was chosen as the best option for producing $3.3V$ from the $3.7V$ supply, and it has performed very well in the different modules of the μrover under outdoor and thermal vacuum testing with a quiescent current of only $100\mu A$. An exception is the powering of the

AT91RM9200 microcontroller on the OBC, as it requires an additional 1.8V supply and is powered by an LTC3407 dual-output supply, which is small and performs well but is difficult to hand solder. The MAX1736 is the only pulse-charging IC for Li-Ion batteries, but integrates PWM charge control into a SOT23 package and provides very high efficiencies, and potential currents when compared to linear regulated chargers such as the MAX1811, with the limitation that the voltage source must be limited as it is in the case of solar charging. Some parts are capable of being used in both buck and boost configurations, such as the LTC1111 and 1173. The MAX608 with a quiescent current of $200\mu A$ was used successfully to produce $5V$ for USB in the prototype μrover until the Li-Ion battery stack was implemented.

Table 4.3: Comparison of DC-DC converter ICs for use in the μrover

Part Name	Type	Current	Appr. Efficiency	Comments from Testing
BQ2057	Charger	1A	Linear	Charges fast, but doesn't appear to do any regulation
LTC1513	Charger	3A	60%	SEPIC, step up/down, high rate, complex
LTC4054	Charger	250mA	Linear	low dropout, but thermally current limited
MAX1555	Charger	250mA	Linear	internally regulated
MAX1736	Charger	3A	95%	can charge at high rates, good input range
MAX1811	Charger	500mA	Linear	has soft-start but narrow input range
MIC79050	Charger	500mA	Linear	simple, const rate, low dropout
LTC1147	3.3V Buck	1A	75%-85%	DIP pkg, Good regulation, dropout at 0.4V
LTC1474	3.3V Buck	200mA	88%	very efficient, but 0.4V dropout at 3.7V
LTC1530	3.3V Buck	-	-	complex to use, prototype did not function
LTC1627	3.3V Buck	Ĩ0mA	40%	prototype provided barely any current
LTC1771	3.3V Buck	3A	90%	very good regulation and efficiency, Very low dropout
LTC3407	3.3V Buck	800mA	92%	provides both 1.8 & 3.3V, SMT-only, nearly no dropout
MAX710	3.3V Buck	500mA	60%	good regulation, not efficient, hot at 400mA and up
MAX1626	3.3V Buck	2A	90%	$I_{in} = I_{out}$, low dropout (0.10V), Using MTB50P03 FET
MAX1627	3.3V Buck	2A	90%	$I_{in} = I_{out}$, good regulation low dropout (0.10V)
LM3671	5V Boost	600mA	60%	lower efficiency, not well-regulated, low dropout (0.15V)
LTC1111/1173	5V Boost	300mA	80%	DIP pkg, buck/boost, Good regulation, medium dropout
LTC1515	5V Boost	100mA	65%	can do 25mA to 8.4V from 4.7V (use ¡100kohm FB)
MAX608	5V Boost	1.5A	83%	Excellent regulation at high I, efficient, Using MTP3055 FET
MAX682	5V Boost	250mA	50%-90%	output I=input I/2, good efficiency at low voltages
MAX710	5V Boost	500mA	70%	buck/boost, good regulation, hot at 400mA and up
MAX770	5V Boost	1A	60%	very bad regulation, low efficiency, shorted often in testing
MAX1703	5V Boost	-	-	IC failed in prototype circuit after only moderate loading

4.2.2 Motor Drive Testing

The four-channel hybrid drive motor controller designed for the μrover prototype was constructed and connected to the BLWRPG093S-12V-3500-R139 drive motors. For programmed control, a set of embedded motor control functions were added to the Atmel AVR microcontroller software library written for the μrover and implemented in a program for the ATMega1281 microcontroller that controls the half-bridges on the drive board. In order to drive brushless motors, commutation of the motor coils must be very fast and reliable, so the commutation algorithm in software for switching the half-bridges was implemented as a look-up table. This is shown in Table 4.4. The entries prefixed by $0b$ in this table are numbers given in binary to more clearly indicate the bit-wise operation of the microcontroller pins that are set using the table.

The table is used as follows: the three-bit input from the three-phase Hall sensors on the motor are used to determine the position of the rotor by directly indexing the rows of the table as a numerical index from 1 to 6. This value is shown in the "Hall Output" column. The binary entry in the "Enables" column then is used to directly set three adjacent pins on the microcontroller to control turning each of the three half-bridges on or off, and the binary entry in the "High/Low" column is used to directly set another three adjacent pins on the microcontroller that set whether each half-bridge connects the end of a coil to the supply voltage (high) or ground (low). For each step of rotation of the BLDC motor, there are two choices for selecting high and low side for each of the three half-bridges depending on whether clockwise (CW, negative) or counter-clockwise (CCW, positive) rotation is desired. For clockwise rotation, the "CW High/Low" column is used, while for counter-clockwise rotation, the "CCW High/Low" column is used. Note that the settings of the enable signals are the same for either clockwise or counter-clockwise rotation due to the fact that rotational direction is determined by the direction that current flows in the active motor coil, but separate columns in the table are used to increase flexibility in future implementations and because there is no speed performance decrease by using a slightly larger table. Turning the motor controllers off is done by indexing the table with 0 (in which case High/Low selection doesn't matter), and braking the motors by connecting both terminals of all phases to the high side is done by indexing the table with $0b111$, as neither value can be encountered during Hall

Table 4.4: Commutation Output Look-Up Table for Hybrid Motor Drive Brushless DC Operation

Description	Hall Output	CW Enables	CW High/Low	CCW Enables	CCW High/Low
Off:	$0b000$	$0b000$	$0b000$	$0b000$	$0b111$
Step 0:	$0b001$	$0b011$	$0b010$	$0b011$	$0b001$
Step 2:	$0b010$	$0b110$	$0b100$	$0b110$	$0b010$
Step 1:	$0b011$	$0b101$	$0b100$	$0b101$	$0b001$
Step 4:	$0b100$	$0b101$	$0b001$	$0b101$	$0b100$
Step 5:	$0b101$	$0b110$	$0b010$	$0b110$	$0b100$
Step 3:	$0b110$	$0b011$	$0b001$	$0b011$	$0b010$
Brake:	$0b111$	$0b111$	$0b000$	$0b111$	$0b111$

sensor operation. Starting the motor involves several periods of auto-commutation, in which the table is stepped through automatically at low speed in the same sequence as the Hall sensors would provide until sufficient rotational momentum is built up that the sensors can be used to determine the next rotor position. This is done automatically when the control program identifies that the half-bridge outputs are on but no movement is occurring.

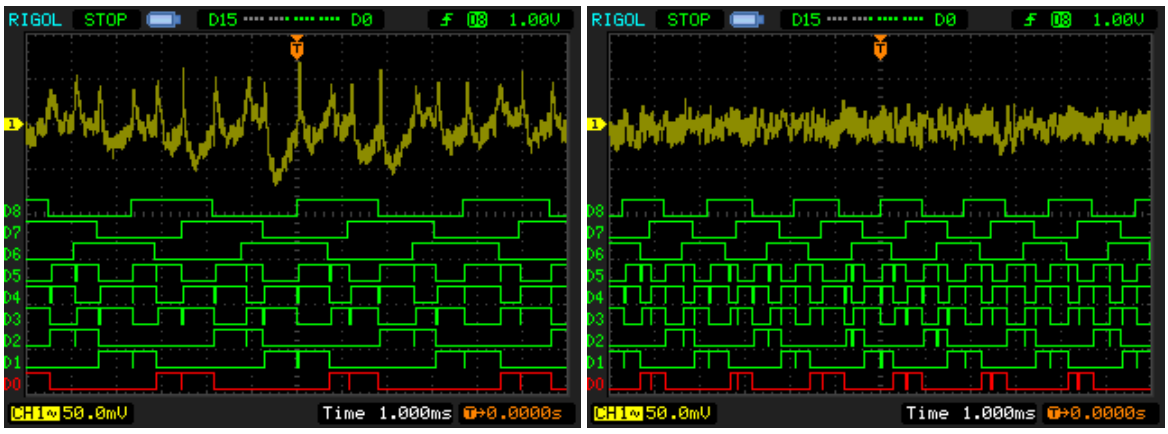
For efficient operation, commutation of the motor must be very fast to keep pace with the rotation of the motor. Commutation table lookups and control pin settings are interrupt-driven to ensure that no commutation steps are missed. A cascade of three exclusive-OR gates is used to multiplex all three of the hall sensor outputs on a motor to a single interrupt pin, as not enough interrupt pins are available on the ATMega1281 to separately connect all Hall sensors. The table indexing operation is very fast, taking only a few clock cycles on the microcontroller to complete, and by making use of the interrupt priority sequencing on the AVR, interrupt-driven commutation of four motors simultaneously by a single microcontroller core running at $14.7456MHz$ has proven to be unexpectedly reliable. The main limitation of this efficient direct-indexing method is that the enable and side selection pins for the half-bridges must both be on three adjacent and sequential pins on a port, which is not usually a problem on large microcontrollers. Speed control is accomplished by setting a timer/counter on the microcontroller to limit the amount of time that the half-bridges are allowed to remain on at each commutation step, synchronously limiting the amount of total energy allowed into a coil. Although a small number of extra microcontroller clock cycles are required to set the timer, it does not noticeably affect the reliability of the motor.

Table 4.5: Encoder Increment Look-Up Table for Hybrid Motor Drive Brush DC Operation

Current State:	$0b00$	$0b01$	$0b10$	$0b11$
Last state $0b00$:	0	-1	1	0
Last state $0b01$:	1	0	0	-1
Last state $0b10$:	-1	0	0	1
Last state $0b11$:	0	1	-1	0

To make the hybrid drive symmetric in operation when used for brush DC motors, the same structure is used for brush DC operation, as shown in Table 4.5. However, brush DC motors are not usually equipped with three-phase Hall sensors, but instead use conventional rotary encoders that typically use a two-bit Gray code scheme where successive increments differ by only one bit. The bit that is changed determines the direction of rotation. So the table is indexed directly from the two-bit encoder input for the motor with the row selected by the previous binary state, and the column selected by the current binary state. Rather than providing the ordering of bit outputs directly as would be needed for commutation, the table provides the increment or decrement in position used for determining the speed of the motor. An increment indicates counter-clockwise (right-handed) rotation and a decrement indicates clockwise rotation. The increments and decrements are added up between controller software updates, an operation that is again driven by interrupts and takes very few cycles to complete, and then divided by the elapsed time between updates to determine the motor speed, which is then used as feedback for the control algorithm.

For electronic testing of the software-commutated motor controller, a single unloaded brushless DC motor was connected to one channel (a set of three half-bridges) on the controller, and an oscilloscope with logic analyzer (a Rigol DS1052D) was connected to the controller outputs and used to probe both the drive voltage and half-bridge control signals. Figure 4.1 shows the resulting traces. The analog channel 1 in the top half of the plot shows the voltage at the "A" terminal of the motor, which is indicative of all three motor terminals as the windings and drive topology are symmetric. The topmost digital signals D6, D7, and D8 show the outputs from the three-phase Hall sensors on the motor that overlap in commutation intervals as they should, the next three D3, D4, and D5 show the enable signals for the three half-bridges where always two out of the three bridges must be enabled, and the last three D0, D1, and D2 show the side select



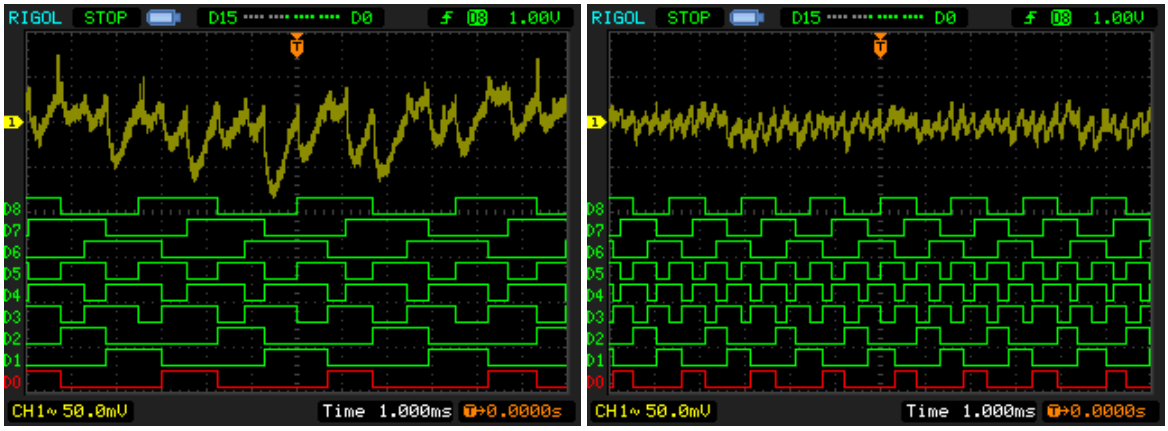
(a) 6V Operation (b) 12V Operation

Figure 4.1: Control outputs for BLDC motor with software commutation at 6V (a) and 12V (b)

signals, such that one of the two enabled half-bridges must be set to high side and the other must be set to low side at any given time. It is relatively simple to see that the digital signals follow the same pattern as the binary entries in the commutation table 4.4. Two tests were performed to determine the difference between $6V$ (half nominal) and $12V$ (nominal) input voltage for the motor.

For comparison of the software-commutated motor controller with purely hardware commutation methods, the logic circuit shown in Figure 2.34 was implemented using discrete logic ICs on a separate prototyping board with a separate set of three half-bridges to run the motor. The same configuration of enable and side select pins was probed by oscilloscope so that the operation of the hardware and software based drivers could be compared. Figure 4.2 shows the results from this test, in which the same motor, rotation direction, test conditions and oscilloscope trace connections are used as in Figure 4.1, and Figure 4.3 shows the testing setup for the hardware-commutated motor controller.

The motor speed observed at $6V$ is slightly more than half of the motor speed at $12V$, indicating that the scaling of speed with input voltage at nearly constant output torque is approximately linear, but not completely due to the effects of small changes in dynamic friction at different motor speeds and nonlinear dynamics within the motor coils. In all tests, the brushless DC motor operated as designed. The most significant difference between the software-commutated and



(a) 6V Operation (b) 12V Operation

Figure 4.2: Control outputs for BLDC motor with hardware commutation at 6V (a) and 12V (b)

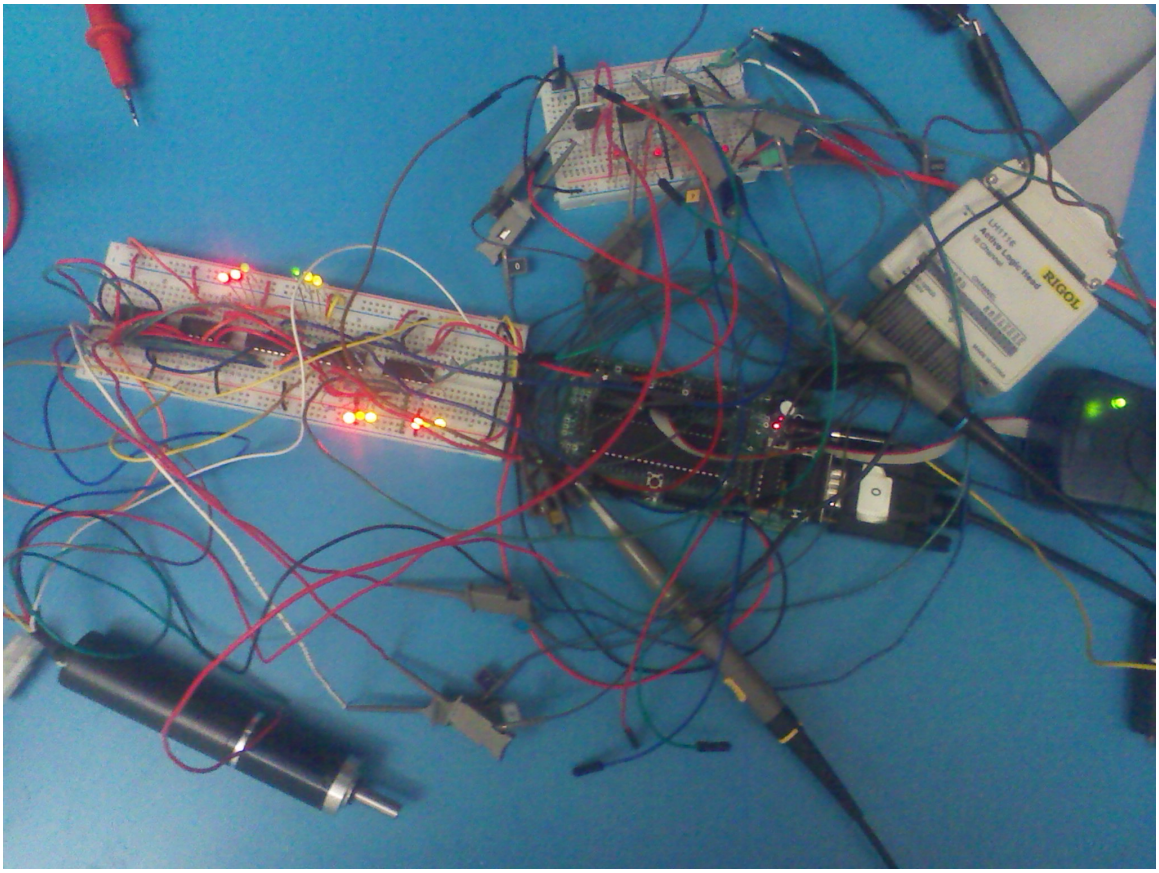


Figure 4.3: Testing of brushless DC control with hardware commutation

hardware-commutated configurations was the regular appearance of small zero transients in the pin outputs, which can be seen when the enable and side select outputs are high. No software explanation could be found for these transients, though they are only microseconds in length and therefore do not have a significant impact on controller operation due to limitations on the switching speed of the half-bridges. The timing and amplitude of the analog motor voltage is less regular, indicating that switching speeds are not precisely the same in every commutation cycle, which is to be expected in a limited-frequency clocked system. In contrast, the hardware-commutated controller provides extremely reliable and clean timing as it has neither a clock nor significant latency in its response to the Hall sensors. Higher speed is observed in both of the hardware-commutated tests relative to the software-commutated tests because the faster and cleaner switching times of the discrete logic ICs decreases phase lag on the motor bridges and increases overall efficiency.

However, the major drawbacks of the hardware-commutated configuration are that it cannot easily support brush DC motors, does not auto-start, and there is no speed control. An external microcontroller with PWM capability or digitally-controllable PWM or DAC chip would be required for DC motor operation in any case. Starting the hardware-commutated motor requires turning the rotor externally to build up momentum before commutation can occur. This requires a change in the commutation logic and significantly complicates the design of the hardware. Speed control could potentially be accomplished simply by introducing an external pulse-width modulated (PWM) signal into the enable pin logic. The problem is that the switching speed of the half-bridges is not greatly higher than the needed commutation speed and aliasing occurs at useable frequencies, such that a different number of PWM pulses may occur on each commutation step, greatly decreasing efficiency and causing unreliable operation, so speed control must be synchronous with commutation, again complicating the design. Due to its flexibility, operational capabilities, and the smaller form factor of not requiring external discrete logic, a software-commutated hybrid motor driver was ultimately used for the μrover motor controller, and has performed very reliably and consistently overall.

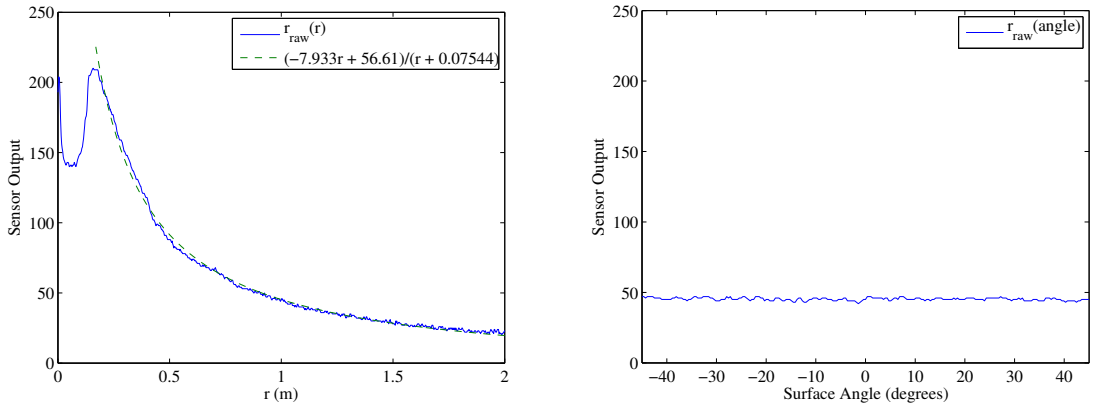


Figure 4.4: Infrared Range Sensor Profiles for Distance (left) and Surface Angle (right)

4.2.3 Infrared Range Sensor Characterization

To evaluate the performance of the μrover infrared range sensors used for navigation, the μrover was placed at $2m$ from one of the mapping obstacles and driven forward while taking positional measurements. By driving the infrared emitter with its maximum design voltage of $7V$, a maximum range of $r_{max} = 2m$ is possible, which is assumed by the sensor model. Additionally, to ensure that the range sensor would function at oblique angles to a target, a mapping obstacle was placed at $1m$ distance and the normal of the facing surface rotated through $\vartheta_o = (45° \ldots -45°)$ with respect to the line of sight of the sensor. Profiles of digital range r_{raw} versus actual range and digital range with respect to the facing surface angle of the target ϑ_o are plotted in Figure 4.4.

An 8-bit ADC is used for measurement of the range sensor, and the sensor output noise level remains within ±1 8-bit unit throughout most of the test. Fitting the profile of r_{raw} to a first-order rational function yields the polynomial fit $r_{raw} \approx (-7.933r + 56.61)/(r + 0.07544)$, which is solved for r to obtain the transfer function $r = -(0.08(943r_{raw} - 707625))/(1000r_{raw} + 7933)$ used to calculate the actual range from the given measurements. The combination of sensor, ADC, and polynomial fit noise results in an RMS error of $\sigma_r = 2.777$ 8-bit units (1.33%) in r_{raw}, which is more than acceptable for the main driver of uncertainty in the map, our rover positioning error. Variation with target surface angle of $r_{raw}(\vartheta_o)$ is also very low, showing no appreciable dependence on ϑ_o and only slightly higher noise than with no rotation. As there is no fitting

estimation, an RMS error of 1.015 is observed for this case.

In practice, infrared range sensors such as these are limited in capability by ambient sunlight. Strong direct sunlight can overwhelm the intensity of the infrared beam generated from the sensor and make it almost impossible for the sensor to distinguish the location of the reflected beam on nearby objects. For this reason, the current models of infrared sensor are considered to be low-cost development and testing alternatives to more robust range sensors such as laser rangefinders or LIDAR systems that would be used on actual flight hardware. Range sensors used as flight hardware would have a number of additional performance factors to consider with respect to the environment of Mars and the specific conditions expected, ambient sunlight being chief among these. Similarly-designed optical rangefinders could potentially be used in daylight on a planetary mission if the central electromagnetic frequency of the optical emitter is sufficiently different from that of the ambient light expected at the mission site. While the broad spectrum of sunlight that passes through a thin atmosphere such as that on Mars makes this difficult despite lower peak intensity from being further from the sun [Edmondson *et al.* 2005], intense low-infrared or ultraviolet illumination could provide feasible detection. Another important consideration is surface reflectivity. Knowing which frequencies of electromagnetic radiation are most highly reflected from expected obstacles, chiefly rocks, allows an appropriate choice of frequency for the sensor. This also involves an understanding of the diffuse coefficient of reflection of the rock surface at that frequency to ensure that obstacles can be detected at oblique angles and will not specularly reflect away incident beams. Taking these considerations into account, selecting a very narrow band-pass filter for the receiving sensor or using a photodetector with very a small frequency response range will also increase the accuracy of the sensor by eliminating ambient light that has not been cast by the emitter. Accurately obtaining the peak location of illumination for a triangulation-based sensor or the peak time for a time-of-flight-based sensor in a similar manner to centroiding on an array-based sun sensor will then provide accurate range estimation. Naturally, such sensors must also not be placed such that their sensing angles overlap or the light cast by each may interfere with adjacent sensors.

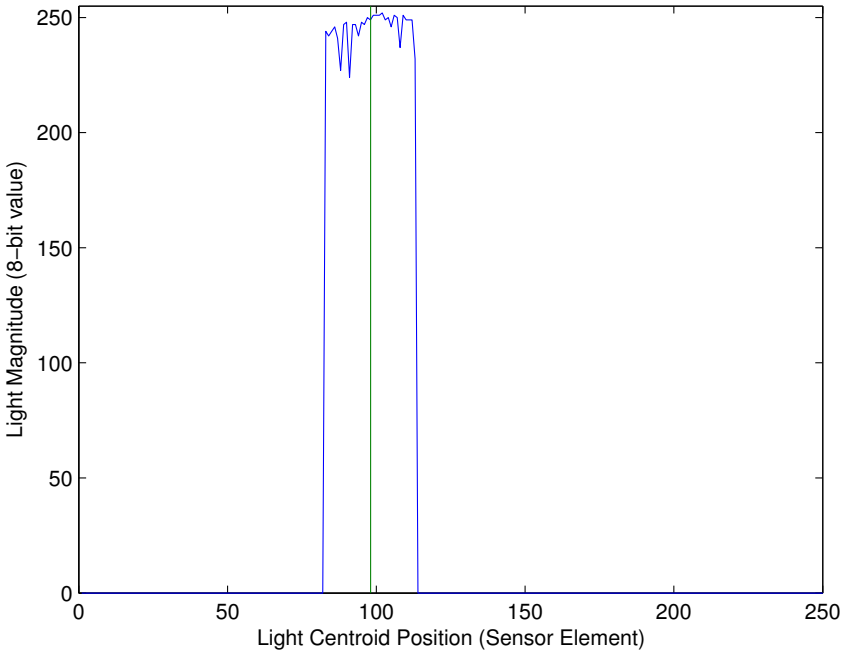


Figure 4.5: Example of Linear Array Output

4.2.4 Sun Sensor Characterization

The customized sun sensing methods for aiding in μrover navigation was tested thoroughly in the laboratory environment and briefly on a sunny day outdoors to verify that operation was as expected and sensor outputs were within reasonable error bounds.

4.2.4.1 Photodiode Array Sensing

Typical illumination measurements from the photodiode array sun sensor are shown in Figure 4.5.

The estimated, centroided solar angle θ is plotted in Figure 4.6, with the actual solar angle used on the gimbal test apparatus shown as a straight line for comparison. In general, very good agreement is achieved between the estimated and actual solar angles. It is worth noting that this angular data is not filtered or smoothed, but still achieves consistent tracking.

If an N-slit is used to perform estimation of the transverse angle ϕ, multiple illumination

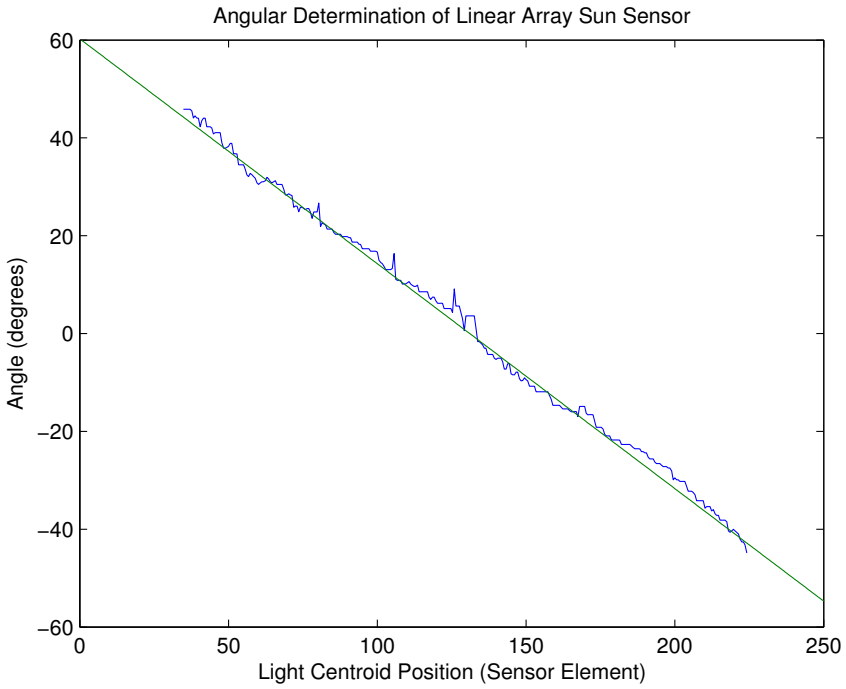


Figure 4.6: Sun Angle θ Results from Linear Array for 60° to −60°

centroids are present on the sensor as shown in Figure 4.7, and must be partitioned separately. However, as the angle ϕ increases from the vertical, it is common for one of the illuminated regions to move off the sensor for moderately high angles of θ, and the other two illuminated regions to approach and ultimately merge together as seen in Figure 4.8. This poses a problem for centroid partitioning as angle increases. To mitigate this problem, the outer boundaries of the illuminated areas are tracked, and the centroids are constrained to be between these boundaries and the midpoint between the boundaries. Having a precise geometry for the N-slit is essential for accuracy in this case, and some calibration must be performed for the alignment of the sensor and N-slit, particularly for the angle δ that the side slits make with the center slit. Using the centroid from the center slit located at position $d2$ and the centroid of one of the side slits at position $d1$, the angle ϕ can be estimated by using [M.-S. Wei & You 2011]

$$\phi_n = \text{atan}(\frac{d2 - d1}{h \tan(\delta)}). \tag{4.1}$$

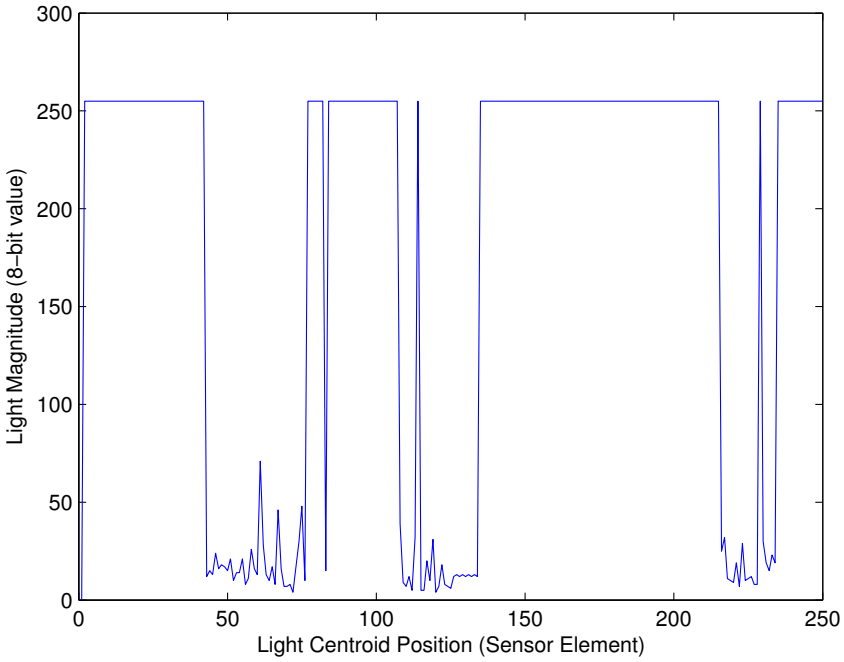


Figure 4.7: Example of N-Slit Output at low θ angles

The estimated transverse solar angle ϕ obtained from N-slit measurements by using Equation 4.1 is shown in Figure 4.9. Reasonable agreement is obtained, but the error is higher than in simple linear slit measurements due to the greater complexity of constructing a precise N-slit. Accurate measurements of angles above approximately 35° could not be obtained due to difficulties accurately partitioning and centroiding the illuminated areas on the sensor at high angles of ϕ. The use of a wider N-slit or thinner material and slit width can be used to extend the measurable angles of this sensor.

Using both the angles θ and ϕ and precalculated solar ephemeris data, a test of calculating the heading of the μrover was conducted. The sun sensor was mounted on the μrover with the orientation aligned with the body frame as stated above, and the μrover rotated from 165° to 135° away from north. The estimated heading angle for the 30° sweep is plotted in Figure 4.10, with the externally measured angle superimposed for reference. To improve the accuracy of this measurement, a 2-point moving average was used in processing.

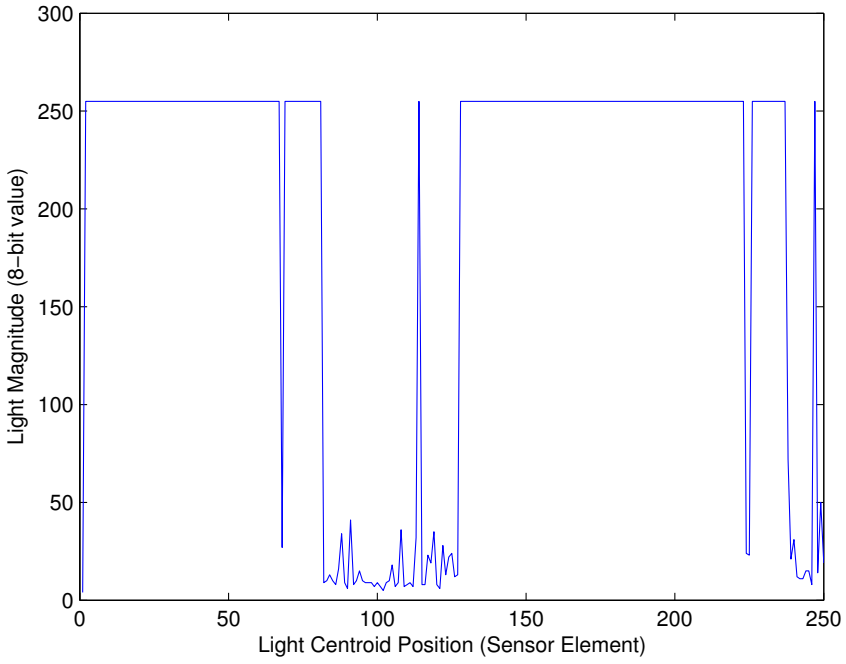


Figure 4.8: Example of N-Slit Output at high θ angles

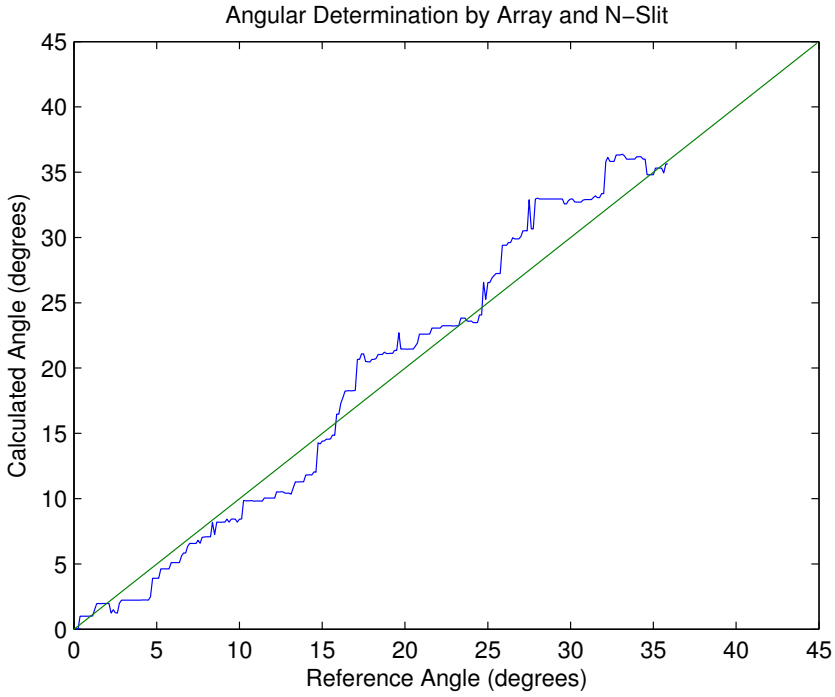


Figure 4.9: Transverse Sun Angle ϕ Results from N-Slit for 0° to 45°

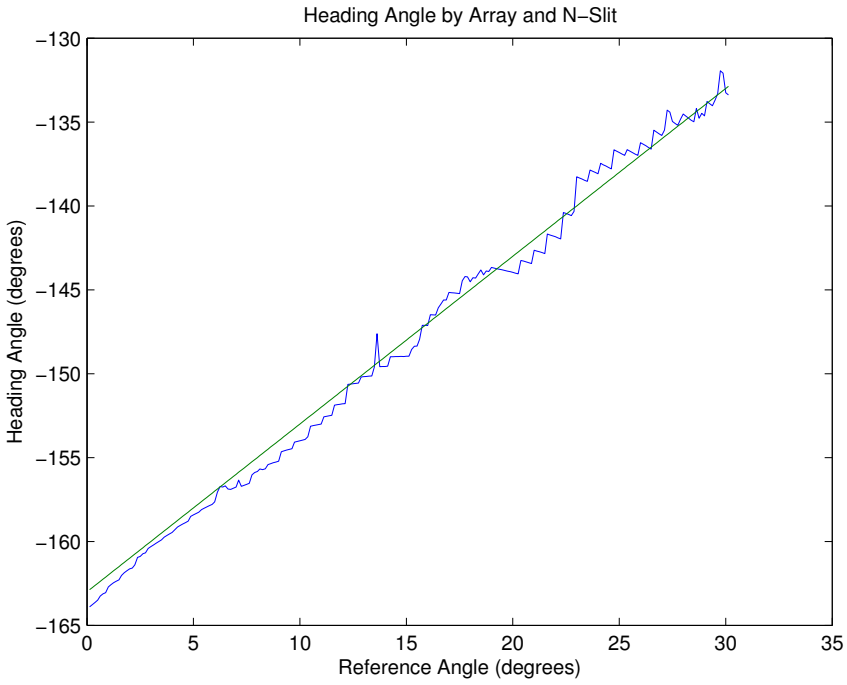


Figure 4.10: Estimated Micro-Rover Heading Across 30° of Rotation

4.2.4.2 Solar Current Sensing

Current measurement using high gains and ADC sensing generates much more noise than digital sensing using a linear array. Although a constant current draw and capacitive decoupling of the amplifiers and microcontroller pins was used in this study, applying a windowed average to the data assuming slow changes in angle was necessary to achieve consistent results. Figure 4.11 shows the solar panel current output I_{n-1}, I_n, and I_{n+1} for three solar panels enumerated as $n-1$ (facing the +Y axis), n (facing the +X axis), and $n+1$ (facing the -Y axis) in the direction of increasing angle about a 1U CubeSat. Smooth reference curves for the actual solar angles presented in gimbal testing with respect to each panel $\cos(\theta_{n-1})$, $\cos(\theta_n)$, and $\cos(\theta_{n+1})$ are superimposed for reference.

After filtering the current I_n from each panel n, the quadrant that actual solar angle lies in with respect to the satellite body must be determined. The most straightforward method of doing this is to simply identify which panels are exposed to the most sunlight by comparing the relative solar

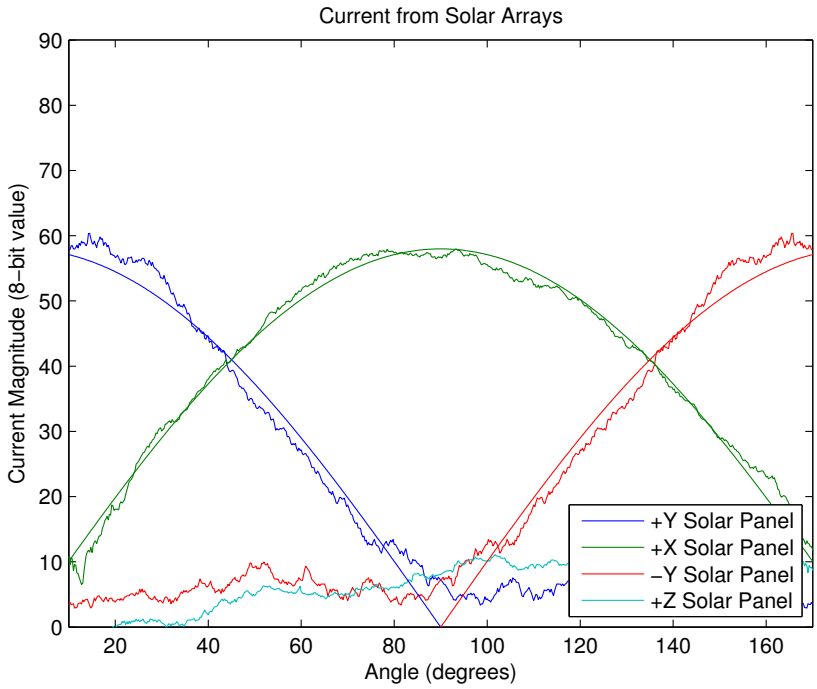


Figure 4.11: Solar Panel Current Measurements for −180° to 180°

panel currents and assigning the appropriate sinusoid quadrant function using mapping functions. As the greatest change in illumination is present at high solar angles to each panel, it is possible to determine the quadrant of a sine function for the satellite body frame angle θ_b by using only I_{n-1} and I_{n+1} to determine the mapping for only the current I_n.

$$\begin{aligned} I_{n-1} > I_{n+1} &\Rightarrow \theta_b = \text{asin}(I_n/max(I_n)) \\ I_{n-1} > I_{n+1} &\Rightarrow \theta_b = \text{asin}(-I_n/max(I_n)) + \pi/2 \end{aligned} \tag{4.2}$$

Using Equation 4.2, the current from each solar panel is used to obtain an estimated body frame angle θ_b over a 180° arc, shown in Figure 4.12. It is evident that there is a discontinuity and higher inaccuracy near the angle $\theta_b = 90°$ which corresponds to $\theta_n = 0$. This is due to the sudden jump in assignment, but also to the inaccuracy determining angles close to the vertical. To mitigate this, a revised mapping given in Equation 4.3 can be used that takes advantage of the

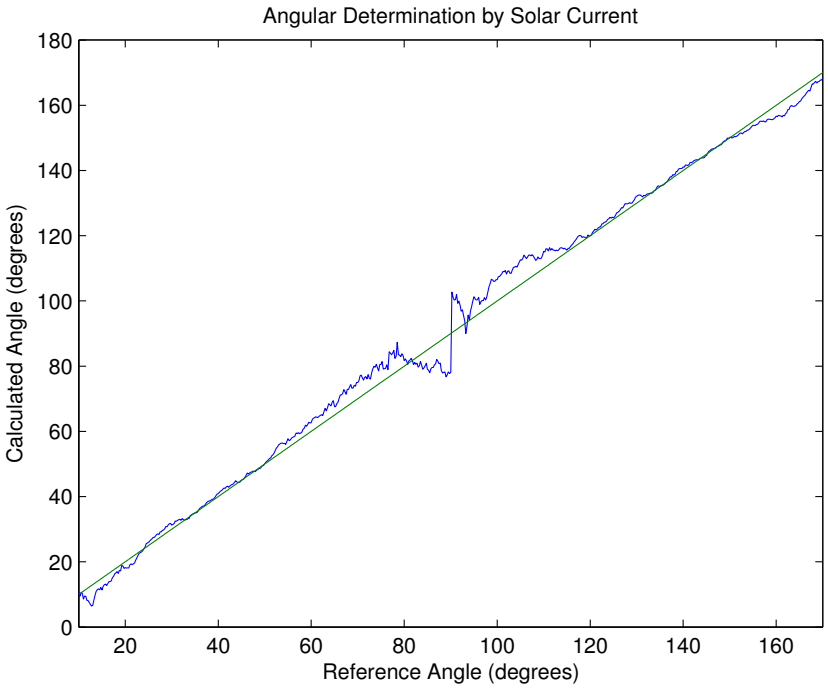


Figure 4.12: Angle from Single Solar Panel Current for 0° to 180°

other solar panels' contributions at high angles to increase the accuracy of measurement. The estimated body frame angle θ_b for this case is shown in Figure 4.13.

$$
\begin{aligned}
I_{n-1} > I_n > I_{n+1} &\Rightarrow \theta_b = \text{asin}(I_n/max(I_n)) \\
I_n > I_{n-1} > I_{n+1} &\Rightarrow \theta_b = \text{asin}(-I_{n-1}/max(I_{n-1})) + \pi/4 \\
I_n > I_{n+1} > I_{n-1} &\Rightarrow \theta_b = \text{asin}(I_{n+1}/max(I_{n+1})) + \pi/4 \\
I_{n+1} > I_n > I_{n-1} &\Rightarrow \theta_b = \text{asin}(-I_n/max(I_n)) + \pi/2
\end{aligned} \tag{4.3}
$$

It should be noted that as the μrover receives most of its power from vertically-oriented solar panels, Equation 4.2 is more appropriate for μrover use where less useful information is obtained at $\theta_b = 90°$. Current sensing results are much more noisy and less linear than the results from the photodiode array. In particular, the ADC offsets and gains must be calibrated for each panel

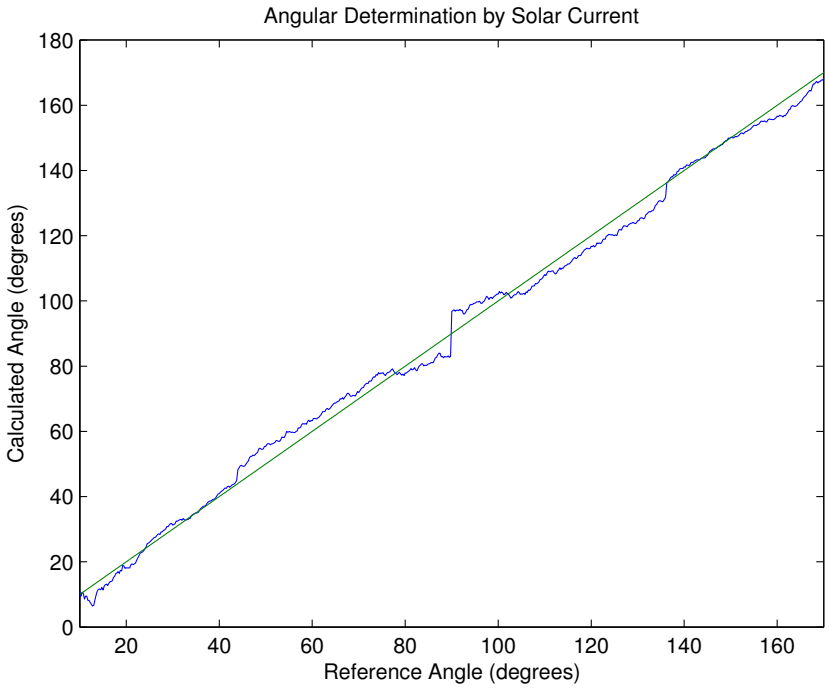


Figure 4.13: Angle from All Solar Panel Currents for 0° to 180°

separately to ensure that measurements can be compared. By applying Equation 2.106 and determining the angular quadrant around the satellite that the sun is in, it is possible to extract an estimate of relative solar angle to each panel. Combining several panels allows determination of a solar angle with respect to the vehicle body at any angle observed by solar panels. While this method may be considered generally less reliable than direct solar measurement, it does allow solar angle measurement without dedicated sensors at a wider range of angles than a single external sensor would be able to measure. Equations 2.103, 2.104, 2.105 can be applied for a rover so long as two orthogonal axes of angular measurement can be obtained from the solar panel geometry.

4.2.4.3 Comparison of Sensing Methodologies

To effectively compare the two methodologies described here, it is important to include the error of measurement with respect to ground truth during testing. Figure 4.14 shows the estimation

error for the linear array angle measurement of θ from Figure 4.6 using Equation 2.102. Figure 4.15 shows the estimation error of the transverse angle ϕ for the array while using an N-slit from Figure 4.9 using Equation 4.1, and Figure 4.16 shows the error in Figure 4.10 using an N-slit for heading estimation. Finally, Figure 4.17 shows the estimation error for the solar panel current angle measurement from Figure 4.13 using Equation 4.3. The linear array shows a maximum error of approximately $\pm 5^\circ$ overall with less consistency in the N-slit measurement, while the solar current sensing shows a maximum error of approximately $\pm 7^\circ$. These are comparable results, but the linear array data is obtained by centroiding and is otherwise unfiltered, while the solar current data requires significant filtering to remove measurement noise. Hence the use of a discrete digital sensor is still expected to provide better reliability and overall accuracy, though with appropriate data processing, solar current measurement can also provide useable and complimentary coarse angle measurements. The tracking accuracy and noise present in μrover heading estimation is comparable to the sensor laboratory tests, but slightly lower as a moving average was used, and indicates that useable heading information can be extracted using a single N-slit sensor.

4.2.5 Power Budget

The power budget for all electronic components in the μrover is given in Table 4.6. Each major component currently used for the μrover was tested to evaluate its average power consumption. As the $3.3V$ and $11.1V$ power buses are used to drive different components and run on effectively separate Lithium-Ion cells (with the exception of sharing one ground-level cell), the high and low voltage power consumption was evaluated separately. Also, power production, storage, and consumption are listed separately. The duty cycle of each component gives an estimate of what percentage of time it would be active during a typical mission. In the case of high current devices such as the drive motors, the duty cycle is largely dependent on how much power is available from the solar arrays over the day, currently estimated to be 50%. Two complimentary estimates are given for the OBC as it is expected to be constantly in use, but draws significantly less power when not at a high computational load. An efficiency is also calculated for each component that is based on how approximately efficient the conversion of power to that component is with the

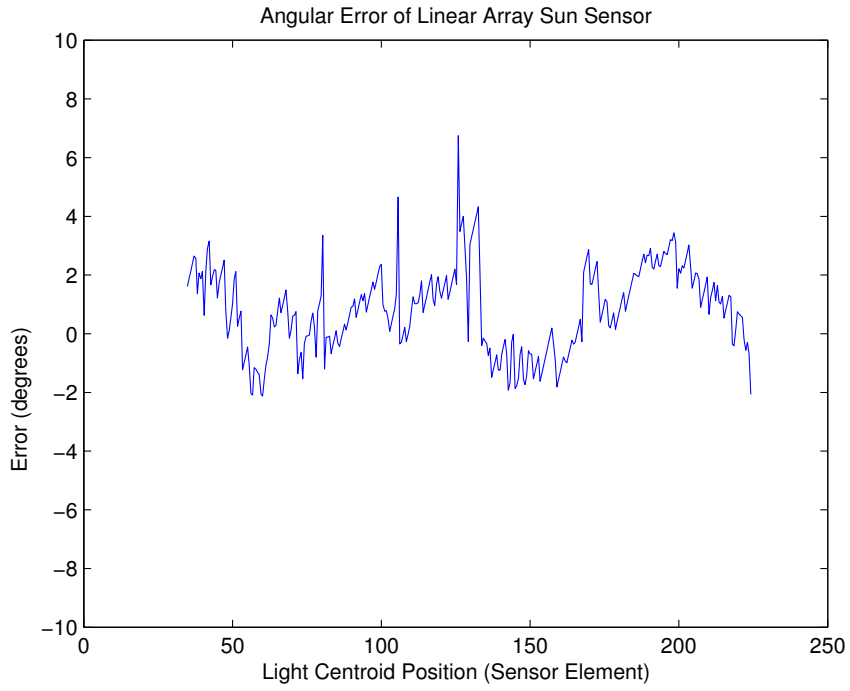


Figure 4.14: Error in Linear Array θ Estimation

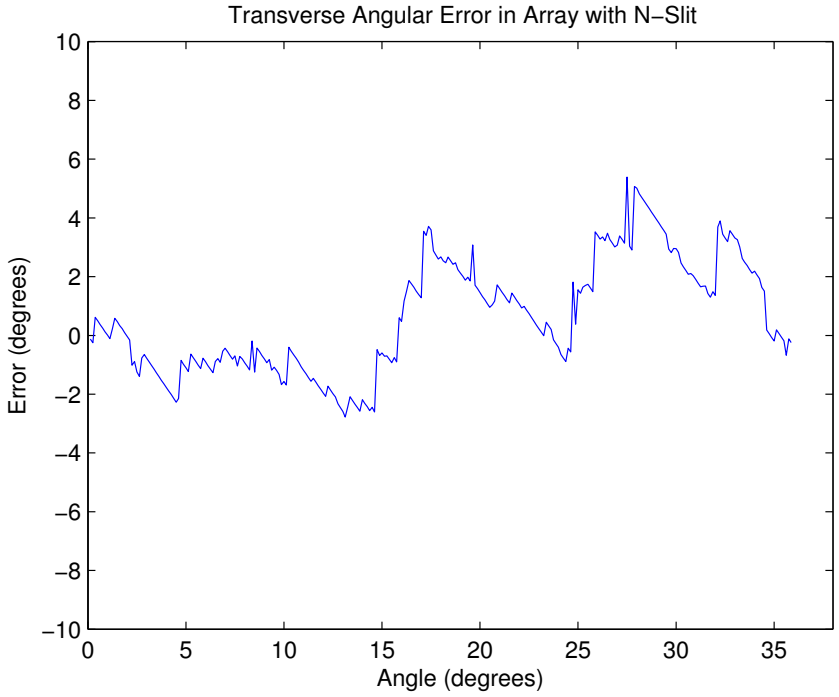


Figure 4.15: Error in Linear Array with N-Slit ϕ Estimation

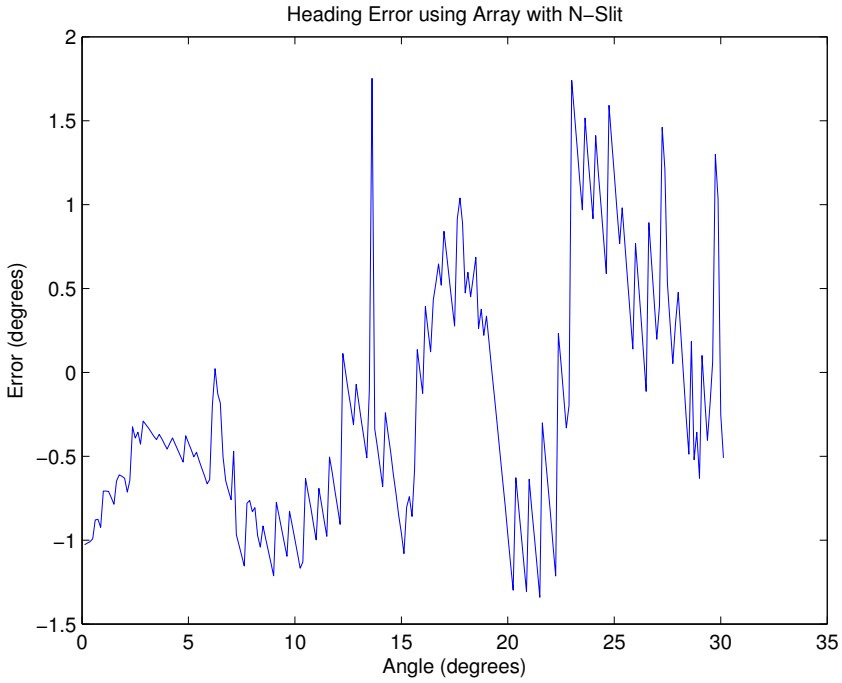


Figure 4.16: Error in Heading Angle Estimation

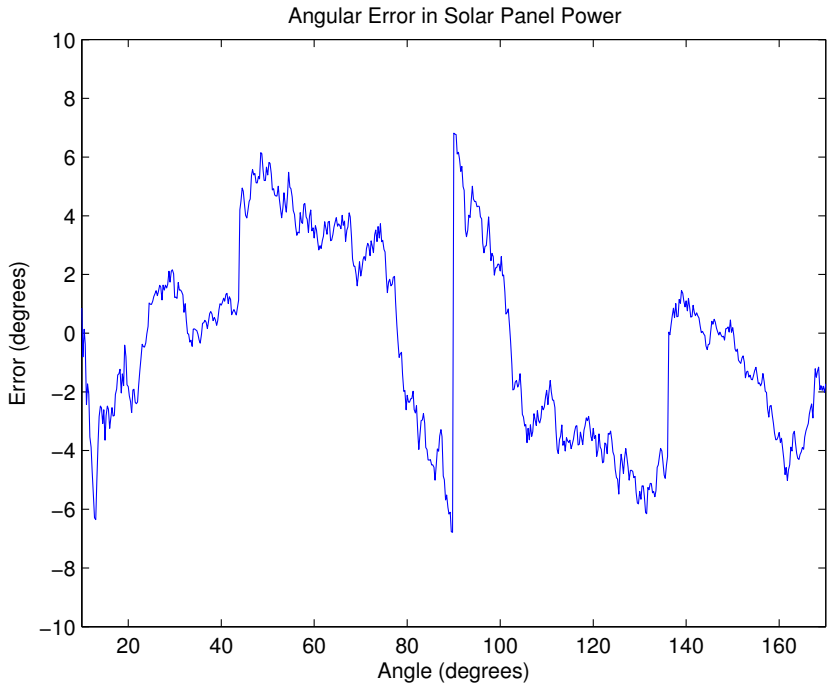


Figure 4.17: Error in Solar Current Angle Estimation

DC-DC switching converters in use. The result is a total power estimate for each component and each bus. Power estimates for the solar arrays and batteries were verified by Navarathinam in previous testing [Navarathinam 2010].

The table indicates that with all solar arrays operating optimally, just over $7W$ can be produced. Just under $1W$ of power is needed under normal conditions for the onboard electronics, and the payloads and drive motors with the estimates given here will require approximately $5W$. This leaves a budget of $1W$ for potential changes in mission duty cycles and charging the onboard batteries, which could store an estimated $17.6W$ in total but are unlikely to retain their optimal capacity in the extreme conditions that will be experienced. If the solar arrays are not deployed or damaged and operating at 50% production, the mission can still be performed with approximately half the duty cycle expected, mainly dependent on the amount of time spent driving. If the batteries are fully charged and at nominal capacity, nearly three hours of normal mission operations can be carried out without solar energy. The μrover monitors how much battery capacity is remaining and shuts down heavy power use components such as the motors if battery power is too low. Shutdown at night is also preferred for avoiding operation in extreme cold and preserving battery power.

Table 4.6: Beaver μrover power budget

Power Use (3.7V)

Part Name	Num Units	Unit (mW)	Duty (%)	Total (mW)	Efficiency (%)	Predicted Use (mW)
OBC Board (idle)	1	297	70%	207.9	85%	244.6
OBC Board (active)	1	561	30%	168.3	85%	198.0
Drive Board	1	415	20%	83.0	85%	97.6
Vision Board	1	693	30%	207.9	85%	244.6
Camera	1	120	20%	24.0	90%	26.7
Radio Module	1	693	10%	69.3	80%	86.6
Total:				760.4		898.1

Power Use (11.1V)

Part Name	Num Units	Unit (mW)	Duty (%)	Total (mW)	Efficiency (%)	Predicted Use (mW)
Drive Motor	4	3800	20%	3040.0	80%	3800.0
Payload Allowance	2	5000	10%	1000.0	80%	1250.0
Total:				4040.0		5050.0

Power Production

Part Name	Num Units	Unit (mW)	Duty (%)	Total (mW)	Efficiency (%)	Predicted Generation (mW)
Main Solar Array	1	8818	50%	4409.0	80%	3527.2
Folding Solar Array	2	4408	50%	4408.0	80%	3526.4
Total:				8817.0		7053.6

Power Storage

Part Name	Num Units	Unit (mWh)	Duty (%)	Total (mWh)	Efficiency (%)	Predicted Storage (mWh)
Batteries (3.7V)	1	5180	100%	5180.0	85%	4403.0
Batteries (11.1V)	3	5180	100%	15540.0	85%	13209.0
Total:				20720.0		17612.0

4.3 Fixed-Point Arithmetic Testing

The value of fixed-point arithmetic is determined largely by the increase in processing efficiency it holds over floating-point arithmetic, particularly on systems without a dedicated FPU, while trading off numerical precision. Here, we test both the precision of the fixed-point representation and the relative speed with which calculations can be performed with respect to floating-point operations implemented by the compiler largely using software methods. Though some ARM instructions are designed to speed up floating-point calculations, a dedicated FPU is regardless much faster, so a significant difference is expected. To generate as large a set of numbers with as high numerical entropy as possible without causing overflows that may alter test results, a list of $N = 1000$ numbers was initially created in both "double" and "fix" formats using the rule

$$a_n = \log_2(n\pi), \quad n = 1 \ldots N. \tag{4.4}$$

This list of numbers was used to create an opportunity for $N^2 = 1000000$ total iterations of each calculation by nested loops over two indices in the appropriate list and performing a calculation for each combination of list elements $[a_n, a_m] \forall (n = 1 \ldots N, m = 1 \ldots N)$. Calculations for addition, subtraction, multiplication, division, inverse, square root, sine and cosine were performed and timed separately for both floating-point ("double") fixed-point ("fix") operations. The extra function call overhead for fixed-point functions was minimized by declaring the fixed-point operations as inline. To prevent the compiler from optimizing out the operations used, the result was added to an accumulation variable in each case, which was printed after timing completed to ensure that the variable was used. Table 4.7 shows the time taken for all one million iterations of each operation on the 180MHz AT91RM9200 ARM9 microcontroller that is used for the μrover on-board computer. The operation that is run for 1 million cycles is shown, then the time taken using "double" floating point in seconds, the time taken using *16.16* fixed-point in seconds, and the speed up ratio $t_{floating}/t_{fixed}$ are given for each operation.

In all results, a significant increase in speed is evident, though less so for the operations of division and inversion, due to the extra division required in fixed-point format. Multiplication

Table 4.7: Results of Timing Tests for Floating-Point and Fixed-Point Math Operations

Operation	Time, Floating-Point (s)	Time, Fixed-Point (s)	Ratio
Addition	1.003	0.057	17.69
Subtraction	1.144	0.056	20.27
Multiplication	0.976	0.1460	6.678
Division	3.662	3.215	1.138
SquareRoot	15.733	2.001	7.862
Invert	4.119	2.890	1.425
Sine	25.183	0.202	124.8
Cosine	24.994	0.202	123.6

retains a noticeable improvement. Addition and Subtraction show a predictably large increase in speed as they can be done as native integer operations, which is very beneficial for algorithms that use large sums and differences. Sine and cosine operations show the greatest increase in speed due to the use of look-up tables, which are only practical for sine and cosine as they are periodic functions.

To evaluate the precision error of different operations in *16.16* fixed-point format relative to floating point "double" format, sets of pseudo-random numbers were chosen in increasing orders of magnitude and each operation was performed using these numbers. The rationale is that the error of fixed-point will change as numbers change in scale because the fractional step size is fixed at $|1/2^{N_f}|$, and therefore will be less important for larger numbers. The method for choosing sets of numbers uses the pseudo-random function rand which produces numbers between 0 and 1, and the exponential constant e to produce exponentially-increasing numbers with high entropy, and is defined as

$$a_n = \text{rand} \times e^n, \quad n = 1 \ldots N. \tag{4.5}$$

Using these sets of test numbers, the logarithmic error for each operation $log_{10}(|c_{fix} - c_{float}|)$ was plotted against the logarithm of the resulting floating-point value c_{float} from each operation, or just the value itself for sin and cos. It is evident that the error does decrease as the order of magnitude for each number increases, and in all cases, the error is manageably small.

Thoughout these tests, subsequent runs gave consistent results, so the information stated here

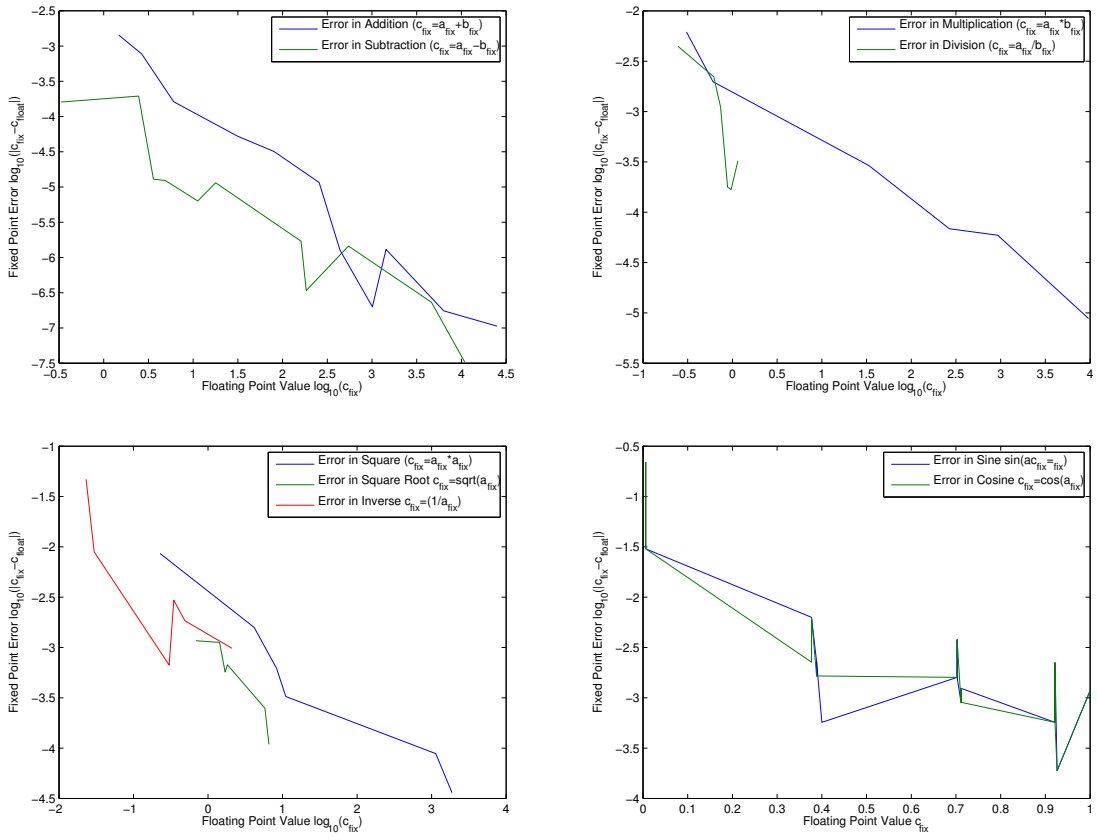


Figure 4.18: X-Y Position and State Error for UKF Positioning Simulation

should remain consistent on similar ARM9 architecture machines. Matrix and vector operations were verified by comparing the results of operations to those in the MATLAB environment, and all functions were debugged and validated as operating properly within the precision of the *16.16* fixed point representation, and compatible in terms of output with MATLAB and Octave.

4.4 Kalman Filtering Results

Several sigma-point Kalman filters based on the UKF and CKF were coded from scratch in MATLAB (without using any of the built-in toolboxes). Variants of these such as the augmented UKF, square-root UKF, adaptive UKF, and fuzzy adaptive UKF were constructed as well. A simulation of a simple skid-steered vehicle with sensors functioning as wheel encoders, a GPS, and a magnetometer was created to test these filters. The state vector used for this filter is $\mathbf{x} = [v, \theta, x, y]$, and it is updated at $1Hz$ using the vehicle model in Equations 2.6 and 2.8. An S-curve maneuver is programmed as the path and positional white (Gaussian) noise of mean $1cm$ and standard deviation $10cm$ is added at each time step, and the same pseudo-random noise is used for all filters in this test (generated from the same random seed). Using this model, the UKF, Adaptive UKF (AUKF), Minimum set UKF (MUKF), and CKF algorithms were tested. Figure 4.19 shows the simulation results of absolute position and state variable error for the UKF. The estimated error for each sample point is shown next to the position, and the total RMS error σ and correction factor of the filter output versus noisy input $(\mathbf{x}(t+1) + q)/\tilde{\mathbf{x}}(t+1)$ is shown under each error plot. An additional test was performed with a fault injected that causes the estimate of position to be 0 between iterations 20 and 30 (a condition which can mirror a temporary loss of communication or hardware failure of the GPS unit) to evaluate whether the filter would become unstable and how fast recovery occurs. Results for this test are shown in Figure 4.20.

The UKF with the addition of adaptivity for Q using Equations 2.132-2.134 was tested next. Additionally, Equation 2.135 was used to examine the system for unacceptably large errors in estimation, with the error factor shown next to the position sample points. Green numbers are error factors below the threshold for this test $\varsigma_t = 11$, and red numbers are larger error factors, for

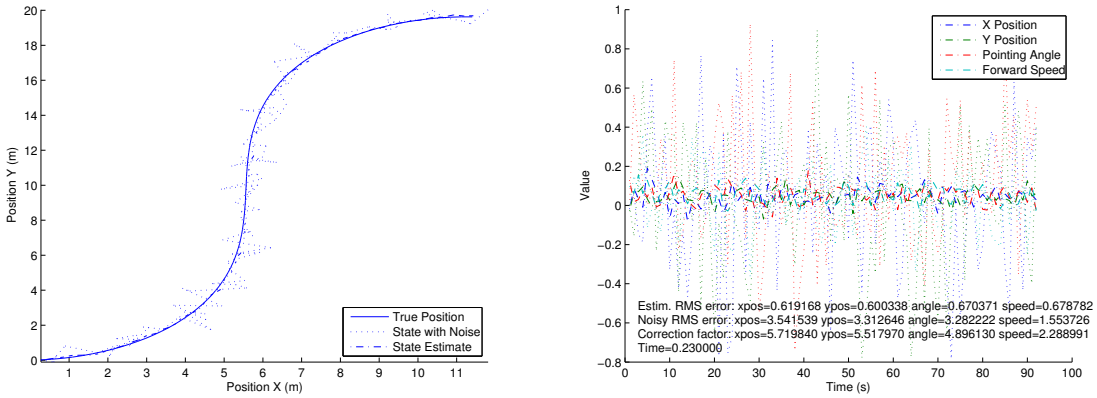


Figure 4.19: X-Y Position and State Error for UKF Positioning Simulation

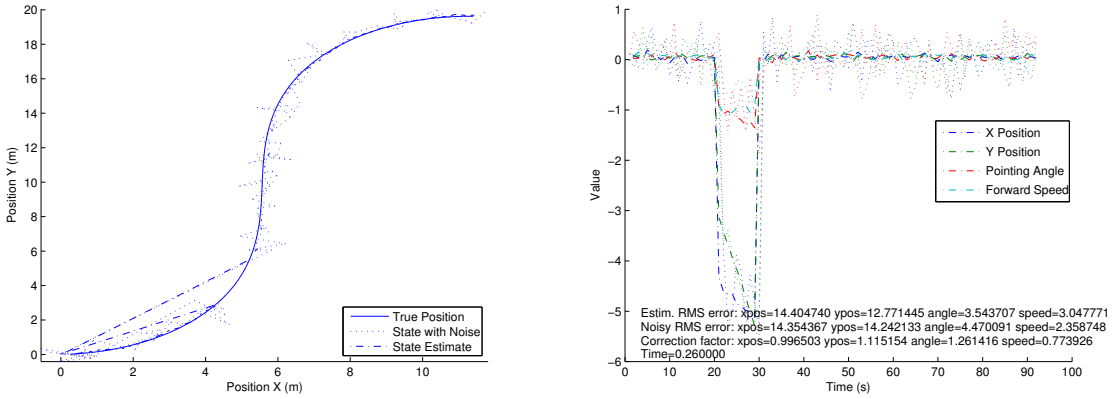


Figure 4.20: X-Y Position and State Error for UKF Positioning with Position Sensor Fault

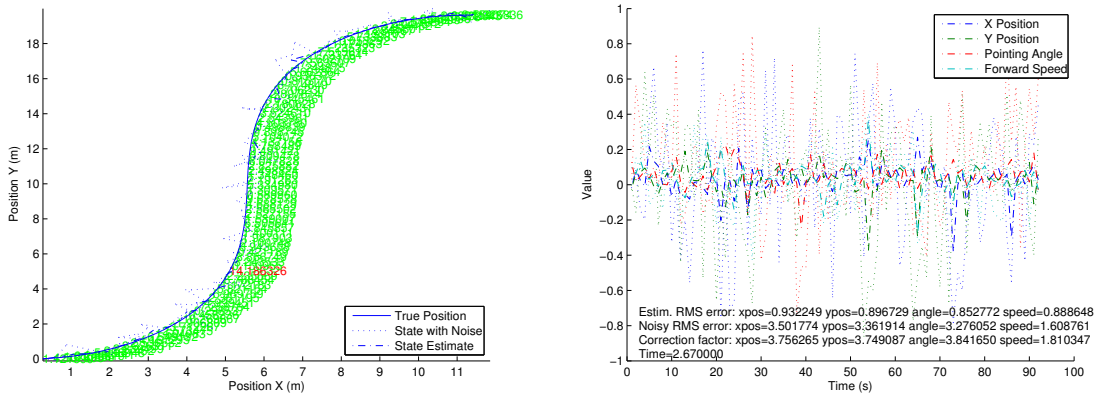


Figure 4.21: X-Y Position and State Error for Adaptive UKF Positioning Simulation

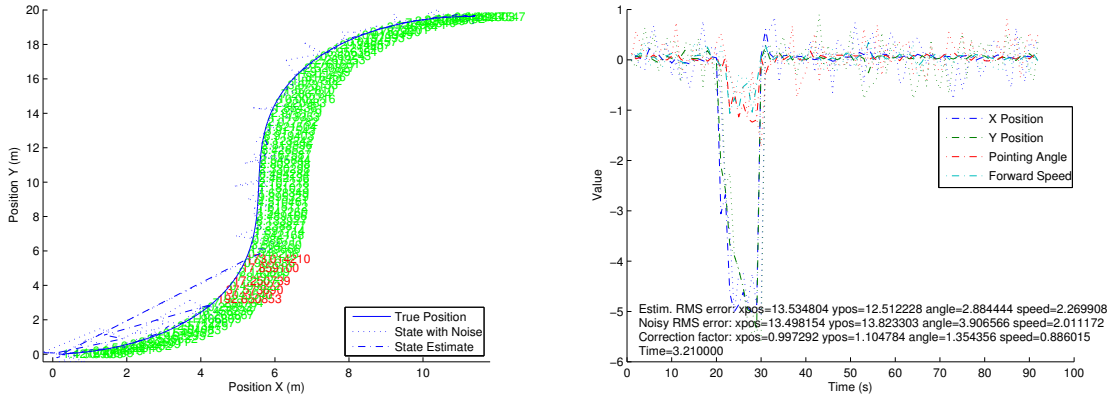


Figure 4.22: X-Y Position and State Error for Adaptive UKF Positioning with Position Sensor Fault

which the state and sensor error covariance matrices should be adapted to improve performance. Results are shown in Figure 4.21. The results for the same test with the fault injected are shown in Figure 4.22

To illustrate the adaptation of the $\mathbf{P}$ and $\mathbf{Q}$ matrices over successive iterations of the filter, the elements of $\tilde{\mathbf{P}}(t+1)$ and $\hat{\mathbf{Q}}(t+1)$ are plotted in Figure 4.23 for the case with no fault, and in Figure 4.24 for the case with the fault injected. It can be noted that the adaptivity of $\mathbf{Q}$ has a subsequent effect on lowering $\mathbf{P}$, and both change rapidly in response to the fault.

For comparison, the implementation of the Minimum-set UKF using Equations 2.136-2.142 was tested under the same conditions, with results shown in Figure 4.25. Similar results to the

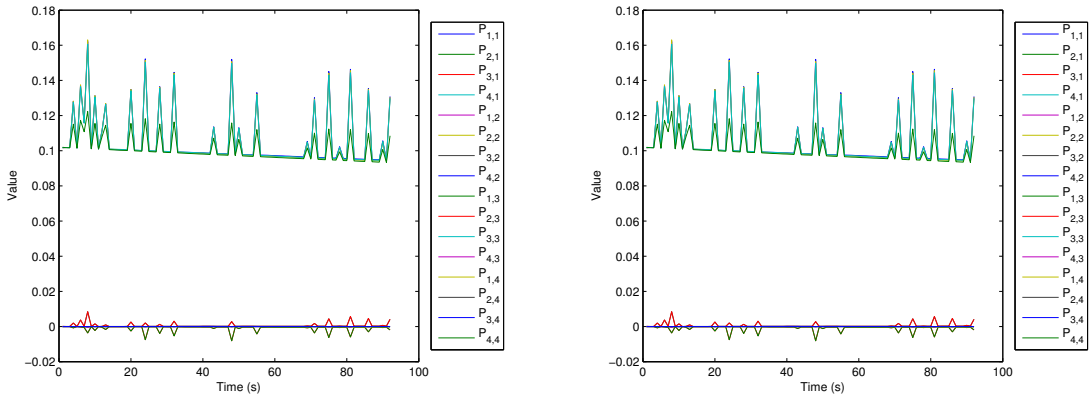


Figure 4.23: Elements of **P** and **Q** Matrices for Adaptive UKF with Position Sensor Fault

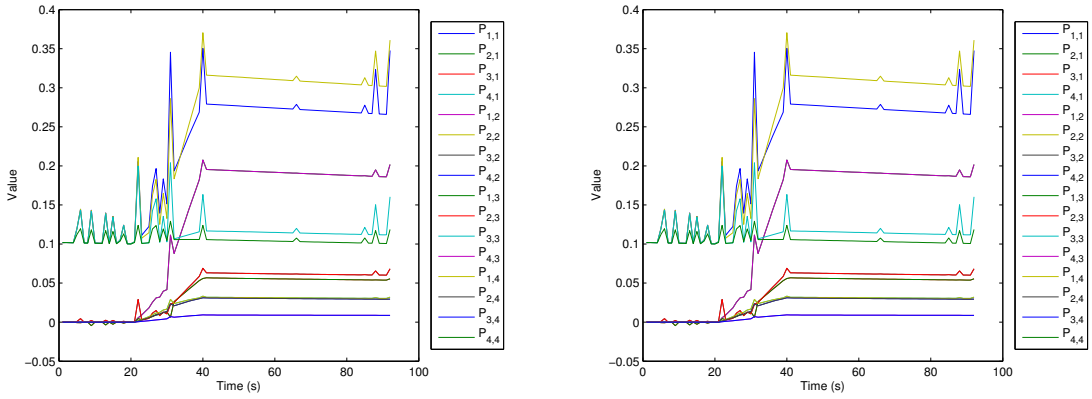


Figure 4.24: Elements of **P** and **Q** Matrices for Adaptive UKF with Position Sensor Fault

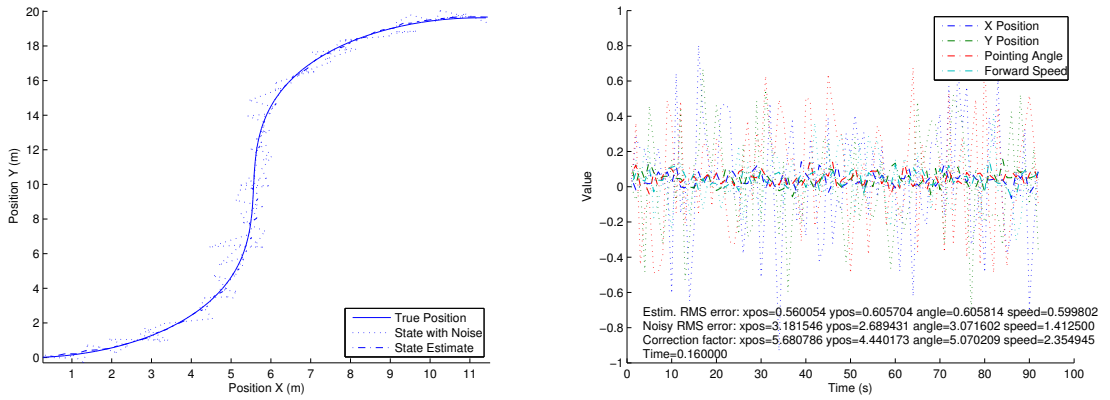


Figure 4.25: X-Y Position and State Error for Minimum-Set UKF Positioning Simulation

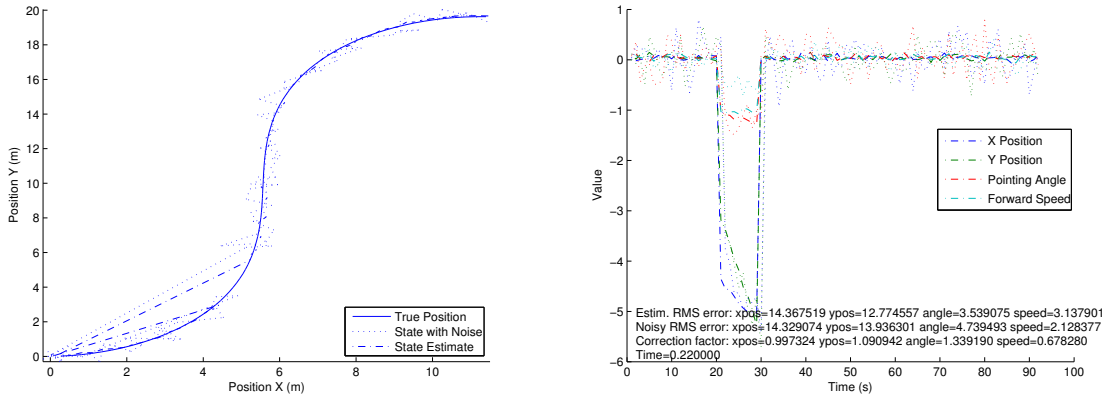


Figure 4.26: X-Y Position and State Error for Minimum-Set UKF Positioning with Position Sensor Fault

UKF can be observed primarily due to the simplicity of the system model, which allows the MUKF to remain stable and comparable in performance.

Finally, the CKF implementation was tested under the same conditions, with the fault-free case shown in Figure 4.27 and the fault shown in Figure 4.28. For this example, the CKF performs marginally better than the UKF in most cases, though relative performance depends to some extent on the noise injected.

The best results in general were achieved with the comparably-performing basic UKF and CKF, although this is partly due to the simplicity of the system model. Adaptive updating of noise covariance matrices operates, but does not improve performance in this case where the system

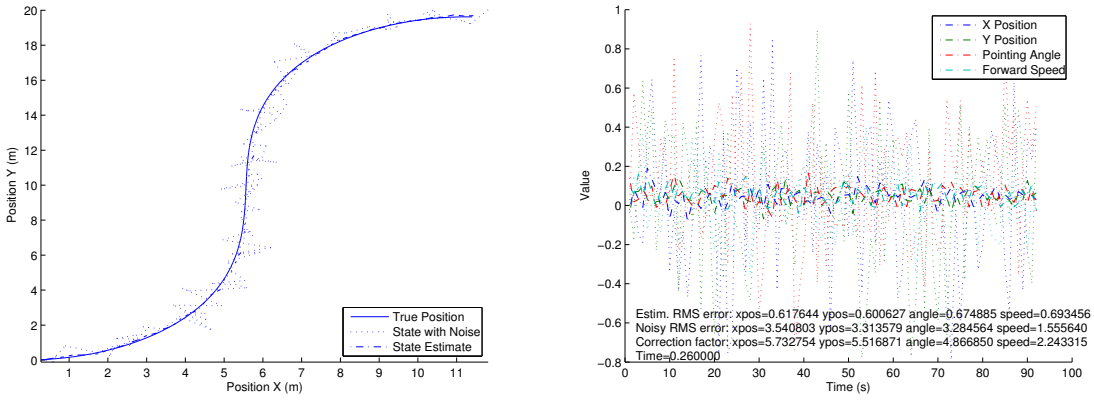


Figure 4.27: X-Y Position and State Error for CKF Positioning Simulation

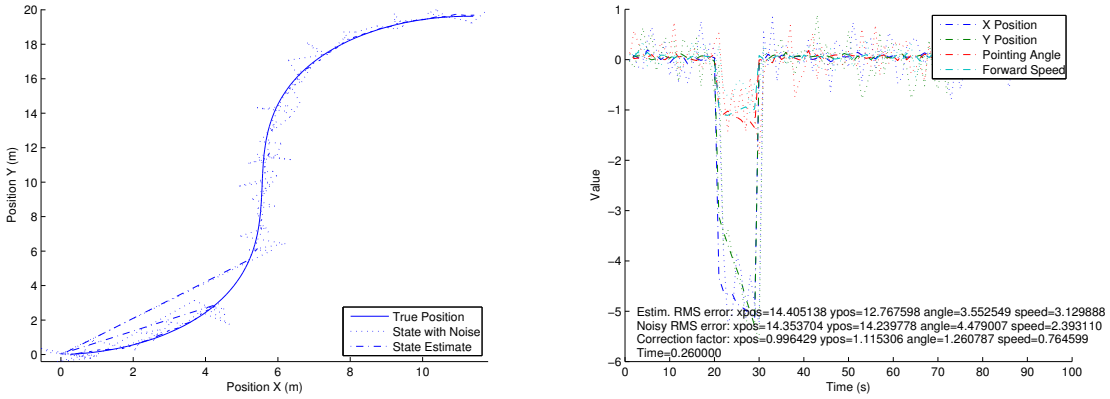


Figure 4.28: X-Y Position and State Error for CKF Positioning with Position Sensor Fault

is well-characterized and may provide better performance in real applications due to changing statistics. These implementations have verified that the CKF performs at worst, equivalently to a UKF and at best, better than a UKF on simple nonlinear systems with Gaussian statistics. The MUKF performed well in general for this case and can be used to decrease execution time for the filter. However, the UKF itself is used for most of the remainder of this research due to the flexibility in adjusting sigma points and the option of adding adaptivity for noise covariances, which involves a simple set of extra calculations. Choosing the appropriate noise covariances is important for accurate assumptions made by the filter. For example, if the system noise covariance estimate is too small, there will be a tendency for the state to execute a "random walk" by deemphasizing the sensor readings.

4.5 RTOS Performance Results

4.5.1 Test Setup

The test suite consists of three programs. Each program should be executed separately (i.e. not at the same time on the same board) The tasks are described below. Each platform has a separate source code tree including the three tests and all associated code.

- Program "rt_int_measure", executes an ISR when an external line is triggered
- Program "rt_video_measure", performs a simple feature tracking algorithm on a video file
- Program "rt_comm_measure", performs receive-retransmit operations on Ethernet packets

Each program executes the appropriate task at 1ms intervals while sampling the system clock at entry and exit. The logic fabric on the attached FPGA runs from a separate high-resolution timer and logs the signalled start and stop times to a First-In First-Out buffer (FIFO). The start time of the thread should exhibit a constant offset from the high-resolution timer. The time from start to execution of an interrupt is considered to be interrupt latency. The time from execution to end of the thread is considered to be run time. Any deviation in start or end time over successive runs is considered to be jitter.

The test programs are based on Express Logic's open-source Thread-metric routines, which provide both single-threaded and multi-threaded testing for RTOS characteristics. While co-operative scheduling of 5 threads at once is possible with these tests, the focus will be on the performance of a single high-priority thread under load from lower-priority processes. For bare-hardware targets, only a single thread may be possible as no scheduler is available, so loading tasks will be run only before and after the high-priority task. Loading is accomplished using code from the stress program.

The rt_int_measure program first sets up an interrupt handler for a hardware interrupt line, creates a low-priority load thread if needed and then messages the RT logic core periodically to generate a hardware interrupt. The program then services the interrupt, signalling the logic core at execution, and records the time between interrupt initiation and latency. For Linux tests, a kernel driver must be used to service the interrupts, and the rt_int_measure program then reads the time that the driver recorded through a device node. The data is written as a single column of values, each representing an interrupt latency measurement. The rt_vid_measure program was designed to read a short segment of MJPG video into memory from the SD card, initiate a low-priority load thread if needed, and then initiates a thread that runs a frame-by-frame decompression and FAST keypoint processing algorithm. However, due to the complexity of reading in such a video segment to bare hardware, for comparison with bare hardware a simple loop with a square-root operation is currently used in the comparison results across all platforms that provides a consistent algorithm. Messages are sent to the logic core at the start and stop of each process, and the difference between these measurements is used as the processing metric. Data is output in three comma-separated columns for the starting time measurement, the ending time measurement, and the difference between the two which is the processing time.

The rt_comm_measure program measures real-time processing of Ethernet packets, and is currently unavailable on bare-hardware targets. When started, this program will start a loading thread and then listen for connections on port 30000. The Ethernet port should be connected directly (with a crossover cable if necessary) to a host computer running Linux, or alternately directed to the loopback address 127.0.0.1. The companion program rt_comm_txrx can be run on a separate

computer or alternately on the same computer using the loopback address. The rt_comm_measure program will send each packet, receive the same packet sent back by the rt_comm_txrx program, and parse the contents. The round-trip transmission time is measured by the real time logic. It should be noted that unless the target computer is also running a real-time OS, the processing time will not be real-time deterministic due to the processing required by the connected computer. Data is output in three comma-separated columns for the starting time measurement, the ending time measurement, and the difference between the two which is the round-trip time.

4.5.2 Test Results

In general, the results show performance attributes between the different platforms that would be expected. Unpatched Linux shows comparatively high latencies, processing time and jitter in all tests performed and serves as the reference for real-time performance, though large latencies are often experienced as other tasks interrupt the system.

Column graphs of the data show the accumulated timing values for all RT tests, with interrupt testing in Figure 4.29, communications testing in Figure 4.30, and processing testing in Figure 4.31. Average, overall maximum, and standard deviation values are shown for all performed tests. For non-RT tests, the outliers described above have been removed from the data so that a more accurate comparison can be made. Note that outlying values have been removed for a closer comparison, but in the case of Linux or applications that have non-RT components such as communications with non-RT systems, large peak latency values (orders of magnitude larger) can occur in up to 5% of measured latency cases.

The Linux kernel with the Xenomai RT layer does not show a vast difference, but performs slightly better in terms of processing time and latency and much better in terms of jitter. This underlines the most important reason to use a hard real-time kernel: execution time has a fixed limit and does not experience the large outlying latency spikes that occur in non-RT multitasking systems. These large outliers (2 orders of magnitude higher or more) are present in non-RT Linux and the communications tests done where a non-RT system is used to relay packets back to the RT system, and can sometimes account for up to 5% of the latencies in heavy-load cases. In

general, performance is far more deterministic when using Xenomai but not much faster in terms of system call overhead.

As could be expected, bare-hardware code on the ARM CPU runs much faster and more efficiently than Linux code due to much lower system overheads and simpler operation. However, this is also due to the fact that bare hardware does not natively multi-thread, so these gains are somewhat misleading as there is no context switching and no chance for other processes to interrupt the thread. Also, loading is done sequentially rather than in parallel so the difference between light and heavy CPU loading is not as significant. The bare hardware testing does underline the performance and determinism gains that are possible when the overhead of the OS is removed. Similar conclusions can be drawn from the Microblaze tests, but the major difference is that of overall speed. To obtain comparable times, the timing of the Microblaze processing tests must be scaled down by five times the difference in CPU frequencies between the processors, suggesting that on a per-cycle basis, the ARM processor could be up to five times more efficient in terms of real-time performance. The interrupt processing time at the lower CPU frequency is better and is comparable to Xenomai times, though the amount of code overhead is much less.

Although patching the Linux kernel with Xenomai limits the versions available, the performance and flexibility of Linux is retained and there appears to be no significant drawbacks to running with a patched kernel. As well, both userspace applications and kernel drivers show noticeably better timing performance when run with Xenomai real-time support from ipipe. Although not as fast and deterministic as a dedicated RTOS, Xenomai provides all of the benefits of a Linux system and eliminates the occasional large latencies and accumulated jitter that occur in non-RT Linux. It appears that hard-real-time performance in Xenomai is practical. In an embedded system where real-time performance for some processes may be desirable, there is really no reason not to use a Xenomai kernel, so if Linux kernel versions have to be clearly defined for development, exclusively Xenomai kernels should be used. While bare-hardware code runs more deterministically, and could be arguably more reliable with sufficient work, over the course of this project the bare-hardware code proved to be the least reliable and hardest to maintain and port. Additionally, only one process at a time can be run without a multitasking or interrupt

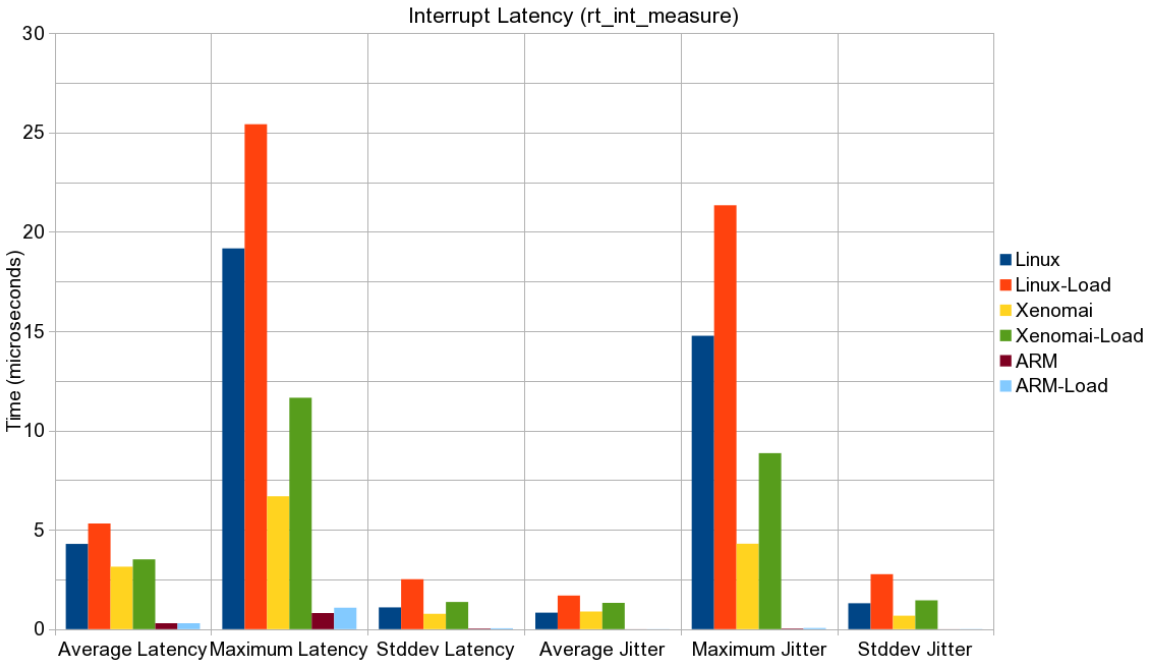


Figure 4.29: Results of real-time interrupt testing of OS candidates

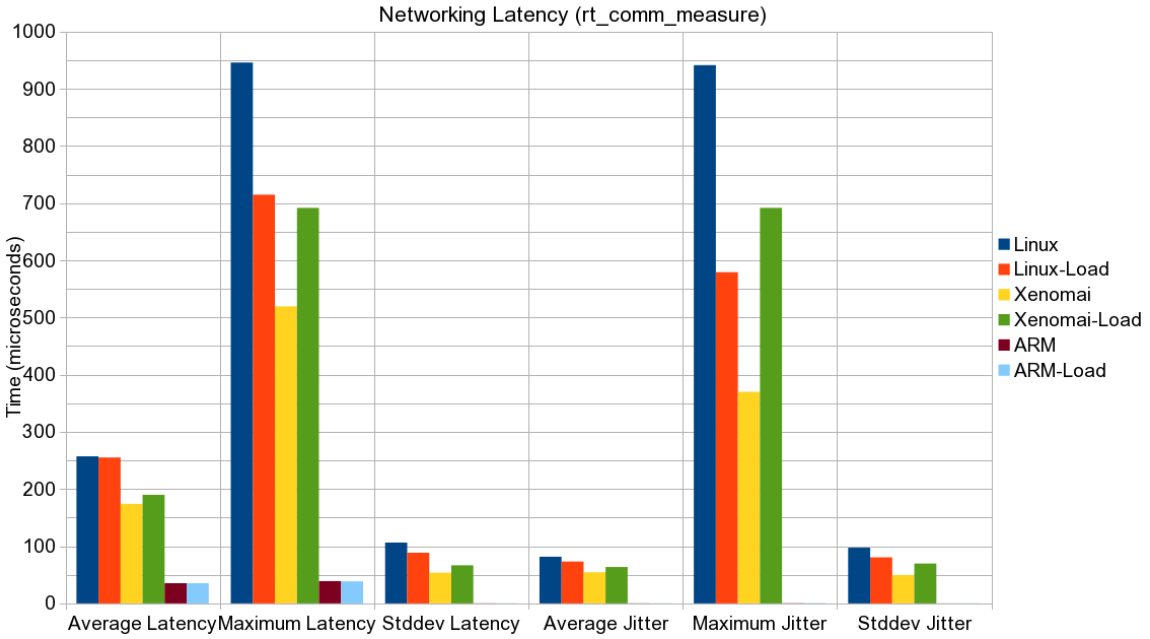


Figure 4.30: Results of real-time Ethernet testing of OS candidates

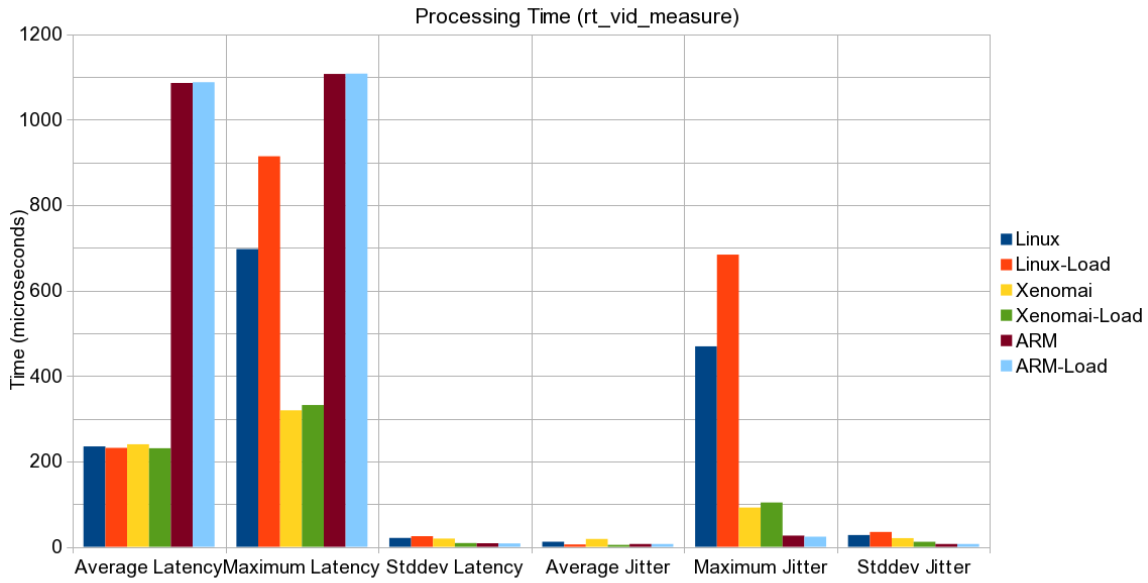
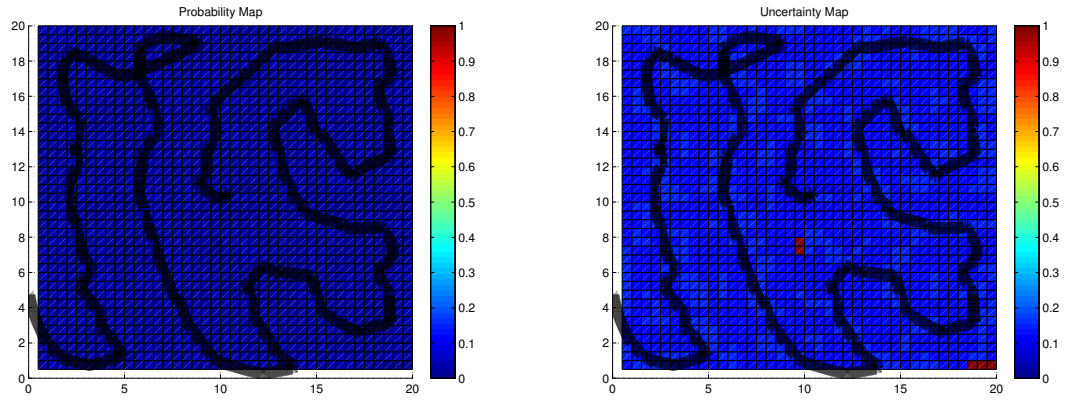


Figure 4.31: Results of real-time video testing of OS candidates

mechanism and making peripherals such as Ethernet and file systems functional is much more difficult as detailed above. The use of an operating system framework with POSIX interfaces, even a simple RTOS, is recommended to avoid these problems.



(a) P($O|R, r, \mathbf{x}$)

(b) H($O|R, r, \mathbf{x}$)

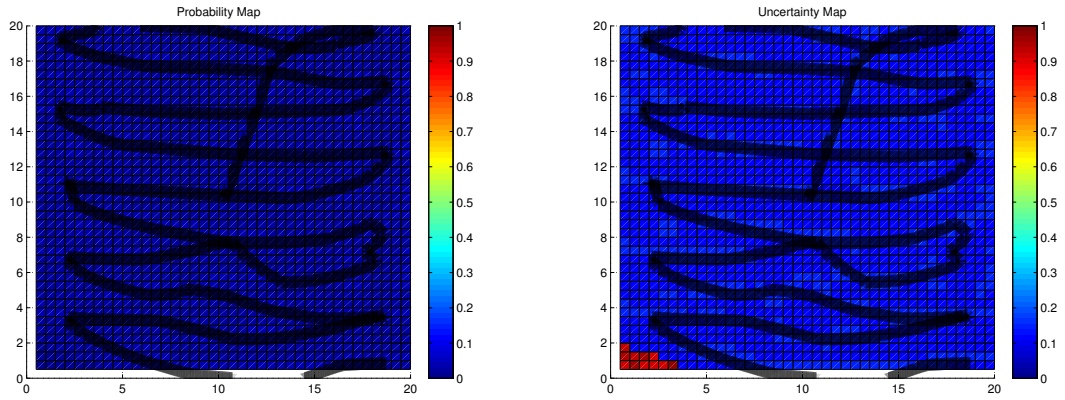
Figure 4.32: Probability and Uncertainty Maps, No Obstacles

4.6 Bayesian Navigation Results

Using the methods described in Sections 3.5-3.6, the Beaver was given a 20m by 20m area for motion, and mapping was constrained to this area in software, although the rover could physically leave the map due to turning radius constraints. Using the Bayesian mapping strategy with the goal of exploring all the given map area thoroughly, the rover was allowed to move freely in an area with no obstacles. For this test, $\beta_b = 0.2$, $d_{max} = 8m$, P(R) = 0.2, and a grid with $0.5m$ resolution were used. The grid resolution reflects not only the noise and uncertainty in GPS measurements, but also the safety margin around obstacles that is desired to avoid collisions.

First, each location in the probability map was initialized to 0 and each in the entropy map was initialized to 1. Figure 4.32 shows the probability and uncertainty maps. The path that the rover took during the test is shown as an overlaid black line. The initial entropy map was then modified with a pseudo-random offset as suggested above with $\delta_h = 0.1$. Figure 4.33 shows the probability and uncertainty maps. The new path that the rover chose is shown.

Three obstacles were then placed in the 20m by 20m area to test the obstacle avoidance methodology. Two $1mx1m$ obstacles were placed at $(14m, 16m)$ and $(7m, 10m)$, and a $0.5mx1m$ obstacle was placed at $(12.5m, 3m)$ in grid coordinates. The layout of the testing area used is shown in Figure 3.5. The rover was run with the same parameters as the tests above with the



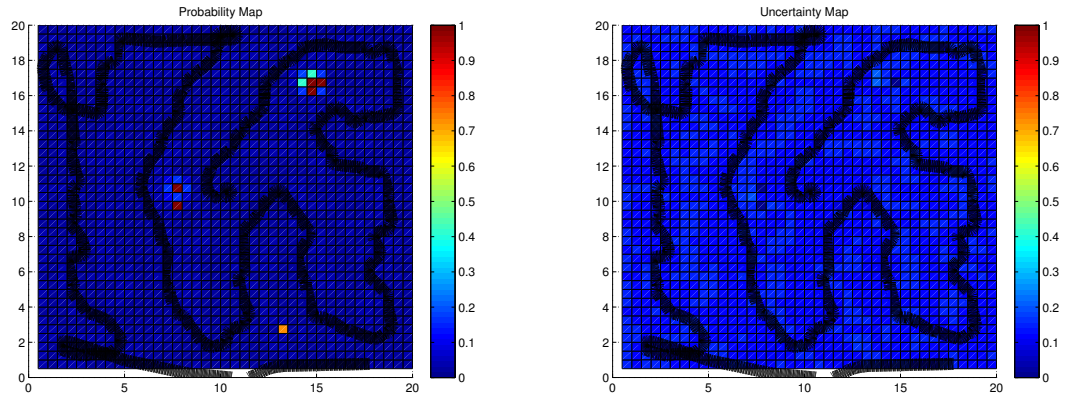
(a) P($O|R, r, \mathbf{x}$) (b) H($O|R, r, \mathbf{x}$)

Figure 4.33: Probability and Initially-Randomized Uncertainty Maps, No Obstacles

uncertainty map initialized to 0 and the entropy map initialized to 1. The resulting path and maps are shown in Figure 4.34. The test was repeated with the pseudo-random offset as suggested above with $\delta_h = 0.1$, and the results are shown in Figure 4.35.

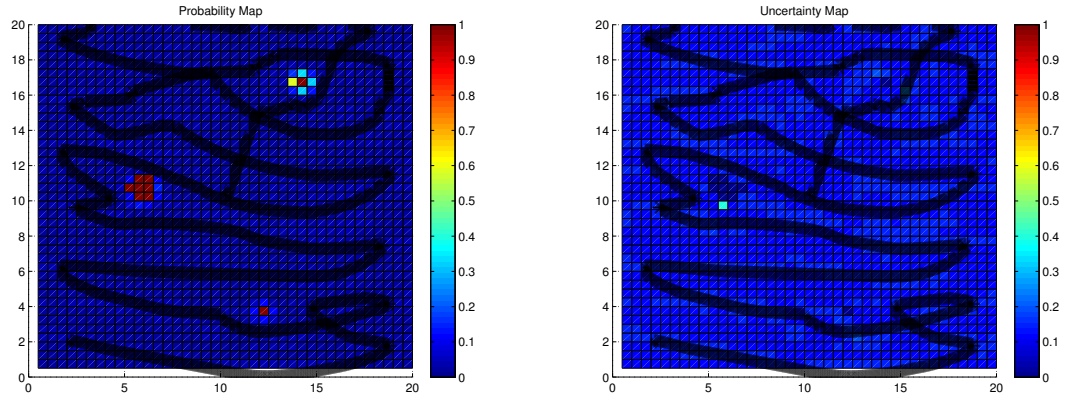
The obstacles are not overly obvious given the statistical nature of the mapping, but they are visible as points of high probability and low uncertainty, while the remainder of the mapped area retains an uncertainty of close to 0.1 on average. The peaks in obstacle probability vary between runs due to small differences in location $\mathbf{x}$ or sensor reading r that occur due to real-world uncertainties. However, even though the probability map varies with each run, the results obtained regarding object presence are very consistent, as statistical methods are generally robust to uncertainties. Because mapped probability depends on previous map measurements as well as current ones, if the micro-rover moves too fast, the reliability of the measurements will decrease as well. It can be noted that obstacles that lie directly in the path of the rover have better characterization in terms of high probability, because if the uncertainty driver does not force the rover to get close to obstacles, the sensor model will not place as high a reliability on the resulting probability.

The path was fairly consistent between test runs, with the exception of occasional sensor errors that momentarily cause the rover to begin obstacle avoidance behaviours, and show up as small course deviations or loops in the path. The pseudo-random offset caused the rover to



(a) P($O|R, r, \mathbf{x}$) (b) H($O|R, r, \mathbf{x}$)

Figure 4.34: Probability and Uncertainty Maps, With Obstacles



(a) P($O|R, r, \mathbf{x}$) (b) H($O|R, r, \mathbf{x}$)

Figure 4.35: Probability and Initially-Randomized Uncertainty Maps, With Obstacles

prefer a Cartesian side-to-side movement indicating that the greatest uncertainty search in Eq. 3.66 was dominant, while a flat uncertainty initialization caused a shorter and less regular pattern indicating that the closest point search in Eq. 3.67 was dominant. The same random seed was used on all tests, so observing the same motion pattern in both cases is expected. This also causes a more thorough traversal of the map, resulting in better obstacle characterization. In both cases, it is evident that the obstacles are detected as peaks in the probability map and successfully avoided, although due to the high granularity of the mapping, the obstacle location accuracy is quite coarse. No comparison was done in this study between different test areas and obstacle layouts, and it is possible that a less predictable path could be desired. In this case, greater randomization in the algorithm would likely suffice.

4.7 Vision System Testing

To determine the suitability of Structure-from-Motion methods for multi-robot mapping, a test was conducted with the μRover prototype and a camera-equipped remote-controlled quadrotor. The main purposes were to determine how high a spatial resolution for terrain mapping could be achieved using SfM techniques and whether SfM was suited to mapping done by multiple vehicles. Processing was done using the free but partially closed-source VisualSFM software suite, which uses hardware-accelerated SURF and the SSBA sparse bundle adjustment package to perform feature matching and triangulation. An image of the quadrotor and μRover operating together is shown in Figure 4.36 and the resulting tracks as processed by VisualSFM are shown in Figure 4.37. It is evident that with a sufficient number of feature points, both the position of the cameras and the features in the image can be inferred with reasonable accuracy. However, computing just this one map took over 7 hours on an Intel Core I7 2600 equipped with an NVidia GTX570 (GF110) video card for GPU-based acceleration.

For use on the μrover, we tested a set of much computationally-simpler vision algorithms based on the ORB feature descriptor and pairwise triangulation. Testing was done outdoors in an open area with rocky ground, similar terrain to what would be expected on Mars. The Farneback optical flow algorithm from Equations 3.92-3.95 was implemented for short-distance ego-motion estimation as described in Section 3.7, and Figure 4.38 shows a typical example of how optical flow operates between two frames in a short distance of motion. Monochrome versions of captured images are used as the Farneback algorithm does not operate on colour images. Although all pixels in the image are used for ego-motion estimation, the green arrows show the directions of flow for a subset of test locations in the image. Using the ego-motion estimation algorithm described in Equations 3.102-3.108, the three-dimensional translation and rotation of the camera is estimated as well. The red circle in the image shows the "target" of ego-motion as projected on the two-dimensional image plane.

Visual SLAM and point cloud construction is done based on the ORB feature descriptor implemented in OpenCV as described in Section 3.8. Feature detection using ORB results in a

Figure 4.36: Combined μrover and quadrotor mapping

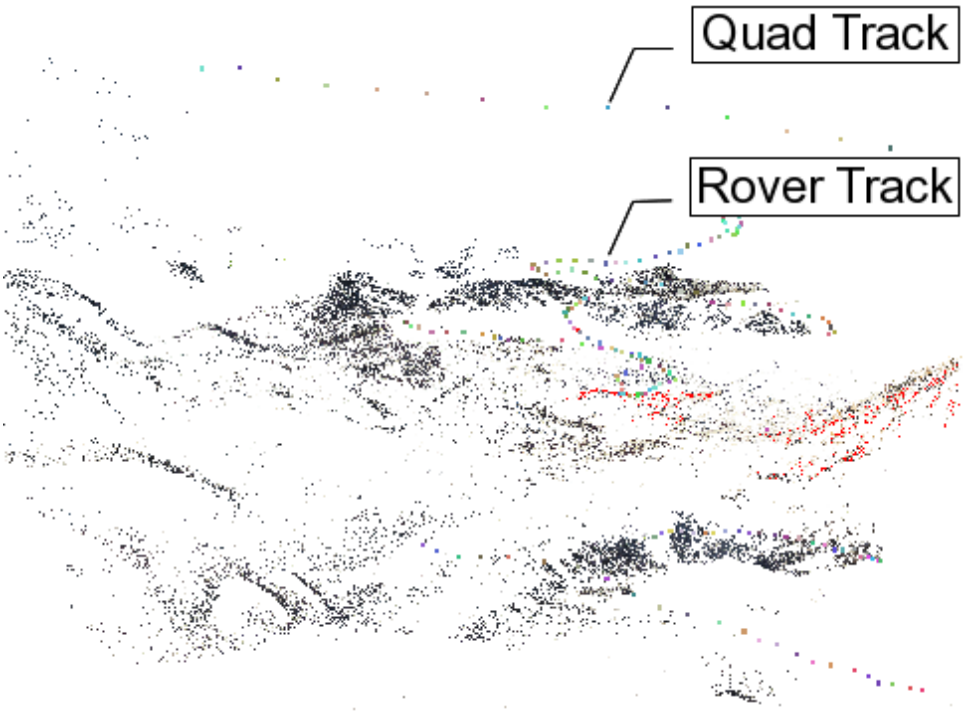


Figure 4.37: Point cloud and Motion Tracks of μrover and quadrotor

Figure 4.38: Optical Flow Field for Forward Movement

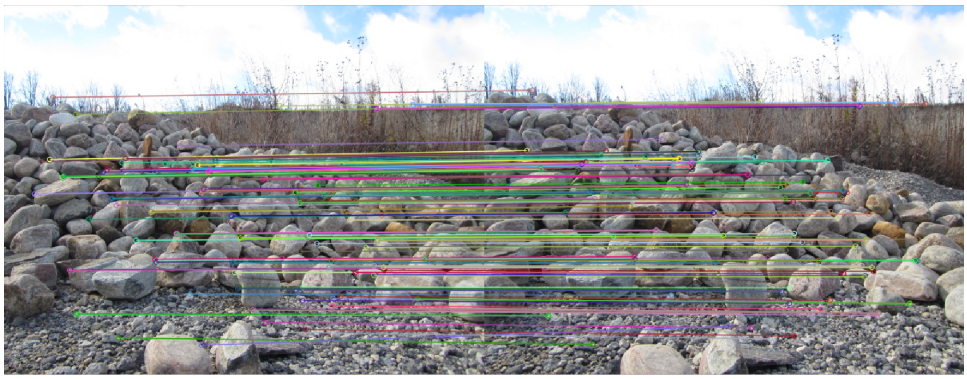


Figure 4.39: Good Matches from ORB Feature Points

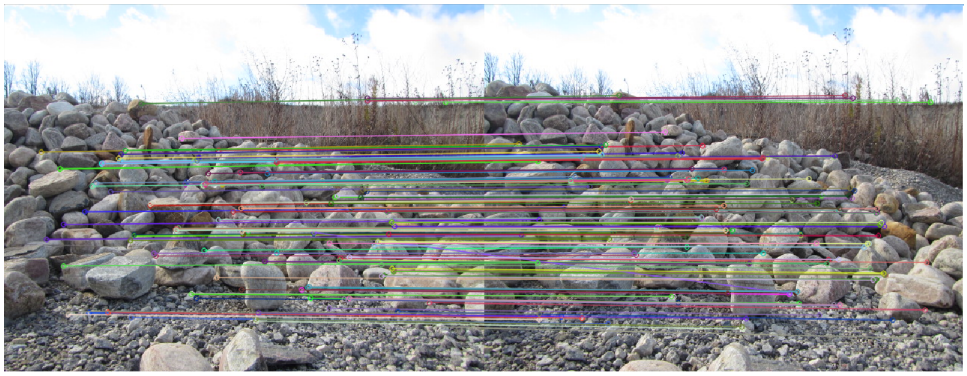


Figure 4.40: Fundamental Matrix Matches for Triangulation

set of feature points that can be matched between successive images such as shown in Figure 4.39. Note that basic removal of points that are obviously matched incorrectly is necessary here. Using these matched points, the fundamental matrix F representing the transformation between the two images was extracted and additional matches were removed that did not fit this model to a tolerance of 10%. The remaining matches are shown in Figure 4.40.

Triangulation of the point matches can then be performed based on the fundamental matrix. The feature point cloud by itself, coloured to represent different depths from the camera, is shown in Figure 4.41. Reprojection of these points into the image gives an idea of the quality of the triangulation, as shown in Figure 4.42.

The resulting 3-D reconstruction, including both the point cloud and camera/rover motion, is shown in Figure 4.43. The uncorrected ego-motion measurements are shown in blue, while the

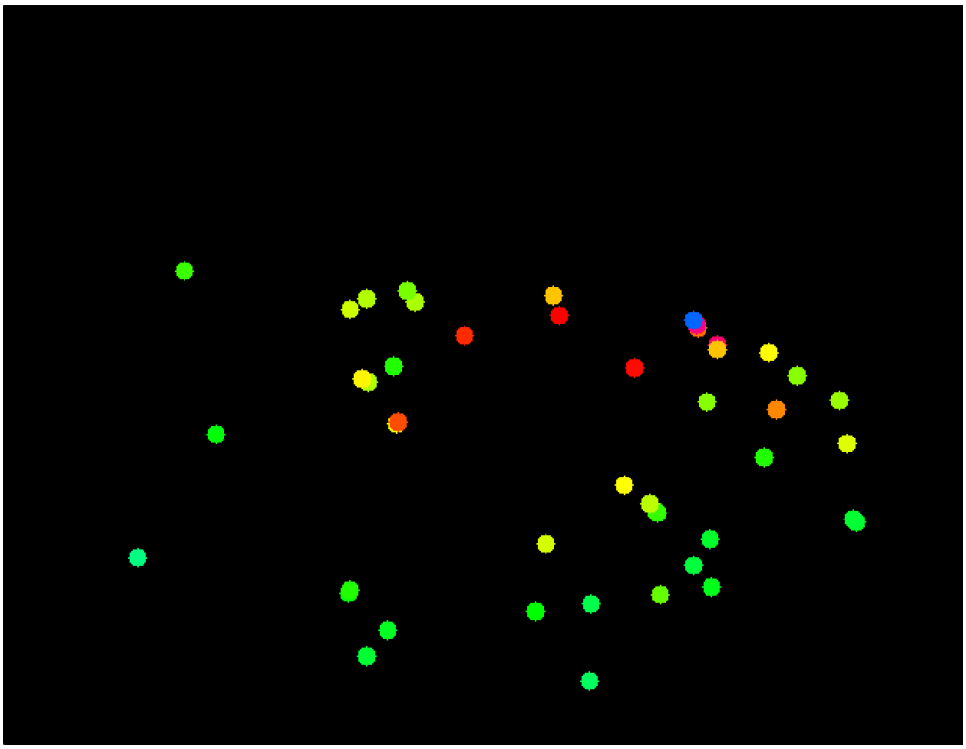


Figure 4.41: ORB Feature Point Cloud

filtered pose estimates are shown in green.

Although the accuracy of each feature point is not high, this reconstruction is sufficient to obtain a statistical measure of obstacle locations and heights in front of the μrover. While optical flow contributes to the motion estimate of the μrover, it is easily confused by scenes that have many similar textures, and this occasionally causes large deviations in position that are filtered out of the final position estimate. The pose estimates given by structure-from-motion methods are in general more accurate as long as a sufficient number of good matches between successive frames are available. Increasing the accuracy performance of the algorithm is mainly a question of increasing the number of correct feature points or landmarks used for triangulation.

Figure 4.42: Reprojected Points after Triangulation

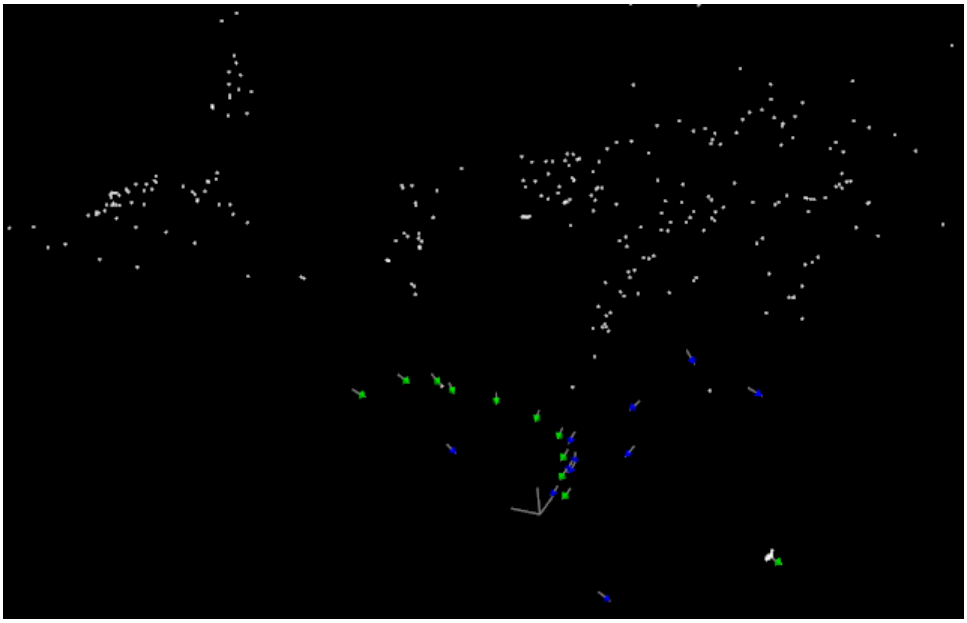


Figure 4.43: SFM Point Cloud for a Simple Maneuver

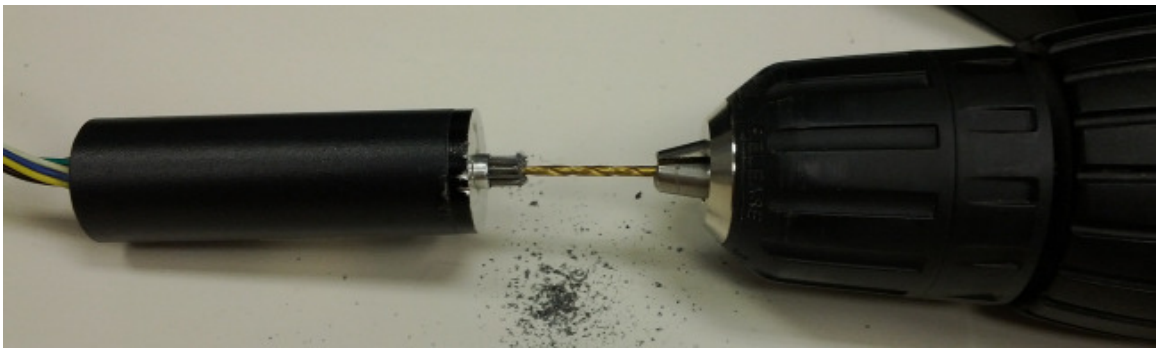


Figure 4.44: Removal of front pinion gear prior to disassembly

4.8 Thermal Vacuum Tests

To prepare a spare brushless drive motor for vacuum testing, the gearbox and front shaft were removed by unscrewing the long front screws holding the gearbox together. The motor itself was then disassembled by removing the back cover and pulling out the stator. The covers were glued in and the transmission gear on the shaft was riveted in place, which made disassembly difficult. As the front cover could not be removed, it was necessary to drill out the shaft to remove the gear and stator from the back of the motor, as shown in Figure 4.44. This meant that the gearbox could not be included in testing.

The disassembled motor housing, stator, and bearings were separated as shown in Figure 4.45 then soaked in a series of baths in an ultrasonic cleaner. A small flask containing the parts was submerged in water in the ultrasonic cleaner tray, and the flask was filled with solvents for intervals of 30 minutes at a time. First, a turpentine bath was used to clean the parts thoroughly and remove any traces of oils or polymers. Then an acetone bath was used to clean the turpentine and more oil from the parts. Three isopropyl alcohol baths were then used to clean the parts even further. Finally, two baths in distilled water cleaned any traces of alcohol remaining from the parts. The motor was then reassembled and successfully tested.

Monitoring of the hardware was done through TTL serial connection to the OBC with a terminal emulator. Connections were made to the DB-25 pass-through connector on the thermal vacuum chamber as follows:

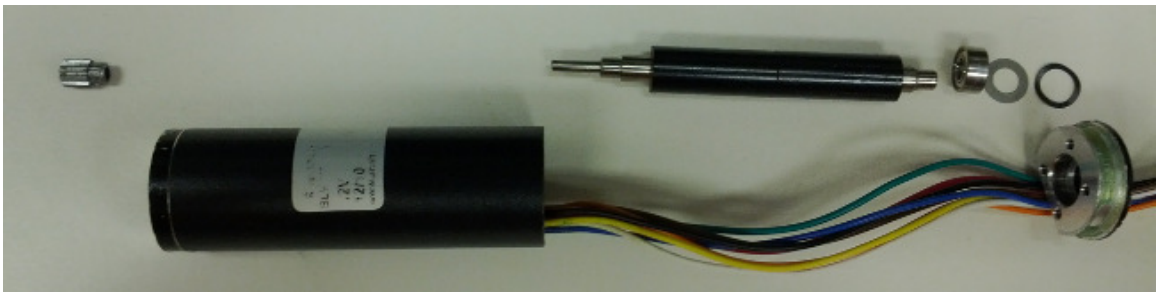


Figure 4.45: Disassembled brushless DC motor prior to cleaning

- Pin 2 and Pin 3: Motor Power - to 11.1V supply with readout
- Pin 5: OBC power - to 4.2V supply with readout
- Pin 6: serial TX at 115200 baud 8n1 - to TTL serial adapter RX pin
- Pin 7: serial RX at 115200 baud 8n1 - to TTL serial adapter TX pin
- Pin 18, pin 20, pin 24: Ground - to GND or V- of 11.1V supply and 4.2V supply

In preparation for the test, the motor and electronics were baked in a thermal chamber at 60 ^{o}C for 24 hours to remove all traces of moisture and oils, as shown in Figure 4.46.

When assembling the electronics for the T-VAC chamber, temperature probes were placed in the following locations, listed with peak rated temperatures:

- one on the AT91RM9200 microcontroller chip on OBC (max is $85^{o}C$)
- one on the motor driver closest to the opening where the Teflon wires exit (max is $150^{o}C$)
- one on the body of the motor (estimated max is $150^{o}C$)
- one just inside of the enclosure (max is estimated at $85^{o}C$)

The components were placed next to nanosatellite components for testing as shown in Figure 4.47.

The thermal vacuum test procedure was originally stated as four days, with alternating 24 hour periods of a $-40^{o}C$ cold soak and a $60^{o}C$ hot soak. As manual control of the chamber was

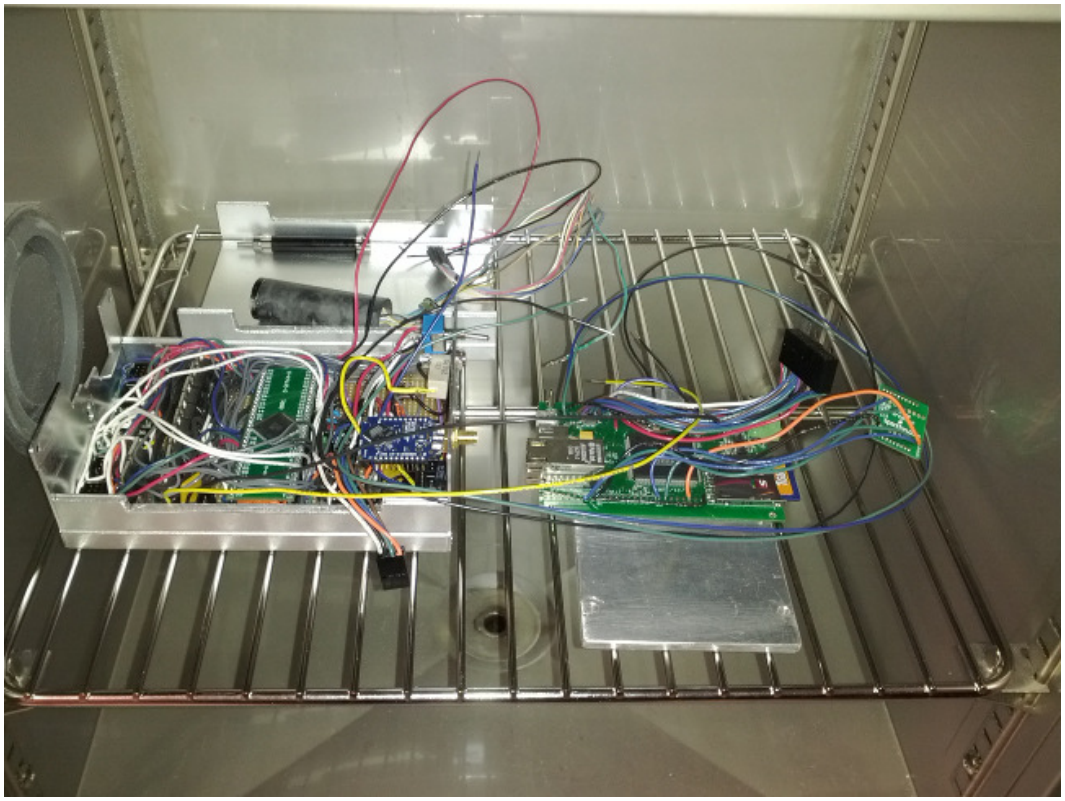


Figure 4.46: Baking of electronics and motor components in preparation for thermal vacuum

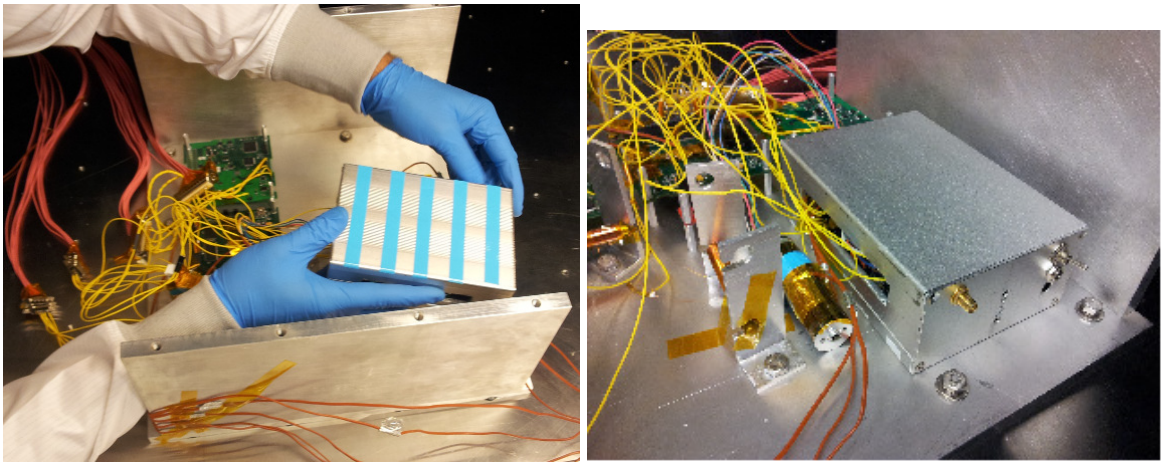


Figure 4.47: Placement of electronics and motor for thermal vacuum testing

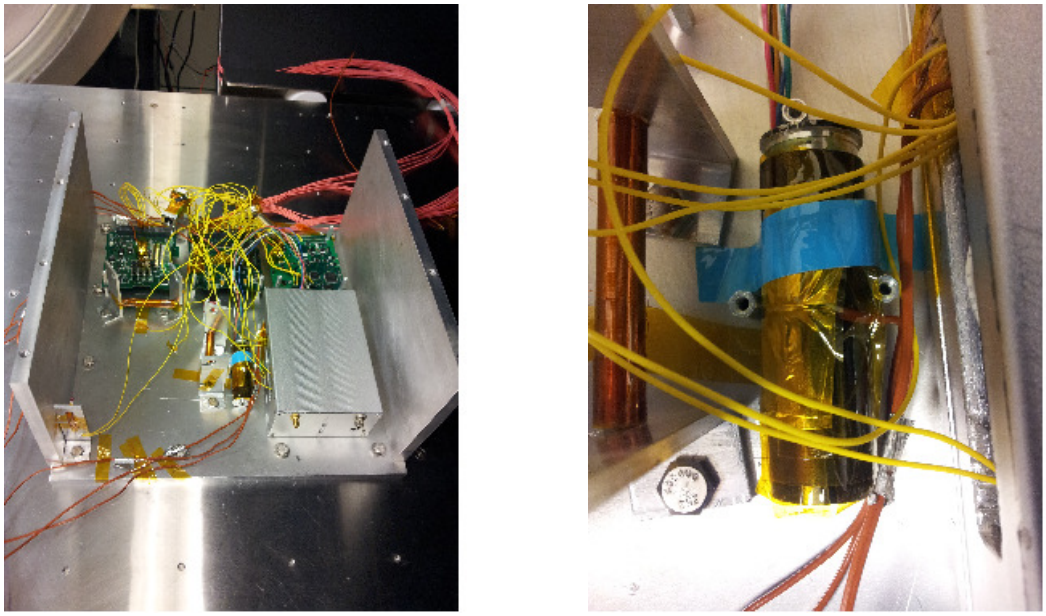


Figure 4.48: μrover and nanosatellite components on thermal vacuum chamber tray

required and due to the complexity of accurately slewing the chamber temperature to constant values over long periods, some variations in timing and temperature were observed. As the motors are not intended to be used in space where high heat is encountered, motor tests were only run at -40^oC, while OBC tests were run at both -40^oC and 60^oC. Data points from the temperature sensors were manually recorded at 10 minute intervals. Also, the $4.2V$ and $11.1V$ supply currents were monitored to ensure they stayed at less than $250mA$, or a hardware fault would be probable. The μrover and nanosatellite components with the motor on the tray of the thermal vacuum chamber are shown in Figure 4.48.

The motor test is designed to be simply a slow start, 30 minute run at full power, and slow stop. The test program is written in C and sends commands to the motor driver to change speed and read back the motor drive status. It is started with the command "./runmotor" It normally runs for 34 minutes, but should be aborted after 40 minutes if it does not stop by itself with the Ctrl-C keystroke. It will dump a logging file to the current directory with the test data in it.

The OBC test uses the GNU "stress" program, which is commonly used for stress testing of operating systems and hardware by creating multiple threads that fully use available computing

resources. As the micro-rover hardware is limited in capacity, only two CPU threads, four I/O threads, and one virtual memory thread of 8MB size were used, as more than this allotment would occasionally crash the OBC even under normal conditions. A 15 minute test time was used so that too much time was not taken from other tests required. The options for this test were used as:

./stress –cpu 2 –io 4 –vm 1 –vm-bytes 8M –timeout 15m –verbose

In case the program crashed or locked up, an alternate set of options for two CPU and two I/O threads was provided, as I/O and VM operations were the most troublesome.

./stress –cpu 2 –io 2 –timeout 15m –verbose

A script was created to start the program with the former options, and the test run at both -40^oC and 60^oC temperature cycles. The Ctrl-C keystroke can be used in the event of the program not finishing in 15 minutes. The resulting temperature data from the temperature sensors during thermal vacuum testing is shown in Figure 4.49.

It is evident that longer test cycles would result in higher final temperatures, but overall, the hardware performed well within expectations. The OBC tests showed well-defined temperature spikes that can be estimated to stabilize at approximately 85^oC in hot soak, the maximum rated temperature for the OBC hardware. The motor ran without any visible problems in vacuum, and the drivers appeared to heat the enclosure up by approximately 20^oC, while the motor itself experienced a temperature spike of up to 40^oC in cold soak.

While this thermal vacuum test was specified for orbital conditions due to the need to test the μrover at the same time as CubeSat hardware, it does have relevance to the atmospheric conditions on Mars. The surface pressure on Mars of $0.6kPa$ is not hard vacuum, but it is low enough that moving parts specified for operation on Earth should function similarly, and oil evacuation is necessary for the transit to Mars in hard vacuum regardless. Assuming operation of the rover in Martian summer at warm latitudes, temperatures can be expected to range between $18°C$ during the day and $-65°C$ at night [JPL/ASU 2013]. While the $25°C$ difference between the tested and expected minimum temperatures is a limitation, there is no indication that electronic hardware that operates perfectly at $-40°C$ will fail catastrophically or physically at $-65°C$. Extended

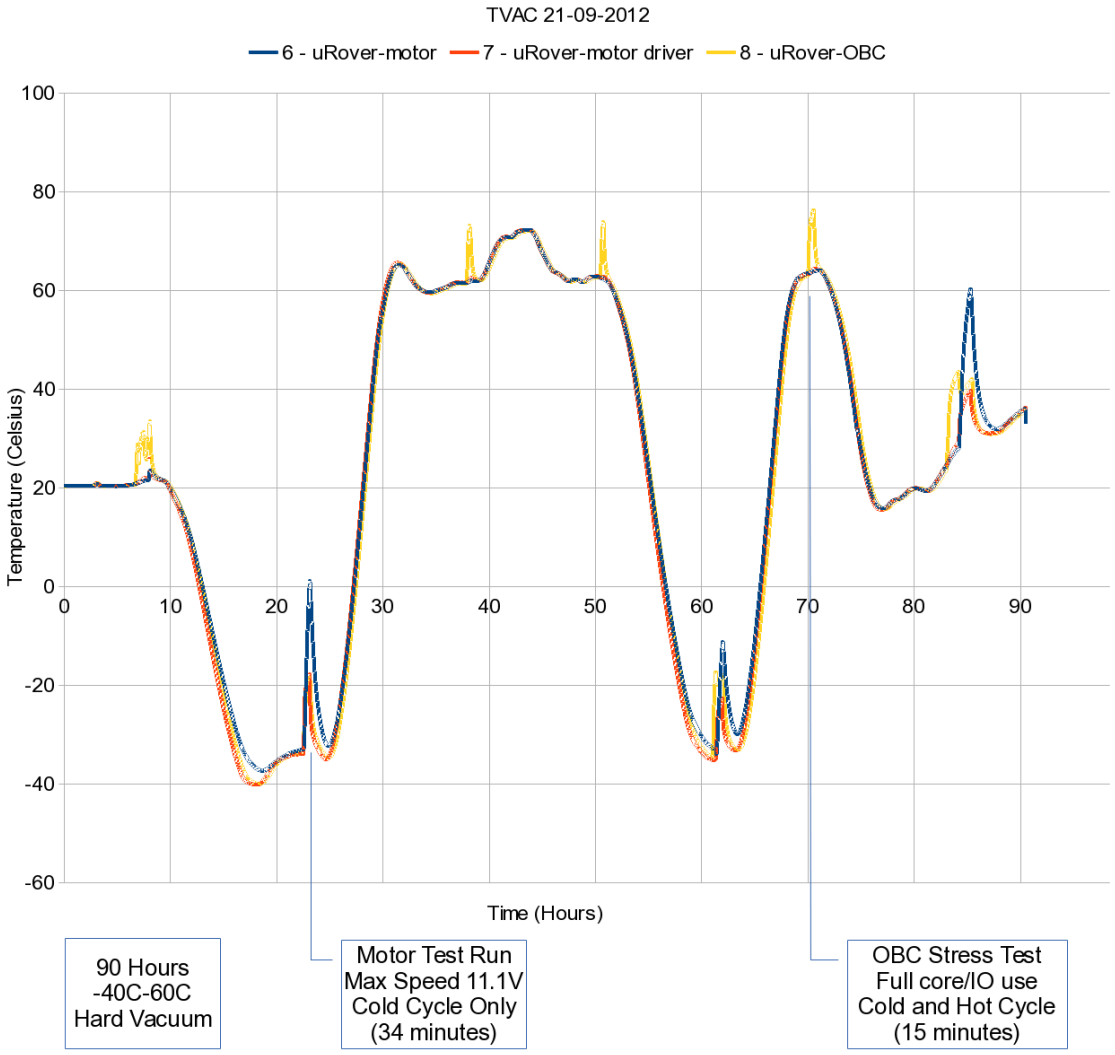


Figure 4.49: Resulting temperatures of μrover components during thermal vacuum testing

temperature range parts are also available that will operate down to $-55°C$ with very similar construction. It is therefore reasonable to assume that by powering-down the electronic hardware, it can survive a Martian night and be enabled by a light sensor or thermal switch during the daytime, similarly to how the MER were powered down to survive the Martian winters. Additional testing can verify this when the opportunity arises. While a longer test would be necessary to verify final temperatures and reliability, the high operability of the system in thermal vacuum indicates that operations on Mars would be possible with relatively little modification.

Chapter 5

Conclusion

This chapter concludes the dissertation and offers a summary of several potential areas for future development and improvement of the μrover technologies. We have examined the complete mechanical, electronic, and algorithmic design of a modern micro-rover that can perform simple exploration tasks autonomously in harsh environments, and shown that intelligent, reliable operation is feasible using innovative probabilistic techniques and custom-created hardware and software.

5.1 Mechanical Design

We have designed and constructed a series of μrover prototypes to validate the concept that a small, simple planetary rover can perform many of the tasks that larger rovers can, and for the most part these prototypes have performed well. A series of successive designs has allowed experimentation with many different design factors, and correction of common points of failure. The application of a minimalist philosophy in design has greatly simplified construction and maintainance, and minimizes moving parts that could be prone to failure in hostile environments. The suspension design that was ultimately used has kept the μrover stable and mobile in many outdoor environments while using a minimum of moving parts. Standardization of fasteners and sizes also simplifies construction.

The tubular chassis design has proved to be both lightweight and sturdy, and while being easy to fabricate, has the significant benefit of being able to contain and protect sensors, actuators, and wiring. Storage and deployment of the chassis is very simple also, involving a retaining strap that can be cut automatically. The $100mmx150mmx50mm$ electronics enclosures have been sufficient to enclose all electronic hardware required for operation of the μrover including batteries and drive controller, and two payload spaces are available for additional sensors and actuators such as rock drills or ground-penetrating radar, currently under development.

The kinematic analysis and parameterization done for the μrover suspension is the first analysis of this kind of four-arm independent suspension, and with the addition of intelligent control, provides extra flexibility in movement while allowing mechanical stability. While it does not appear to be an obviously stable system, the suspension has performed well on several different kinds of terrain even without intelligent control, and active sensing of the suspension arm angles allows the system to be stable when manoeuvring on complex terrain. A nonlinear type-1 or type-2 fuzzy controller provides intuitive, parameterized control of the movement of the μrover. Quaternions are used for storing the chassis rotation angles to avoid singularities.

In design of the drivetrain, evaluation of the performance of YURT rovers provided valuable confirmation of the metrics used to design the drive system, and using the current generation of

motors, the μrovers have proven to be capable in off-road movement. Though there are few motors suitable for μrover use, the $12V$ NEMA-09 brushless DC motors with a $1:139$ ratio provide more than sufficient torque for pulling the μrover over steep slopes and obstacles. However, they require a significant amount of the total power available. No mechanical or electrical failures were experienced in testing of the current system.

5.2 Electronics

The design approach we have taken in developing the μrover emphasizes modularity and expandability in hardware and software design. We use a centralized control topology, in particular with respect to OBC design and communication protocols, and a subsystem design methodology making use of highly-integrated commercial-off-the-shelf (COTS) parts and low-cost components. The methodology of connecting multiple programmable components by reliable serial buses and embedded communication methods has proven to be quite successful for mobile robot control.

For internal communications between components, SPI implementations have functioned very reliably and are planned for common use, although asynchronous serial still works quite well in general. The use of I^2C sensors has occasionally caused concerns, although the protocol has functioned well in most cases. Parallel synchronous interfaces for cameras and other high-bandwidth devices is practical. For external communications, the RS-485 serial standard has proven superior in all cases to RS-232, and is more accessible than many other resiliant physical bus standards. Radio communications via the $900MHz$ XBee radios has functioned well overall, although more control in the operation of the mesh network is preferable

The AT91RM9200 has fulfilled the role well for a low-power but computationally capable ARM microcontroller that can run Linux. Although more complex tasks such as image processing require more specialized hardware, tasks such as communications, control, navigation, and decision-making are easily accomplished within the software framework developed for the μrover.

The use of highly efficient down-conversion of power by distributed DC-DC converters has produced a reliable and practical modular power architecture. The evaluation of many different converter ICs is also useful information for future designs. The μrover produces more than a sufficient amount of power for onboard components from its solar arrays, and pulse-charging for Li-Ion battery cells from solar power works more reliably and efficiently than buck or boost architectures. Balanced charging of an asymmetric stack of Li-Ion cells by a set of charge pumps allows a variety of bus voltages to be produced from a minimal battery footprint.

As there are very few motor controllers available that are suitable for planetary micro-rover use, the development of such a controller is an important contribution. The new motor controller provides software selectable brush or brushless DC motor management, very high current capacity, and low power operation in a small package. A single 8-bit microcontroller can provide adequate feedback control for four motors simultaneously, and allows a high degree of flexibility in controller design.

For navigation, we have implemented and compared two useful methods for coarse solar angle sensing. Using only simple hardware and embedded software implementation, very coarse attitude estimation results can be achieved using either photodiode array or solar panel current measurement methodologies for nanosatellite attitude tracking or μrover navigation. The photodiode array provides good overall accuracy to within $\pm 5^{\circ}$ without additional filtering and thus requires minimal processing but can be improved beyond this measure if additional filtering is implemented. Dual-axis sensing is possible for a linear array using an N-slit configuration, but precise construction of the slit is essential and transverse angular measurements are more limited. Solar panel current measurements without the use of a discrete sensor can provide angular approximations over the entire exterior of the vehicle to $\pm 7^{\circ}$, but require significant filtering and averaging of measurements, and thus tend to be less accurate and more processing-intensive.

The $500MHz$ Blackfin-based vision board developed to augment the visual navigation abilities of the μrover is still under software development, but algorithms being tested are showing promise in being able to extract visual information from one or two cameras in real time for navigation and decision-making use.

5.3 Programming, Control and Communications

A software architecture for the μrover based on open components was developed with modularity in mind. By using freely-available and open software and standards, the use of a collaborative environment and interoperability with a wealth of software and libraries was leveraged to greatly increase the flexibility of the μrover software platform. Real-time operating systems have been compared in depth for use on the μrover, and a dedicated mesh networking structure was adopted to accommodate efficient point-to-point messaging communications between devices, components, and vehicles.

A library for fixed-point mathematical operations was developed specifically for use on the resource-constrained microcontrollers used for μrover control, and provides acceptable precision with much more efficient execution for most programming applications. The use of fixed-point mathematics is not a replacement for a dedicated floating-point unit or emulation, as the limits in representation and precision must still be respected at all times. However, if numerical boundedness and precision are acceptable within the programs using it, fixed-point math is an acceptable substitute to obtain significantly more efficient programs.

A layered communications and message-passing architecture was designed to fill the need for a communications infrastructure suitable for small planetary robots. The system has worked reliably thus far on the μrovers and the YURT rovers it has been tested on. By using a dedicated communications implementation, networking packet overhead has been reduced to 16 bytes (128 bits) compared with systems such as TCP/IP over Ethernet which has a minimum 64-byte packet size. Simple operations require only a 20 byte packet in total (160 bits), and processing is easily possible on 8-bit microcontrollers which would otherwise require significant effort to decode more complex networking protocols.

To improve estimation and accuracy on board the μrover, several unscented Kalman filtering algorithms, including the recently-developed cubature Kalman filter were extensively developed and tested for use in fixed-point implementation. The addition of noise covariance adaptivity functions, while providing minimal difference in simulation, has the potential to significantly

improve filter performance in the long term for sensors with limited reliability. A reduced sigma-point set can also decrease the computational time required in filtering for some implementations.

To summarize the results of RTOS testing, the real-time performance of Xenomai is better than Linux mainly in terms of determinism and low jitter rather than fast execution. Bare-hardware ARM programming is more difficult and not necessarily better if drivers and multiple processes are needed, but provides the extreme in fast performance, while bare-metal soft-core Microblaze performed similarly but much more slowly. Based on other platforms, the performance of RTEMS is expected to be fast and in between these two extremes. For deeply embedded mission-critical applications where deterministic performance and simplicity is essential and the facilities provided by Linux are not needed, RTEMS or a similar RTOS such as FreeRTOS is preferable to bare-hardware programming. If a larger and more complex kernel with longer latencies can be tolerated, real-time Linux using the Xenomai patch provides the easiest development and greatest flexibility, and is in all cases preferable to non-RT Linux if the kernel version is suitable.

5.4 Probabilistic Intelligence

Improving the reasoning capability on modern robots is an important part of making small planetary rovers more independent of humans. One of the greatest challenges for small planetary rovers is being able to make intelligent, appropriate decisions in the absence of expert knowledge in environments that are inherently unknown and uncertain. Probabilistic methods are a good solution to this problem, providing a way to interpret large interrelated sets of data logically and choose more likely options while explaining away others. For this reason, we base the autonomy and machine intelligence of the μrover on Bayesian networks that are constructed from both prior knowledge and known relationships between hardware components and abstract concepts.

A comprehensive description of how statistical and probabilistic methods are applied to the μrover has been given, with particular emphasis on the practicality of probability distributions with random variables as a way of representing abstracted data and making queries in a robotic system. Making decisions using logical propositions is described within the context of the Bayesian Robot Programming paradigm. We have applied this paradigm specifically to the Bayesian network structure, leading to an efficient and intuitive way of representing robotic knowledge and drawing conclusions based on statistical information encapsulated in the network.

We have implemented a novel, efficient, structured framework for practical use of Bayesian networks and probabilistic queries. By building networks of the four different kinds of nodes with either discrete or continuous probability distributions, any set of information can be obtained with relevance dependent on the accuracy of the distributions represented. The programming is efficient enough that queries can be made at high speed even on the embedded μrover hardware. The applicability of these methods to robotic programming and control is effectively endless, and many future applications are expected.

Using this framework, we have proven the feasibility of Bayesian methods on embedded micro-rover hardware for processing and mapping statistical data from a basic set of sensors. Results show that statistical methods can be used effectively with a simple sensor set and embedded hardware to provide systematic mapping and obstacle avoidance algorithms for outdoor naviga-

tion. This method can be extended to much more complex systems simply by adding sensors and their appropriate networked random variables.

5.5 Vision

To provide visual sensing and navigation capabilities to resource-constrained platforms such as the μrover, a complete, efficient, vision processing pipeline was programmed using both customized C++ routines and OpenCV library functions. Both optical flow and feature triangulation methods were investigated with ego-motion estimation and SLAM capability as the goal.

Optical flow methods for both downward-pointing sensors and cameras were researched. Sensors such as those from optical mice can provide very high short-distance accuracy, but with restrictions on the surfaces and focal distances allowed. Camera optical flows are less accurate due to inconsistencies in flow estimation that can be caused by uniform or indistinguishable surfaces, but are able to supplement ego-motion estimation made by other visual methods. Ego-motion reconstruction is complex, but useable. Optical flows are also generally less computationally expensive than other methods.

The ORB algorithm that combines FAST keypoint detection and BRIEF feature descriptors provides an efficient and free alternative to the more well-known SIFT and SURF algorithms, providing good tolerance to rotation and scaling of features. For useful reconstructions later, it is important to get as many features as possible. For this reason, complex scenes with many different colors, edges, and shapes generally provide the most useful image information for feature-based systems such as this.

Using projective geometry, we have successfully performed three-dimensional reconstruction of feature points on obstacles from a sequence of images taken with a single camera. With a continuous sequence of images, we can obtain point clouds that indicate ground and obstacle positions relative to the camera and map them using probabilistic methods.

5.6 Environmental Testing

Thermal vacuum testing of the μover onboard computer and drive system has established that operation in space conditions is possible. The use of an oil-evacuated brushless motor has proven that relatively unspecialized but autoclaveable brushless motors can provide short-distance mobility for the μrover. The electronics do not heat past their design tolerances during normal operation even in hard vacuum. Noticeable heat buildup is possible during continuous operation of the drive motors, but not to destructive levels under the assumption that drive will not be used during periods of heating in space. On Mars, the thin atmosphere will still provide better cooling performance than hard vacuum. It is also expected that under surface pressure of $0.6kPa$ and summer conditions of $18°C$ to $-65°C$, the μrover can carry out mission operations, with the potential modification of nighttime power-down, so no significant environmental problems are expected for an operation such as the Northern Light mission.

5.7 Summary

In this dissertation, we have presented the Beaver μrover, a 6kg solar-powered planetary rover that is designed to perform exploration missions such as the Northern Light Mars mission, as well as extending the capabilities of modern robotics here on Earth. By developing systems from the ground up using a pragmatic design approach, commercial hardware, open-source software, and novel implementations of probabilistic algorithms, we have obtained a comprehensive set of hardware and software designs that can form the basis for many kinds of intelligent but low-cost robots. It is our hope that this work will help to pave the way to a future of accessible, commonplace exploration robots that can perform missions everywhere for the benefit of human knowledge and curiosity.

With respect to the original goals of this project, we have accomplished the following key objectives:

1. The μrover prototype has been built from the ground up on modern, commercially-available components selected for optimal functionality and robustness. Independent and reliable operation in outdoor environments with a sufficient number of sensors for autonomy and basic navigation has been successful, and environmental testing indicates that operation in space and particular Martian conditions would be possible.

2. Based on probabilistic logic and Bayesian networks, the framework for intelligent navigation and decision-making has been developed to be suitable for embedded fixed-point hardware, and a vision system that is feasible for use on a small, low-power monocular camera system has been developed using structure-from-motion methods.

3. Using the task of autonomous mapping and obstacle avoidance as an example of simple planetary operations, a Bayesian network for motor control and map building from infrared range sensors was constructed and tested successfully, and a three-dimensional obstacle and rover location map was created using only monocular sequential imaging, both in a real outdoor environment.

The hardware and software developed in this work is by design, flexible and applicable to a wide variety of future space hardware projects, and the framework for probabilistic autonomy and embedded intelligence will form the basis of future work on machine intelligence and learning. The many useful products of this research include:

1. Electronic designs for the embedded on-board computer, inertial and external sensor systems, multi-cell battery charging and power distribution systems, and flexible high-current DC motor drive system

2. A multi-point, multi-layered robust packet communications architecture that can be adapted for many purposes and on many physical systems

3. The fixed-point embedded matrix mathematics, Kalman filtering, and probabilistic calculation functions that are useable on even highly-constrained computing platforms

4. An API for constructing Bayesian networks and performing inference and learning efficiently and robustly based on uncertain quantities and the structure of the rover hardware

5. An embedded vision DSP board and efficient algorithms for sequential localization and mapping from a monocular sequence of images.

These are enabling technologies for future work across many areas of mobile robotics and intelligent machine systems, and will be well-leveraged in future research for creating the next generation of capable and robust space hardware technologies.

5.8 Future Directions

5.8.1 Modular Electronics

In future, we plan to implement the modular μrover hardware in many other projects for both space and terrestrial use such as unmanned aerial or underwater vehicles, distributed sensor networks, small satellites, and a variety of other robotic or autonomous applications.

5.8.2 Multi-Robot Coordination

The existing μrover is capable of communicating and coordinating with others of its kind, but the use of many rovers navigating simultaneously has not been examined yet. Additional rovers will be built in future work so that the use of distributed Bayesian learning and probabilistic coordination of multiple agents can be developed and tested in real environments.

5.8.3 Visual Navigation

In future work, localization and bearing information will be used to determine which images are used for keypoint matching, to increase the amount of information that can be used for characterizing a specific location.

5.8.4 Reliable Computing

Although ARM and AVR-based microcontrollers have performed well under a range of conditions, it is preferable for space applications to use reliability critical architectures such as the ARM Cortex-R4. The only implementation currently available is the TI TMS570 and Hercules safety MCUs, designed to use ECC on all RAM and incorporate two cores in lock-step to immediately identify a calculation error. The TMS570 is being considered for future use in μrovers.

5.8.5 Embedded Communications

The LVDS voltage standard, which specifies lower voltages and higher communications rates, is being considered for use in the future as a replacement for the RS-485 electrical standard, as it also provides differential operation of serial devices and is used in the SpaceWire specification for space hardware devices.

5.8.6 Data Formats

Rather than using XML for all data communications, to reduce both space required and decoding overhead, the use of YAML (YAML Ain't Markup Language) is proposed for a next step. YAML is more human-readable than XML, can be serialized into a byte stream just as easily, and is widely supported with several libraries available in different Languages [Ben-Kiki *et al.* 2009].

5.8.7 Radio Communications

The development of an open-hardware transceiver module and purpose-designed mesh network protocol for space hardware systems would be very beneficial. For higher-power radio systems, an OBC daughterboard can be used. The advantage of using simple serial communications rather than Ethernet, Wireless LAN, or other IP communications standards, is that a wider variety of radio systems and communications frequencies can be used, up to and including S-band and UHF radio systems for space hardware.

5.8.8 Distributed Processing

It is hoped that the communications infrastructure described here can be developed into a general message-passing interface for communications between distributed components. If high enough efficiency can be achieved, a distributed operating system running on the connected components could be developed as a high-reliability alternative to centralized control, effectively allowing a robot with many small parts or even several interconnected robots to operate as a single entity.

5.8.9 Operating Systems

As a platform for modular, high-reliability, and distributed computing operations, the use of RTEMS as an embedded real-time OS will be further developed and improved for space use. The use of Xenomai real-time Linux will be retained as a development platform and testing system for ease of software development. The use of distributed processing through robust and flexible real-time messaging will allow RTEMS and Xenomai platforms to seamlessly co-operate within and across connected robotic systems.

5.8.10 Bayesian Processing

The practicality of probabilistic representation of real quantities makes the use of a probability-based general computing system promising. One potential implementation is a programming architecture that mirrors conventional computing, with random variables replacing exact variables and statistical operations replacing exact mathematical operations. Such an architecture could also operate as a fuzzy system if inference is not used, so the flexibility of the system could potentially be quite large. Such an implementation will be designed and evaluated in future work, and also used as part of the inference system.

5.8.11 Bayesian Learning

Currently, map traversal and obstacle learning form most of the machine learning capabilities of the system, but the use of probabilistic learning techniques is flexible enough for application to a variety of other variables as well, and will be further explored. In particular, adaptability to unexpected system conditions and tolerance to hardware and software faults by means of probability distribution adaptation similar to that use in a Kalman filter is an area of interest.

5.8.12 Nonparametric Bayesian Processes

Although more research is necessary to determine the best methodology for constructing the network and implementing machine learning, one promising area is that of empirical learning using

Bayesian Field Theory. Rather than using parametric approaches, for example with parameters ξ to evaluate $P(y|x,\xi)$, a nonparametric approach is used. The function $\phi(x,y)$ represents a field of dimension that includes all possible values of x and y in $0 <= P(y|x,\phi)$ for all x and y, and can represent the likelihood PDF $P(y|x,\phi)$ itself. These nonparametric "free form" approaches are adaptable and flexible. Functions of continuous variables $\phi(x,y) = P(y|x,\phi)$ are known as fields in physics, and the a priori information for a Bayesian method is represented by a stochastic process $P(\phi) = P(\phi(X = x, Y = y))$ like those in physical field theory, hence "Bayesian Field Theory" [Lemm 2003].

References

[A. H. Mohamed 1999] A. H. Mohamed, K. P. Schwarz, "Adaptive Kalman filtering for INS/GPS," *Journal of Geodesy*, Vol. 73, No. 4, May, 1999, Pages: 193–203. 185

[Ackerman 2013] Ackerman, Evan. "NASA Lets Curiosity Rover Loose on Mars in Autonomous Driving Mode," . Online, http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/nasa-mars-curiosity-rover-autonomous-driving-mode, accessed 2013/11/25 2013. 49

[Aharonov *et al.* 1988] Aharonov, Yakir, Albert, David Z., & Vaidman, Lev, "How the result of a measurement of a component of the spin of a spin-1/2 particle can turn out to be 100," *Phys. Rev. Lett.*, Vol. 60, Apr, 1988, http://link.aps.org/doi/10.1103/PhysRevLett.60.1351 Pages: 1351–1354. 53

[Altemose 2008] Altemose, George, "Achieving cell balancing for lithium-ion batteries," *Hearst Electronics Products*, 2008. 124

[Amato *et al.* 2012] Amato, J.L., Anderson, J.J., Carlone, T.J., Fagan, M.E., Stafford, K.A., & Padir, T., "Design and experimental validation of a mobile robot platform for analog planetary exploration," . In: *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 2012, Pages: 2686–2692. 14

[Ambrosch *et al.* 2009] Ambrosch, K., Zinner, C., & Kubinger, W., "Algorithmic Considerations for Real-Time Stereo Vision Applications," . In: *MVA09*, 2009, Page: 231. 271

[Anderson & Dorfman 1991] Anderson, Christine, & Dorfman, Merlin, *Aerospace software engineering: a collection of concepts*, Vol. 136, AIAA, 1991. 156

[Andert 2009] Andert, F., "Drawing stereo disparity images into occupancy grids: Measurement model and fast implementation," . In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009, Pages: 5191–5197. 284

[Arasaratnam & Haykin 2008] Arasaratnam, I., & Haykin, S., "Square-Root Quadrature Kalman Filtering," *Signal Processing, IEEE Transactions on*, Vol. 56, No. 6, 2008, Pages: 2589–2593. 46

[Arasaratnam & Haykin 2009] Arasaratnam, Ienkaran, & Haykin, Simon, "Cubature Kalman Filters," *IEEE Transactions on Automatic Control*, Vol. 54, No. 6, 2009, Pages: 1254–1269. 46, 189

[Arasaratnam & Haykin 2011] Arasaratnam, Ienkaran, & Haykin, Simon, "Cubature Kalman Smoothers," *Automatica*, Vol. 47, No. 10, October, 2011, http://dx.doi.org/10.1016/j.automatica.2011.08.005 Pages: 2245–2250. 46

[Arasaratnam *et al.* 2007] Arasaratnam, I., Haykin, Simon, & Elliott, R.J., "Discrete-Time Nonlinear Filtering Algorithms Using Gauss-Hermite Quadrature," *Proceedings of the IEEE*, Vol. 95, No. 5, 2007, Pages: 953–977. 45

[Arasaratnam *et al.* 2010] Arasaratnam, I., Haykin, Simon, & Hurd, T.R., "Cubature Kalman Filtering for Continuous-Discrete Systems: Theory and Simulations," *Signal Processing, IEEE Transactions on*, Vol. 58, No. 10, 2010, Pages: 4977–4993. 46

[Atmel 2012] Atmel. "ATmega640/1280/1281/2560/2561 Complete Datasheet," . Atmel Corporation, http://www.atmel.com/Images/doc2549.pdf 10, 2012. 143

[Bacchus *et al.* 1998] Bacchus, Fahiem, Halpern, Joseph Y., & Levesque, Hector J., "Reasoning about Noisy Sensors and Effectors in the Situation Calculus," *Artificial Intelligence*, Vol. 111, 1998, Pages: 111–1. 57

[Bae & Kim 2010] Bae, J. H., & Kim, Y. D., "Attitude Estimation for Satellite Fault Tolerant System using Federated Unscented Kalman Filter," *International Journal of Aeronautical and Space Science*, Vol. 11, No. 2, 2010, Pages: 80–86. 180

[Bailey & Durrant-Whyte 2006] Bailey, Tim, & Durrant-Whyte, H., "Simultaneous localization and mapping (SLAM): part II," *Robotics Automation Magazine, IEEE*, Vol. 13, No. 3, 2006, Pages: 108–117. 47

[Bailey *et al.* 2012] Bailey, Jordan, Sahani, Shailja, Post, M. A., Lee, R., & Daly, Michael, "York University Rover Team: Space Education Beyond The Classroom, Developing Lasting Institutions For Hands-On Education," . In: *CASI ASTRO 2012*, Quebec City, Quebec, Canada, April 24-26, 2012. 6

[Bajracharya *et al.* 2008] Bajracharya, M., Maimone, M.W., & Helmick, D., "Autonomy for Mars Rovers: Past, Present, and Future," *Computer*, Vol. 41, No. 12, Dec, 2008, Pages: 44–50. 12

[Bares *et al.* 1989] Bares, J.E., Hebert, M., Kanade, T., Krotkov, E., Mitchell, T., Simmons, R., & Whittaker, W.L., "Ambler: an autonomous rover for planetary exploration," *Computer*, Vol. 22, No. 6, June, 1989, Pages: 18–26. 13

[Barlas 2004] Barlas, Fırat. *Design of a Mars Rover Suspension Mechanism*. Master's thesis, Izmir Institute of Technology, Izmir, Turkey, June, 2004. 12

[Bartlett *et al.* 2008] Bartlett, Paul, Wettergreen, David, & Whittaker, William, "Design of the scarab rover for mobility & drilling in the lunar cold traps," . In: *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, 2008. 12

[Ben-Kiki *et al.* 2009] Ben-Kiki, Oren, Evans, Clark, & Ingerson, Brian, "YAML ain't markup language (YAML)(tm) version 1.2," *YAML. org, Tech. Rep., September*, 2009. 359

[Bessiere *et al.* 2000] Bessiere, Pierre, Lebeltel, Olivier, Lebeltel, Olivier, Diard, Julien, Diard, Julien, Mazer, Emmanuel, & Mazer, Emmanuel, "Bayesian Robots Programming," . In: *Research Report 1, Les Cahiers du Laboratoire Leibniz, Grenoble (FR*, 2000, Pages: 49–79. 227, 228, 241, 245, 248

[Biesiadecki *et al.* 2007] Biesiadecki, Jeffrey J, Leger, P Chris, & Maimone, Mark W, "Tradeoffs between directed and autonomous driving on the mars exploration rovers," *The International Journal of Robotics Research*, Vol. 26, No. 1, 2007, Pages: 91–104. 49

[Biham & Seberry 2006] Biham, Eli, & Seberry, Jennifer, "Pypy: another version of Py," *eSTREAM, ECRYPT Stream Cipher Project, Report*, Vol. 38, 2006, Page: 2006. 158

[Blank *et al.* 2003] Blank, Douglas, Meeden, Lisa, & Kumar, Deepak, "Python Robotics: An Environment for Exploring Robotics Beyond LEGOs," . In: *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '03, New York, NY, USA, 2003, Pages: 317–321.
http://doi.acm.org/10.1145/611892.611996 ACM. 159

[Bornstein 2008] Bornstein, Dan, "Dalvik VM Internals," . In: *Google I/O Developer Conference*, Vol. 23, 2008, Pages: 17–30. 158

[Brahmachari & Sarkar 2011] Brahmachari, A.S., & Sarkar, S., "View Clustering of Wide-Baseline N-views for Photo Tourism," . In: *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on*, 2011, Pages: 157–164. 279

[Brown & Martin 2010] Brown, Jeremy H, & Martin, Brad, "How fast is fast enough? Choosing between Xenomai and Linux for real-time applications," . In: *proc. of the 12th Real-Time Linux Workshop (RTLWS'12)*, 2010, Pages: 1–17. 191

[Bruss & Horn 1983] Bruss, A. R., & Horn, B. K. P., "Passive Navigation," *Computer Vision, Graphics, and Image Processing*, Vol. 21, No. 1, January, 1983, Page: 3–20. 266

[Bruyninckx 2001] Bruyninckx, H., "Open robot control software: the OROCOS project," . In: *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, Vol. 3, 2001, Pages: 2523–2528 vol.3. 39

[Bujnak *et al.* 2011] Bujnak, Martin, Kukelova, Zuzana, & Pajdla, Tomas, "New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion," . In: *Proceedings of the 10th Asian conference on Computer vision - Volume Part I*, ACCV'10, Berlin, Heidelberg, 2011, Pages: 11–24. http://dl.acm.org/citation.cfm?id=1964320.1964324 Springer-Verlag. 277

[Campbell *et al.* 2005] Campbell, J., Sukthankar, R., Nourbakhsh, I., & Pahwa, A., "A Robust Visual Odometry and Precipice Detection System Using Consumer-grade Monocular Vision," . In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, Pages: 3421–3427. 263

[Carlone *et al.* 2013] Carlone, T.J., Anderson, J.J., Amato, J.L., Dimitrov, V.D., & Padir, T., "Kinematic Control of a Planetary Exploration Rover over Rough Terrain," . In: *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, Oct, 2013, Pages: 4488–4493. 97

[Carpin *et al.* 2007] Carpin, S., Lewis, M., Wang, Jijun, Balakirsky, S., & Scrapper, C., "USARSim: a robot simulator for research and education," . In: *Robotics and Automation, 2007 IEEE International Conference on*, April, 2007, Pages: 1400–1405. 40

[Chen & Genta 2012] Chen, Feng, & Genta, Giancarlo, "Dynamic modeling of wheeled planetary rovers: A model based on the pseudo-coordiates approach," *Acta Astronautica*, Vol. 81, No. 1, 2012, http://www.sciencedirect.com/science/article/pii/S0094576512002469 Pages: 288 – 305. 14

[Chen *et al.* 2011] Chen, Feng, Genta, G., & Zhao, Guang, "Dynamic modelling of wheeled planetary rovers," . In: *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, Vol. 4, 2011, Pages: 1667–1671. 14, 79

[Cheshire & Baker 1999] Cheshire, Stuart, & Baker, Mary, "Consistent Overhead Byte Stuffing," *IEEE/ACM Trans. Netw.*, Vol. 7, No. 2, April, 1999, http://dx.doi.org/10.1109/90.769765 Pages: 159–172. 173

[Claraco 2008] Claraco, Jose Luis Blanco, "Development of Scientific Applications with the Mobile Robot Programming Toolkit," *The MRPT reference book. Machine Perception and Intelligent Robotics Laboratory, University of Málaga, Málaga, Spain*, 2008. 40

[Clark 1973] Clark, Ronald W, *Einstein, the Life and Times*, Hodder and Stoughton, London, 1973. 53

[Closas & Fernandez-Prades 2010] Closas, P., & Fernandez-Prades, C., "The marginalized square-root Quadrature Kalman Filter," . In: *Signal Processing Advances in Wireless Communications (SPAWC), 2010 IEEE Eleventh International Workshop on*, 2010, Pages: 1–5. 46

[Cover 1999] Cover, Robin. "XML Belief Network File Format," , http://xml.coverpages.org/xbn.html April 14, 1999. 240

[Cox 1961] Cox, R. T., *The Algebra Of Probable Inference*, John Hopkins University Press, 1961. 55

[Cummins *et al.* 2008] Cummins, John, Azhar, MQ, & Sklar, Elizabeth, "Using Surveyor SRV-1 Robots to Motivate CS1 Students," . In: *4th Artificial Intelligence for Interactive Digital Entertainment Conference, AIIDE*, Vol. 8, 2008, Pages: 22–24. 154

[Dagum & Luby 1993] Dagum, Paul, & Luby, Michael, "Approximating probabilistic inference in Bayesian belief networks is NP-hard," *Artificial Intelligence*, Vol. 60, No. 1, 1993,

http://www.sciencedirect.com/science/article/pii/000437029390036B Pages: 141 – 153. 225

[Dean 2003] Dean, WH, "PyMite: A Flyweight Python Interpreter for 8-bit Architectures," . In: *First Python Community Conference, Washington DC*, 2003. 159

[Dhouib *et al.* 2012] Dhouib, Saadia, Kchir, Selma, Stinckwich, Serge, Ziadi, Tewfik, & Ziane, Mikal. "RobotML, a Domain-Specific Language to Design, Simulate and Deploy Robotic Applications," . In: *Simulation, Modeling, and Programming for Autonomous Robots*, edited by I. Noda, N. Ando, D. Brugali, & J. Kuffner, Vol. 7628 of *Lecture Notes in Computer Science*, Pages: 149–160. Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-642-34327-8_16 2012. 40

[Diankov & Kuffner 2008] Diankov, Rosen, & Kuffner, James, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 2008, Page: 79. 40

[Dille *et al.* 2009] Dille, Michael, Grocholsky, Benjamin P, & Singh, Sanjiv, "Outdoor Downward-facing Optical Flow Odometry with Commodity Sensors," . In: *Proceedings Field & Service Robotics (FSR '09)*, July, 2009. 261

[Ding *et al.* 2011] Ding, Liang, Gao, Haibo, Deng, Zongquan, & Li, Weihua, "Advances in simulation of planetary wheeled mobile robots," *Mobile Robots-Current Trends*, 2011, Pages: 375–402. 40

[Dubois & Prade 2001] Dubois, Didier, & Prade, Henri, "Possibility Theory, Probability Theory and Multiple-Valued Logics: A Clarification," *Annals of Mathematics and Artificial Intelligence*, Vol. 32, No. 1-4, 2001, http://dx.doi.org/10.1023/A%3A1016740830286 Pages: 35–66. 56

[Dur 2009] Dur, E., "Optical Flow-based obstacle detection and avoidance behaviors for mobile robots used in unmaned planetary exploration," . In: *Recent Advances in Space Technologies, 2009. RAST '09. 4th International Conference on*, 2009, Pages: 638–647. 264

[Durrant-Whyte & Bailey 2006] Durrant-Whyte, Hugh, & Bailey, Tim, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, Vol. 2, 2006, Page: 2006. 47

[Durrant-Whyte *et al.* 1996] Durrant-Whyte, H., Rye, D., & Nebot, E., "Localisation of automatic guided vehicles," . In: *Robotics Research: The 7th International Symposium (ISRR'95)*. Springer Verlag, October, 1996, Page: 613–625. 47

[Edmondson *et al.* 2005] Edmondson, Kenneth M, Joslin, David E, Fetzer, Chris M, King, Richard R, Karam, Nasser H, Mardesich, Nick, Stella, Paul M, Rapp, Donald, & Mueller, Robert, "Simulation of the Mars surface solar spectra for optimized performance of triple junction solar cells," . In: *19th SPRAT Conference, Cleveland, OH, September 20-22, 2005*. Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2005., 2005. 299

[El Shobaki 2001] El Shobaki, Mohammed. "Non-intrusive hardware/software monitoring for single-and multiprocessor real-time systems," . Rapport technique, Citeseer, 2001. 203

[Engelbrecht 2002] Engelbrecht, Andries P., *Computational Intelligence: An Introduction*, John Wiley & Sons, 2002. 50, 214

[Estier *et al.* 2000] Estier, Thomas, Piguet, Ralph, Eichhorn, Roland, & Siegwart, Roland, "Shrimp, a Rover Architecture for Long Range Martian Mission," . In: *Proceedings of the Sixth ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA'2000)*, 2000, Pages: 5–7. 12, 76

[Estlin *et al.* 2011] Estlin, Tara, Gaines, Daniel, Chouinard, Caroline, Fisher, Forest, Castano, Rebecca, Judd, Michele, Anderson, Robert C., , & Nesnas, Issa, "Enabling Autonomous

Rover Science Through Dynamic Planning and Scheduling," . In: *IEEE Aerospace Conference (IAC 05). Big Sky, MT. March 2005*, 2011. 62

[Evensen 1994] Evensen, Geir, "Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics," *Journal of Geophysical Research: Oceans*, Vol. 99, No. C5, 1994, http://dx.doi.org/10.1029/94JC00572 Pages: 10143–10162. 43

[Fang Lin Luo 2003] Fang Lin Luo, Hong Ye, *Advanced DC/DC Converters*, CRC Press, September, 2003. 126, 128

[Farneback 2001] Farneback, G., "Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field," . In: *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, Vol. 1, 2001, Pages: 171–177 vol.1. 263

[Farnebäck 2000] Farnebäck, Gunnar, "Fast and Accurate Motion Estimation using Orientation Tensors and Parametric Motion Models," . In: *In Proceedings of 15th IAPR International Conference on Pattern Recognition*, 2000, Pages: 135–139. 261, 262

[Farokh Irom 2008] Farokh Irom, Duc N. Nguyen. "Radiation Tests of Highly Scaled High Density Commercial Nonvolatile Flash Memories," . Rapport technique, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 2008. 35, 191

[Faye 2008] Faye, Jan. "Copenhagen Interpretation of Quantum Mechanics," . In: *The Stanford Encyclopedia of Philosophy*, edited by E. N. Zalta. Stanford University, http://plato.stanford.edu/archives/fall2008/entries/qm-copenhagen/ Fall 2008. 53

[Feng & Hung 2003] Feng, C. L., & Hung, Y. S., "A Robust Method for Estimating the Fundamental Matrix," . In: *In International Conference on Digital Image Computing*, 2003, Pages: 633–642. 274

[Fernandez-Prades & Vila-Valls 2010] Fernandez-Prades, C., & Vila-Valls, J., "Bayesian Nonlinear Filtering Using Quadrature and Cubature Rules Applied to Sensor Data Fusion for Positioning," . In: *Communications (ICC), 2010 IEEE International Conference on*, 2010, Pages: 1–5. 46

[Festl *et al.* 2012] Festl, Freya, Recktenwald, Fabian, Yuan, Chunrong, & Mallot, Hanspeter A, "Detection of linear ego-acceleration from optic flow," *Journal of Vision*, Vol. 12, No. 7, http://www.journalofvision.org/content/12/7/10.abstract 2012. 264

[Fischler & Bolles 1981] Fischler, Martin A., & Bolles, Robert C., "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, Vol. 24, No. 6, June, 1981, http://doi.acm.org/10.1145/358669.358692 Pages: 381–395. 273

[Francisco *et al.* 2012] Francisco, J. Delgado, Quero, Jose M., Garcia, Juan, Tarrida, Cristina L., Ortega, Pablo R., & Bermejo, Sandra, "Accurate and Wide-Field-of-View MEMS-Based Sun Sensor for Industrial Applications," *IEEE Transactions on Industrial Electronics*, Vol. 59, No. 12, 2012, Pages: 4871–4880. 31

[Furgale *et al.* 2011] Furgale, P., Enright, J., & Barfoot, T., "Sun Sensor Navigation for Planetary Rovers: Theory and Field Testing," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 47, No. 3, 2011, Pages: 1631–1647. 31

[Gallant *et al.* 2011] Gallant, Marc, Ellery, Alex, & Marshall, Joshua, "Science-influenced mobile robot guidance using Bayesian Networks," . In: *Electrical and Computer Engineering (CCECE), 2011 24th Canadian Conference*, 8-11 May, 2011, Pages: 001135 – 001139. 62

[Garcia 2008] Garcia, Gonzalo, "Use of Python as a Satellite Operations and Testing Automation Language," . In: *GSAW2008 Conference, Redondo Beach, California*, 2008. 159

[Gleick 1987] Gleick, James, *Chaos: Making a New Science*, Viking, New York, 1987. 52

[Gosling *et al.* 2005] Gosling, James, Joy, Bill, Steele, Guy, & Bracha, Gilad, *Java(TM) Language Specification, The (3rd Edition) (Java (Addison-Wesley))*, Addison-Wesley Professional, 2005. 157

[Grewal & Kain 2010] Grewal, M.S., & Kain, J., "Kalman Filter Implementation With Improved Numerical Properties," *Automatic Control, IEEE Transactions on*, Vol. 55, No. 9, 2010, Pages: 2058–2068. 45

[Grotzinger *et al.* 2012] Grotzinger, John P., Crisp, Joy, Vasavada, Ashwin R., Anderson, Robert C., Baker, Charles J., Barry, Robert, Blake, David F., Conrad, Pamela, Edgett, Kenneth S., Ferdowski, Bobak, Gellert, Ralf, Gilbert, John B., Golombek, Matt, Gómez-Elvira, Javier, Hassler, Donald M., Jandura, Louise, Litvak, Maxim, Mahaffy, Paul, Maki, Justin, Meyer, Michael, Malin, Michael C., Mitrofanov, Igor, Simmonds, John J., Vaniman, David, Welch, Richard V., & Wiens, Roger C., "Mars Science Laboratory Mission and Science Investigation," *Space Science Reviews*, Vol. 170, No. 1-4, 2012, http://dx.doi.org/10.1007/s11214-012-9892-2 Pages: 5–56. 12

[Grotzinger *et al.* 2013] Grotzinger, JP, Blake, DF, Crisp, J, Edgett, KS, Gellert, R, Gómez-Elvira, J, Hassler, D, Mahaffy, P, Malin, MC, Mitrofanov, IG, *et al.*, "Mars Science Laboratory: First 100 sols of geologic and geochemical exploration from Bradbury Landing to Glenelg," . In: *Lunar and Planetary Institute Science Conference Abstracts*, Vol. 44, 2013, Page: 1259. 12

[Group 2007] Group, The JAUS Working. "JAUS Reference Architecture Specification, Volume II, Part I," , version 3.3 edition, http://www.openjaus.com/support/jaus-documents June, 2007. 38

[Guowei Cai 2011] Guowei Cai, Ben M. Chen, Tong Heng Lee, *Unmanned Rotorcraft Systems*, Springer London, 2011. 89

[Hajek *et al.* 1995] Hajek, Petr, Godo, Lluis, & Esteva, Francesc, "Fuzzy Logic and Probability," . In: *In Uncertainty in Artificial Intelligence. Proc. of 11th conference*, 1995, Pages: 237–244. 56

[Hartley & Sturm 1997] Hartley, Richard I., & Sturm, Peter, "Triangulation," *Computer Vision and Image Understanding*, Vol. 68, No. 2, 1997, http://www.sciencedirect.com/science/article/pii/S1077314297905476 Pages: 146 – 157. 275, 276

[Hartley & Zisserman 2004] Hartley, R. I., & Zisserman, A., *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521540518, second edition, 2004. 48, 275, 276

[Hartley 1997] Hartley, Richard, "Self-Calibration of Stationary Cameras," *International Journal of Computer Vision*, Vol. 22, No. 1, 1997, http://dx.doi.org/10.1023/A%3A1007957826135 Pages: 5–23. 274

[Hawking 1999] Hawking, Stephen. "Does God Play Dice?," . Public Lecture, www.hawking.org.uk/does-god-play-dice.html, accessed November 2013 1999. 52

[Hebert *et al.* 1996] Hebert, P., Betge-Brezetz, S., & Chatila, R. "Probabilistic map learning: Necessity and difficulties," . In: *Reasoning with Uncertainty in Robotics*, edited by L. Dorst, M. Lambalgen, & F. Voorbraak, Vol. 1093 of *Lecture Notes in Computer Science*, Pages: 307–321. Springer Berlin Heidelberg, http://dx.doi.org/10.1007/BFb0013969 1996. 284

[Heeger & Jepson 1992] Heeger, DavidJ., & Jepson, AllanD., "Subspace methods for recovering rigid motion I: Algorithm and implementation," *International Journal of Computer Vision*, Vol. 7, No. 2, 1992, http://dx.doi.org/10.1007/BF00128130 Pages: 95–117. 266, 268

[Heisenberg 1927] Heisenberg, W., "Uber den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik," *Zeitschrift für Physik A Hadrons and Nuclei*, Vol. 43, No. 3, March, 1927, http://dx.doi.org/10.1007/bf01397280 Pages: 172–198. 53

[Hoag 1963] Hoag, David. "Apollo Guidance and Navigation Considerations of Apollo IMU Gimbal Lock," . Rapport technique, MIT Instrumentation Laboratory, April, 1963. 91

[Huntsberger *et al.* 2002] Huntsberger, T., Aghazarian, H., Cheng, Yang, Baumgartner, E.T., Tunstel, E., Leger, C., Trebi-Ollennu, A., & Schenker, P.S., "Rover autonomy for long range navigation and science data acquisition on planetary surfaces," . In: *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, Vol. 3, 2002, Pages: 3161–3168. 12

[Huntsberger 2009] Huntsberger, Terry, "Onboard Learning of Adaptive Behavior: Biologically Inspired and Formal Methods," . In: *Advanced Technologies for Enhanced Quality of Life Conference*, 2009, Pages: 152 – 157. 62

[Hussein & Stipanovic 2007] Hussein, I. I., & Stipanovic, D. M., "Effective Coverage Control for Mobile Sensor Networks With Guaranteed Collision Avoidance," *IEEE Transactions on Control System Technology*, Vol. 15, No. 4, 2007, Pages: 642–657. 248

[Hy *et al.* 2004] Hy, Ronan Le, Arrigoni, Anthony, Bessière, Pierre, & Lebeltel, Olivier, "Teaching Bayesian behaviours to video game characters," *Robotics and Autonomous Systems*, Vol. 47, No. 2–3, 2004, Pages: 177 – 185. http://www.sciencedirect.com/science/article/pii/S092188900400048X Robot Learning from Demonstration. 241, 245

[Infantes *et al.* 2011] Infantes, Guillaume, Ghallab, Malik, & Ingrand, Félix, "Learning the Behavior Model of a Robot," *Auton. Robots*, Vol. 30, No. 2, February, 2011, http://dx.doi.org/10.1007/s10514-010-9212-1 Pages: 157–177. 55

[Irani *et al.* 1994] Irani, M., Rousso, B., & Peleg, S., "Recovery of ego-motion using image stabilization," . In: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, 1994, Pages: 454–460. 264

[Jang 1997] Jang, Jyh-Shing Roger, *Neuro-Fuzzy and Soft Computing: a computational approach to learning and machine intelligence*, Prentice Hall, 1997. 50

[Jaynes 2003] Jaynes, E. T., *Probability Theory: The Logic of Science*, Cambridge University Press, http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0521592712 April, 2003. 55

[Jia *et al.* 2012] Jia, Bin, Xin, Ming, & Cheng, Yang, "The high-degree cubature Kalman filter," . In: *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 2012, Pages: 4095–4100. 46

[Johnson & Jasik 1984] Johnson, Richard C, & Jasik, Henry, "Antenna engineering handbook," *New York, McGraw-Hill Book Company, 1984, 1356 p. No individual items are abstracted in this volume.*, Vol. 1, 1984. 66

[JPL/ASU 2013] JPL/ASU, NASA. "Mars Day and Night Temperature," . Online, http://cmex.ihmc.us/data/catalog/SurfaceLayer/DayNightTemp.html, accessed 2014/05/01 2013. 342

[Julier 2002] Julier, S.J., "The scaled unscented transformation," . In: *American Control Conference, 2002. Proceedings of the 2002*, Vol. 6, 2002, Pages: 4555–4559 vol.6. 45

[Julier 2003] Julier, S.J., "The spherical simplex unscented transformation," . In: *American Control Conference, 2003. Proceedings of the 2003*, Vol. 3, 2003, Pages: 2430–2434 vol.3. 45

[K. Sarda 2010] K. Sarda, C. Grant, et al., "Canadian Advanced Nanospace Experiment 2 On-Orbit operations; Two years of pushing the nanosatellite performance envelope," . In: *Conference Proceedings of CASI ASTRO 2010, Toronto, Canada*, 2010. 29

[Kalman 1960] Kalman, Rudolph Emil, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, Vol. 82, No. Series D, 1960, Pages: 35–45. 42

[Kanatani 1993] Kanatani, Kenichi, "3-D interpretation of optical flow by renormalization," *International Journal of Computer Vision*, Vol. 11, No. 3, 1993, http://dx.doi.org/10.1007/BF01469345 Pages: 267–282. 268

[Katrin Pirker & Bischof 2011] Katrin Pirker, Matthias Rüther, Gerald Schweighofer, & Bischof, Horst, "GPSlam: Marrying Sparse Geometric and Dense Probabilistic Visual Mapping," . In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, Pages: 115.1–115.12. http://dx.doi.org/10.5244/C.25.115. 279, 284

[Keating *et al.* 1975] Keating, T.J., Wolf, P.R., & Scarpace, F.L., "An Improved Method of Digital Image Correlation," *PhEngRS*, Vol. 41, No. 8, August, 1975, Pages: 993–1002. 258

[Kim *et al.* 2012] Kim, Younkyu, Eom, Wesub, Lee, Joo-Hee, & Sim, Eun-Sup, "Design of Mobility System for Ground Model of Planetary Exploration Rover," *Journal of Astronomy and Space Sciences*, Vol. 29, 2012, Pages: 413–422. 14

[Kjærulff 2008] Kjærulff, Uffe B., *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*, Springer Science, 2008. 51

[Koenig *et al.* 1996] Koenig, Sven, Goodwin, Richard, & Simmons, ReidG. "Robot navigation with markov models: A framework for path planning and learning with limited computational resources," . In: *Reasoning with Uncertainty in Robotics*, edited by L. Dorst, M. Lambalgen, & F. Voorbraak, Vol. 1093 of *Lecture Notes in Computer Science*, Pages:

322–337. Springer Berlin Heidelberg,
http://dx.doi.org/10.1007/BFb0013970 1996. 55

[Koks 2008] Koks, Don. "Using Rotations to Build Aerospace Coordinate Systems," . Rapport technique, Defence Science and Technology Organisation Salisbury (Australia) Systems Sciences Lab, August, 2008. 90

[Koller & Friedman 2009] Koller, Daphne., & Friedman, Nir, *Probabilistic Graphical Models, Principles and Techniques*, MIT Press, Cambridge, Massachusetts, 2009. 206, 207, 213, 224, 248

[Konecny & Pape 1981] Konecny, G., & Pape, D., "Correlation techniques and devices(for photogrammetry)," . In: *(International Society for Photogrammetry, Congress, 14 th, Hamburg, West Germany, July 13-25, 1980.) Photogrammetric Engineering and Remote Sensing*, Vol. 47, 1981, Pages: 323–333. 258

[Kozma *et al.* 2008] Kozma, Robert, Huntsberger, Terry, Aghazarian, Hrand, Ilin, Roman, Tunstel, Eddie, & Freeman, Walter J., "Intentional Control for Planetary Rover SRR," *Advanced Robotics*, Vol. 22, No. 12, 2008, Pages: 1309 – 1327. 51

[Kubota *et al.* 2003] Kubota, Takashi, Kuroda, Yoji, Kunii, Yasuharu, & Nakatani, Ichiro, "Small, light-weight rover "Micro5" for lunar exploration," *Acta Astronautica*, Vol. 52, No. 2–6, 2003, Pages: 447 – 453.
http://www.sciencedirect.com/science/article/pii/S009457650200187X
Selected Proceedings of the 4th IAA International conference on Low Cost Planetary Missions. 12

[Kuipers 2000] Kuipers, J., "Quaternions and Rotation Sequences," . In: *Proceedings of the International Conference On Geometry, Integrability And Quantization*, 2000, Pages: 127–144. 91

[Kwiek *et al.* 2010] Kwiek, Dominique, Setchell, Chris, & Rossiter, Jonathan, "High Accuracy Non-Contact Optical Measurement Systems Embedded Applications With Imetrum," . In: *Programme of the First Annual Research Conference for the Engineering doctorate in Systems*, May 17, 2010, Page: 50. 154

[Kwok *et al.* 2005] Kwok, N. M., Dissanayake, G., & Ha, Q.P., "Bearing-only SLAM Using a SPRT Based Gaussian Sum Filter," . In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, Pages: 1109–1114. 279

[Laubach *et al.* 1998] Laubach, S.L., Burdick, J., & Matthies, L., "An autonomous path planner implemented on the Rocky 7 prototype microrover," . In: *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, Vol. 1, May, 1998, Pages: 292–297 vol.1. 12

[Lauha 2006] Lauha, Jetro. "The neglected art of Fixed Point arithmetic," . Presentation, ftp://chiptune.untergrund.net/users/in4kadmin/files/The_neglected_art_of_Fixed_Point_arithmetic_20060913.pdf 2006. 162

[Lazkano *et al.* 2007] Lazkano, E., Sierra, B., Astigarraga, A., & Martínez-Otzeta, J.M., "On the use of Bayesian Networks to develop behaviours for mobile robots," *Robotics and Autonomous Systems*, Vol. 55, No. 3, 2007, http://www.sciencedirect.com/science/article/pii/S0921889006001412 Pages: 253 – 265. 241

[Lebeltel & Bessière 2008] Lebeltel, Olivier, & Bessière, Pierre. "Basic Concepts of Bayesian Programming," . In: *Probabilistic Reasoning and Decision Making in Sensory-Motor Systems*, Pages: 19–48. Springer, http://hal.archives-ouvertes.fr/hal-00338715 2008. 233

[Lebeltel *et al.* 2000] Lebeltel, Olivier, Diard, Julien, Bessiere, Pierre, & Mazer, Emmanuel, "A Bayesian framework for robotic programming," . In: *Twentieth International Workshop*

on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MaxEnt 2000), Paris, France, http://hal.archives-ouvertes.fr/hal-00089153 2000. 59, 227, 230, 242

[Lebeltel *et al.* 2004] Lebeltel, Olivier, Bessière, Pierre, Diard, Julien, & Mazer, Emmanuel, "Bayesian Robot Programming," *Autonomous Robots*, Vol. 16, No. 1, 2004, Pages: 49–79. 59, 227, 231

[Lee *et al.* 2003a] Lee, J.B., Choi, J.H., Chung, J.K., & Lim, J.H., "Design and implementation of integrated drive circuit for a small BLDC motor," . In: *Electrical Machines and Systems, 2003. ICEMS 2003. Sixth International Conference*, Vol. 2, November, 2003, Pages: 491 – 494 vol.2. 30

[Lee *et al.* 2003b] Lee, JB, Choi, JH, Chung, JK, & Lim, JH, "Design and implementation of integrated drive circuit for a small BLDC motor," . In: *Electrical Machines and Systems, 2003. ICEMS 2003. Sixth International Conference on*, Vol. 2. IEEE, 2003, Pages: 491–494. 139

[Lee *et al.* 2012] Lee, R., Chesser, Hugh, Cannata, Mathew, Post, M. A., & Kumar, K. D., "Modular Attitude Control System Design For CubeSat Application," . In: *CASI ASTRO 2012*, Quebec City, Quebec, Canada, April 24-26, 2012. 6

[Leger *et al.* 2005] Leger, P Chris, Trebi-Ollennu, Ashitey, Wright, John R, Maxwell, Scott A, Bonitz, Robert G, Biesiadecki, Jeffrey J, Hartman, Frank R, Cooper, Brian K, Baumgartner, Eric T, & Maimone, Mark W, "Mars exploration rover surface operations: Driving Spirit at Gusev Crater," . In: *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, Vol. 2. IEEE, 2005, Pages: 1815–1822. 49

[Lemaire *et al.* 2007] Lemaire, Thomas, Berger, Cyrille, Jung, Il-Kyun, & Lacroix, Simon, "Vision-Based SLAM: Stereo and Monocular Approaches," *International Journal of Computer Vision*, Vol. 74, No. 3, 2007, http://dx.doi.org/10.1007/s11263-007-0042-3 Pages: 343–364. 279

[Lemm 2003] Lemm, Jörg C., *Bayesian Field Theory*, The Johns Hopkins University Press, 2003. 51, 361

[Li *et al.* 2003] Li, Fei, Chen, Deming, He, Lei, & Cong, Jason, "Architecture Evaluation for Power-efficient FPGAs," . In: *Proceedings of the 2003 ACM/SIGDA Eleventh International Symposium on Field Programmable Gate Arrays*, FPGA '03, New York, NY, USA, 2003, Pages: 175–184.
http://doi.acm.org/10.1145/611817.611844 ACM. 25

[Li *et al.* 2009] Li, Pengfei, Yu, Jianping, Wan, Mingjie, Huang, Jianjun, & Huang, Jingxiong, "The augmented form of cubature Kalman filter and quadrature Kalman filter for additive noise," . In: *Information, Computing and Telecommunication, 2009. YC-ICT '09. IEEE Youth Conference on*, 2009, Pages: 295–298. 46

[Li *et al.* 2012a] Li, J., Post, M. A., & Lee, R., "Nanosatellite Attitude Air Bearing System using Variable Structure Control," . In: *IEEE 25th Annual Canadian Conference on Electrical and Computer Engineering*, Montreal, Canada, April 29- May 2, 2012. 6

[Li *et al.* 2012b] Li, J., Post, M. A., & Lee, R., "Real Time Fault Tolerant Nonlinear Attitude Control System for Nanosatellite Applications," . In: *AIAA Infotech@Aerospace 2012 Conference*, Garden Grove, California, June 19-21, 2012. 6

[Li *et al.* 2012c] Li, J., Post, M. A., Lee, R., & Kumar, K. D., "Nanosatellite Nonlinear Attitude Control Testing on an Air Bearing System," . In: *CASI ASTRO 2012*, Quebec City, Quebec, Canada, April 24-26, 2012. 6

[Li *et al.* 2013a] Li, J., Post, M., & Lee, R., "Real-Time Nonlinear Attitude Control System for Nanosatellite Applications," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 6, 2013, Pages: 1661–1671. 6, 99

[Li *et al.* 2013b] Li, J., Post, M., Wright, T., & Lee, R., "Design of Attitude Control Systems for CubeSat-Class Nanosatellite," *Journal of Control Science and Engineering*, Vol. 2013, 2013. 6

[Li *et al.* 2013c] Li, J., Post, M. A., & Lee, R., "Cubesat Attitude Control Systems with Magnetic Torquers and Flywheel," . In: *23rd AAS/AIAA Spaceflight Mechanics Meeting*, Kauai, Hawaii, February 10-14, 2013. 6

[Li *et al.* 2013d] Li, J., Post, M. A., & Lee, R., "A Novel Adaptive Unscented Kalman Filter Attitude Estimation and Control System for a 3U Nanosatellite," . In: *12th biannual European Control Conference*, Zurich, Switzerland, July 17-19, 2013. 6, 180

[Li 2012] Li, Junquan. "Intelligent Control of Satellite Formation Flying," . PhD thesis, Ryerson University, 2012. 99

[Lillis *et al.* 2008] Lillis, Robert J., Frey, Herbert V., Manga, Michael, Mitchell, David L., Lin, Robert P., Acuña, Mario H., & Bougher, Stephen W., "An improved crustal magnetic field map of Mars from electron reflectometry: Highland volcano magmatic history and the end of the Martian dynamo," *Icarus*, Vol. 194, No. 2, 2008, http://www.sciencedirect.com/science/article/pii/S0019103507005465 Pages: 575 – 596. 31

[Lin *et al.* 2010] Lin, Tsung-Chih, Chen, Ming-Che, Roopaei, M., & Sahraei, B.R., "Adaptive type-2 fuzzy sliding mode control for chaos synchronization of uncertain chaotic systems," . In: *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, July, 2010, Pages: 1–8. 56, 100

[Lindemann *et al.* 2006] Lindemann, R.A., Bickler, D.B., Harrington, B.D., Ortiz, G.M., & Voothees, C.J., "Mars exploration rover mobility development," *Robotics Automation Magazine, IEEE*, Vol. 13, No. 2, June, 2006, Pages: 19–26. 14

[Liu & Liu 2003] Liu, Yian-Kui, & Liu, Baoding, "Fuzzy Random Variables: A Scalar Expected Value Operator," *Fuzzy Optimization and Decision Making*, Vol. 2, No. 2, 2003, http://dx.doi.org/10.1023/A%3A1023447217758 Pages: 143–160. 56

[Liu 2001] Liu, Jun S., *Monte Carlo Strategies in Scientific Computing*, Springer, 2001. 56

[Longuet-Higgins & Prazdny 1980] Longuet-Higgins, H. C., & Prazdny, K., "The Interpretation of a Moving Retinal Image," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, Vol. 208, No. 1173, 1980, http://rspb.royalsocietypublishing.org/content/208/1173/385.abstract Pages: 385–397. 264, 266

[Longuet-Higgins 1987] Longuet-Higgins, H. C. "A computer algorithm for reconstructing a scene from two projections," . In: *Readings in computer vision: issues, problems, principles, and paradigms*, edited by M. A. Fischler & O. Firschein, Pages: 61–62. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, http://dl.acm.org/citation.cfm?id=33517.33523 1987. 273

[Lu & Milios 1994] Lu, Feng, & Milios, E. E., "Robot pose estimation in unknown environments by matching 2D range scans," . In: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, 1994, Pages: 935–938. 261

[Lundeen *et al.* 2011] Lundeen, Jeff S., Sutherland, Brandon, Patel, Aabid, Stewart, Corey, & Bamber, Charles, "Direct measurement of the quantum wavefunction," *Nature*, Vol. 474, No. 7350, June, 2011, http://dx.doi.org/10.1038/nature10120 Pages: 188–191. 54

[Luong & Faugeras 1995] Luong, Q.-T., & Faugeras, O.D., "The Fundamental matrix: theory, algorithms, and stability analysis," *International Journal of Computer Vision*, Vol. 17, 1995, Pages: 43–75. 273

[M.-S. Wei & You 2011] M.-S. Wei, F. Xing, B. Li, & You, Z., "Investigation of Digital Sun Sensor Technology with an N-Shaped Slit Mask," *Sensors*, Vol. 11, 2011, Pages: 9764–9777. 148, 301

[MacLean 1999] MacLean, W.J., "Removal of translation bias when using subspace methods," . In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, Vol. 2, 1999, Pages: 753–758 vol.2. 265, 268

[Manning 2012] Manning, Robert, "Engineering Truly Resilient Robotic Spacecraft: Can We Break Out of the Complexity Corner We Have Painted Ourselves Into?," . In: *AIAA Infotech@Aerospace 2012 Keynote Address*, 06, 2012. 34

[Mansard *et al.* 2004] Mansard, N., Aycard, O., & Koike, C., "Hierarchy of behaviours Application to the homing problem in indoor environment," . In: *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, 2004, Pages: 895–900. 241

[M.A.Post *et al.* 2013] M.A.Post, Li, J., & Lee, R., "A Low-Cost Photodiode Sun Sensor for CubeSat and Planetary Microrover," *International Journal of Aerospace Engineering*, Vol. 2013, 2013. 6, 145

[Maqsood & Akram 2010] Maqsood, I., & Akram, T., "Development of a Low Cost Sun Sensor using Quadphotodiode," *Proceedings of IEEE/ION PLANS 2010, Indian Wells, CA*, 2010, Pages: 639 – 644. 31

[Marosy *et al.* 2009] Marosy, G., Kovacs, Z., & Horvath, G., "Effective mars rover platform design with Hardware / Software co-design," . In: *Design and Diagnostics of Electronic Circuits Systems, 2009. DDECS '09*, april, 2009, Pages: 148 –151. 34

[Melzer & Swanson 1974] Melzer, Klaus-Jergen, & Swanson, GD. "Performance Evaluation of a Second-Generation Elastic Loop Mobility System.," . Rapport technique, DTIC Document, 1974. 13

[Menegaz *et al.* 2011] Menegaz, H. M., Ishihara, J. Y., & Borges, G. A., "A New Smallest Sigma Set for the Unscented Transform and its Applications on SLAM," *2011 50th IEEE Conference on Decision and Control and European Control Conference, Orlando, Fl, USA*, 2011, Pages: 3172–3177. 180, 187

[Minoni & Signorini 2006] Minoni, Umberto, & Signorini, Andrea, "Low-cost optical motion sensors: An experimental characterization," *Sensors and Actuators A: Physical*, Vol. 128, No. 2, 2006, http://www.sciencedirect.com/science/article/pii/S0924424706000926 Pages: 402 – 408. 257

[Montemerlo *et al.* 2002] Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B., "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," . In: *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI. 280

[Mooij 2010] Mooij, Joris M., "libDAI: A Free and Open Source C++ Library for Discrete Approximate Inference in Graphical Models," *Journal of Machine Learning Research*, Vol. 11, August, 2010, http://www.jmlr.org/papers/volume11/mooij10a/mooij10a.pdf Pages: 2169–2173. 59

[Moreno-Noguer *et al.* 2007] Moreno-Noguer, F., Lepetit, V., & Fua, P., "Accurate Non-Iterative O(n) Solution to the PnP Problem," . In: *IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, October, 2007. 277, 278

[Mu & Cai 2011] Mu, Jing, & Cai, Yuan-Li, "Iterated cubature Kalman filter and its application," . In: *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2011 IEEE International Conference on*, 2011, Pages: 33–37. 46

[Mudrova *et al.* 2011] Mudrova, Lenka, Faigl, Jan, Halgasık, Jaroslav, & Krajnık, Tomas, "Estimation of Mobile Robot Pose from Optical Mouses," . In: *EUROBOT 2010, CCIS*, edited

by D. Obdrzalek & A. Gottscheber, Vol. 156. Springer-Verlag Berlin Heidelberg, 2011, Pages: 93–107. 260

[Muja & Lowe 2009] Muja, Marius, & Lowe, David G., "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," . In: *International Conference on Computer Vision Theory and Application (VISSAPP'09)*. INSTICC Press, 2009, Pages: 331–340. 272

[Navarathinam *et al.* 2011] Navarathinam, Nimal, Lee, Regina, & Chesser, Hugh, "Characterization of Lithium-Polymer batteries for CubeSat applications," *Acta Astronautica*, Vol. 68, No. 11-12, 2011, http://www.sciencedirect.com/science/article/B6V1N-529C3FM-2/2/0d53fd5379ca193a813b311b94ac6d8d Pages: 1752 – 1760. 122

[Navarathinam 2010] Navarathinam, Nimal. "Nanosatellite Electrical Power System Development," . PhD thesis, York University, June, 2010. 311

[Negahdaripour & Lee 1992] Negahdaripour, Shahriar, & Lee, Shinhak, "Motion recovery from image sequences using only first order optical flow information," *International Journal of Computer Vision*, Vol. 9, No. 3, 1992, http://dx.doi.org/10.1007/BF00133700 Pages: 163–184. 261, 265

[Ng 2003] Ng, T.W., "The optical mouse as a two-dimensional displacement sensor," *Sensors and Actuators A: Physical*, Vol. 107, No. 1, 2003, http://www.sciencedirect.com/science/article/pii/S0924424703002565 Pages: 21 – 25. 257

[Orderud 2005] Orderud, Fredrik, "Comparison of Kalman filter estimation approaches for state space models with nonlinear measurements," . In: *Proceedings of Scandinavian conference on simulation and modeling*, 2005. 43

[Pakki *et al.* 2011] Pakki, K., Chandra, B., Gu, Da-Wei, & Postlethwaite, I., "Cubature information filter and its applications," . In: *American Control Conference (ACC), 2011*, 2011, Pages: 3609–3614. 46

[Palacin *et al.* 2006] Palacin, J., Valgañon, I., & Pernia, R., "The optical mouse for indoor mobile robot odometry measurement," *Sensors and Actuators A: Physical*, Vol. 126, No. 1, 2006, http://www.sciencedirect.com/science/article/pii/S0924424705005406 Pages: 141 – 147. 258

[Pantano & Timmerman 2012] Pantano, Nicolas, & Timmerman, Pr Martin, "Real-Time Operating System on Arduino," *OSSEC Mini Project Report*, http://ossec.es2.be/Portals/6/Users/082/82/82/Nicolas_Pantano_report.pdf 2012. 202

[Penna *et al.* 2010] Penna, Giuseppe Della, Intrigila, Benedetto, Magazzeni, Daniele, & Mercorio, Fabio, "Resource-Optimal Planning For An Autonomous Planetary Vehicle," *International Journal of Artificial Intelligence and Applications*, Vol. 1, No. 3, 2010, Pages: 15 – 29. 62

[Pesonen & Piche 2010] Pesonen, H., & Piche, R., "Cubature-based Kalman filters for positioning," . In: *Positioning Navigation and Communication (WPNC), 2010 7th Workshop on*, 2010, Pages: 45–49. 46

[Post & Lee 2009] Post, M., & Lee, R., "Lessons Learned from the York University Rover Team (YURT) and the University Rover Competition 2008," . In: *60th International Astronautical Congress*, Daejeon, Korea, October, 2009. 6

[Post & Lee 2011] Post, M.A., & Lee, R., "Lessons learned from the York University Rover Team (YURT) at the University Rover Challenge 2008-2009," *Acta Astronautica*, Vol. 68, No. 7-8, 2011,

http://www.sciencedirect.com/science/article/B6V1N-511CB62-3/2/fd04742a1459cefaa2dd0042766c4656 Pages: 1343 – 1352. 6, 34

[Post *et al.* 2011] Post, M. A., Lee, R., , & Quine, B., "Modular Design for Space Engineering Research Platforms," . In: *8th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Noordwijkerhout, Netherlands, July 29, 2011. 6

[Post *et al.* 2012a] Post, M. A., Li, J., & Lee, R., "Nanosatellite Air Bearing Tests of Fault-Tolerant Sliding-Mode Attitude Control with Unscented Kalman Filter," . In: *AIAA Guidance, Navigation, and Control Conference*, Minneapolis, Minnesota, August. 13-16, 2012. 6, 180

[Post *et al.* 2012b] Post, M. A., Quine, Brendan M., & Lee, R., "Bayesian Decision Making for Planetary Micro-Rovers," . In: *AIAA Infotech@Aerospace 2012*, Garden Grove, California, June 19 - 21, 2012. 6

[Post *et al.* 2012c] Post, M. A., Quine, Brendan M., & Lee, R., "Beaver Micro-Rover Development for the Northern Light Mars Lander," . In: *CASI ASTRO 2012*, Quebec City, Quebec, Canada, April 24-26, 2012. 4, 6

[Post *et al.* 2013] Post, M. A., Li, J., & Lee, R., "Nanosatellite Sun Sensor Attitude Determination using Low-Cost Hardware," . In: *23rd AAS/AIAA Spaceflight Mechanics Meeting*, Kauai, Hawaii, Feb 10-14, 2013. 146, 152

[Pourtakdoust *et al.* 2007] Pourtakdoust, S.H., Asl, & Ghanbarpour, H., "An adaptive unscented Kalman filter for quaternion-based orientation estimation in low-cost AHRS," *Aircraft Engineering and Aerospace Technology: An International Journal*, Vol. 79, No. 5, 2007, http://dx.doi.org/10.1108/00022660710780614 Pages: 485–493. 186

[Prazdny 1980] Prazdny, K., "Egomotion and relative depth map from optical flow," *Biological Cybernetics*, Vol. 36, No. 2, 1980, http://dx.doi.org/10.1007/BF00361077 Pages: 87–102. 261, 264

[Prazdny 1981] Prazdny, K., "Determining The Instantaneous Direction Of Motion From Optical Flow Generated By A Curvilinearly Moving Observer," *Proc. SPIE*, Vol. 0281, 1981, http://dx.doi.org/10.1117/12.965748 Pages: 199–206. 264

[Quest 2013] Quest, NASA. "Aerospace: Mars Facts," . Online, http://quest.nasa.gov/aero/planetary/mars.html, accessed 2014/05/01 2013. 8

[Quigley *et al.* 2009] Quigley, Morgan, Conley, Ken, Gerkey, Brian, Faust, Josh, Foote, Tully, Leibs, Jeremy, Wheeler, Rob, & Ng, Andrew Y, "ROS: an open-source Robot Operating System," . In: *ICRA workshop on open source software*, Vol. 3, 2009. 39

[Quine *et al.* 1995] Quine, B., Uhlmann, J., & Durrant-Whyte, H., "Implicit Jacobian for linearised state estimation in nonlinear systems," . In: *American Control Conference, Proceedings of the 1995*, Vol. 3, 1995, Pages: 1645–1646. 43

[Quine *et al.* 2008] Quine, B., Lee, R., & Roberts, C. et al., "Northern Light - A Canadian Mars Lander Development Plan," . In: *Conference Proceedings of CASI ASTRO 2008, Montreal, Canada*, 2008. 4

[Quine 2006] Quine, Brendan M., "A derivative-free implementation of the extended Kalman filter," *Automatica*, Vol. 42, No. 11, 2006, http://www.sciencedirect.com/science/article/pii/S0005109806002469 Pages: 1927 – 1934. 43

[Quine 2013] Quine, Brendan. "Northern Light, a Canadian Mission to Mars: Beaver Rover," . Online, http://marsrocks.ca/northernlight/rover.html, accessed 2013/12/30 2013. 8

[Raudies & Neumann 2009] Raudies, F., & Neumann, H., "An Efficient Linear Method for the Estimation of Ego-motion from Optical Flow," . In: *DAGM 2009 LNCS 5748*, edited by J. Denzler, G. Notni, & H. Susse, 2009, Pages: 11–20. 266, 267

[Reichardt *et al.* 2013] Reichardt, Max, Föhst, Tobias, & Berns, Karsten, "On software quality-motivated design of a real-time framework for complex robot control systems," . In: *Proceedings of the 7th International Workshop on Software Quality and Maintainability (SQM), in conjunction with the 17th European Conference on Software Maintenance and Reengineering (CSMR)*, 2013. 40

[Reina & Foglia 2013] Reina, Giulio, & Foglia, Mario, "On the mobility of all-terrain rovers," *Industrial Robot: An International Journal*, Vol. 40, No. 2, 2013, Pages: 121–131. 14

[Ribeiro 2004] Ribeiro, Maria Isabel, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, 2004, Page: 43. 42

[Rigo 2004] Rigo, Armin, "Representation-based Just-in-time Specialization and the Psyco Prototype for Python," . In: *Proceedings of the 2004 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation*, PEPM '04, New York, NY, USA, 2004, Pages: 15–26.
http://doi.acm.org/10.1145/1014007.1014010 ACM. 158

[Robinson 1965] Robinson, J. A., "A Machine-Oriented Logic Based on the Resolution Principle," *J. ACM*, Vol. 12, No. 1, January, 1965,
http://doi.acm.org/10.1145/321250.321253 Pages: 23–41. 228

[Robinson 1979] Robinson, G. K., "Conditional Properties of Statistical Procedures," *The Annals of Statistics*, Vol. 7, No. 4, 1979,
http://www.jstor.org/stable/2958922 Pages: pp. 742–755. 55

[Robson *et al.* 2012] Robson, Nina P, Morgan, J, & Baumgartner, H, "Mechanical Design Of A Standardized Ground Mobile Platform," *International Journal of Modern Engineering*, Vol. 12, No. 2, March, 2012, Page: 59. 14

[Rosten & Drummond 2005] Rosten, Edward, & Drummond, Tom, "Fusing points and lines for high performance tracking," . In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, Vol. 2, 2005, Pages: 1508–1515 Vol. 2. 270, 271

[Rublee *et al.* 2011] Rublee, Ethan, Rabaud, Vincent, Konolige, Kurt, & Bradski, Gary R., "ORB: An efficient alternative to SIFT or SURF," . In: *ICCV 2011*, 2011, Pages: 2564–2571. 270, 271

[S. Chouraqui 2005] S. Chouraqui, M. Benyettou, M.A. Si, "Sensor Vectors Modeling for Small Satellite Attitude Determination," *Journal of Applied Sciences*, Vol. 5, No. 10, 2005, Pages: 1739–1743. 145

[Sarkka & Solin 2012] Sarkka, Simo, & Solin, Arno, "On continuous-discrete cubature Kalman filtering," . In: *Proceedings of SYSID*, Vol. 2012, 2012. 46

[Scharer *et al.* 2004] Scharer, S., Baltes, J., & Anderson, J., "Practical ego-motion estimation for mobile robots," . In: *Robotics, Automation and Mechatronics, 2004 IEEE Conference on*, Vol. 2, 2004, Pages: 921–926 vol.2. 263

[Schilling & Jungius 1996] Schilling, K., & Jungius, C., "Mobile robots for planetary exploration," *Control Engineering Practice*, Vol. 4, No. 4, 1996, http://www.sciencedirect.com/science/article/pii/0967066196000342 Pages: 513 – 524. 14

[Sekimori & Miyazaki 2007] Sekimori, Daisuke, & Miyazaki, Fumio. "Precise Dead-Reckoning for Mobile Robots using Multiple Optical Mouse Sensors," . In: *Informatics in Control, Automation and Robotics II*, edited by J. Filipe, J.-L. Ferrier, J. A. Cetto, & M. Carvalho, Pages: 145–151. Springer Netherlands, 2007. http://dx.doi.org/10.1007/978-1-4020-5626-0_18 10.1007/978-1-4020-5626-0_18. 257

[Serdar Guzel & Bicker 2012] Serdar Guzel, Mehmet, & Bicker, Robert, "A Behaviour-Based Architecture for Mapless Navigation Using Vision.," *International Journal of Advanced Robotic Systems*, Vol. 9, 2012, http://ezproxy.library.yorku.ca/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=91530762&site=ehost-live Pages: 1 – 13. 241

[Shil 2012a] Shil, Roy. "Simple triangulation with OpenCV from Harley & Zisserman," . Online, http://www.morethantechnical.com/2012/01/04/simple-triangulation-with-opencv-from-harley-zisserman-w-code/, accessed 2013/09/30 January, 2012. 276

[Shil 2012b] Shil, Roy. "Structure from Motion and 3D reconstruction on the easy in OpenCV 2.3+," . Online, http://www.morethantechnical.com/2012/02/07/structure-from-motion-and-3d-reconstruction-on-the-easy-in-opencv\-2-3-w-code/, accessed 2013/09/30 February, 2012. 48, 275

[Shima 1999] Shima, J. "DSP Trick: Fixed-Point Atan2 With Self Normalization," . Post in comp.dsp newsgroup, http://www.dspguru.com/comp.dsp/tricks/alg/fxdatan2.htm April, 1999. 164

[Smith *et al.* 1990] Smith, R., Self, M., & Cheeseman, P. "Estimating uncertain spatial relationships in robotics," . In: *Autonomous Robot Vehicles*, edited by I. J. Cox & G. T. Wilfong, Pages: 167–193. Springer-Verlag New York, Inc., New York, NY, USA, http://dl.acm.org/citation.cfm?id=93002.93291 1990. 283

[Smith 2003] Smith, Jim. "What about Ada? The state of the technology in 2003," . Rapport technique, DTIC Document, 2003. 156

[Soken & Hajiyev 2009] Soken, H. E., & Hajiyev, C., "Adaptive Unscented Kalman Filter with Multiple Fading Factors for Picosatellite Attitude Estimation," *4th International Conference on Recent Advances in Space Technologies*, 2009, Pages: 541–546. 186

[Springmann & Cutler 2013] Springmann, John C., & Cutler, James W., "Optimization of Directional Sensor Orientation with Application to Photodiodes for Spacecraft Attitude Determination," *23th AAS/AIAA Space Flight Mechanics Meeting - AAS/AIAA, Hawaii, Feb 2013*, 2013. 150

[Squyres 2008] Squyres, Steven W, "Mars Exploration Rovers," . In: *Bulletin of the American Astronomical Society*, Vol. 40, 2008, Page: 500. 12

[Stalker & Boeren 1995] Stalker, Night, & Boeren, David. "How to use Fixed Point (16.16) Math," , http://textfiles.com/programming/fix1faq.txt, accessed 2013 March, 1995. 163

[Street 2004] Street, M., "A Fixed Point Math Primer," . In: *OpenGL ES Game Development*. Course Technology PTR, 2004. 163

[Torre *et al.* 2010] Torre, Alberto Della, Finzi, Amalia Ercoli, Genta, Giancarlo, Curt, Fabio, Schirone, Luigi, Capuano, Giuseppe, Sacchetti, Andrea, Vukman, Igor, Mailland, Filippo, Monchieri, Emanuele, Guidi, Andrea, Trucco, Roberto, Musso, Ivano, & Lombardi, Chiara, "AMALIA Mission Lunar Rover–The conceptual design of the Team ITALIA Rover, candidate for the Google Lunar X Prize Challenge," *Acta Astronautica*, Vol. 67, No. 7-8, 2010, http://www.sciencedirect.com/science/article/B6V1N-50F45V1-1/2/6e1b98a543a3f132a337cbc96c21ceaa Pages: 961 – 978. 120

[Toyokazu *et al.* 2009] Toyokazu, Tambo., Miki, Shibata., Yuta, Mizuno., & Toyohiro, Yamauchi., "Search Method of Sun Using Fixed Five Photodiode Sensor," *IEEJ Transactions on Sensors and Micromachines*, Vol. 129, No. 2, 2009, Pages: 53–59. 150

[Trebi-Ollennu *et al.* 2001] Trebi-Ollennu, Ashitey, Huntsberger, Terry, Cheng, Yang, Baumgartner, E. T., Kennedy, Brett, & Schenker, Paul, "Design and Analysis of a Sun Sensor for Planetary Rover Absolute Heading Detection," *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 6, 2001, Pages: 139–145. 31, 151

[Triggs *et al.* 2000] Triggs, Bill, McLauchlan, Philip, Hartley, Richard, & Fitzgibbon, Andrew. "Bundle Adjustment — A Modern Synthesis," . In: *Vision Algorithms: Theory and Practice*, edited by B. Triggs, A. Zisserman, & R. Szeliski, Vol. 1883 of *Lecture Notes in Computer Science*, Pages: 298–372. Springer Berlin Heidelberg, http://dx.doi.org/10.1007/3-540-44480-7_21 2000. 48

[Tsao *et al.* 1997] Tsao, An-Ting, Hung, Yi-Ping, Fuh, Chiou-Shann, & Chen, Yong-Sheng, "Ego-motion estimation using optical flow fields observed from multiple cameras," . In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997*, 1997, Pages: 457–462. 264

[Tsunakawa *et al.* 2010] Tsunakawa, Hideo, Shibuya, Hidetoshi, Takahashi, Futoshi, Shimizu, Hisayoshi, Matsushima, Masaki, Matsuoka, Ayako, Nakazawa, Satoru, Otake, Hisashi, & Iijima, Yuichi, "Lunar Magnetic Field Observation and Initial Global Mapping of Lunar Magnetic Anomalies by MAP-LMAG Onboard SELENE (Kaguya)," *Space Science Reviews*, Vol. 154, No. 1-4, 2010, http://dx.doi.org/10.1007/s11214-010-9652-0 Pages: 219–251. 31

[Tuan *et al.* 2006] Tuan, Tim, Kao, Sean, Rahman, Arif, Das, Satyaki, & Trimberger, Steve, "A 90Nm Low-power FPGA for Battery-powered Applications," . In: *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*, FPGA '06, New York, NY, USA, 2006, Pages: 3–11. http://doi.acm.org/10.1145/1117201.1117203 ACM. 26

[Turkowski 1994] Turkowski, Ken. "Fixed Point Square Root," . Rapport technique 96, Apple Technical Report, 1994. Also appears in Graphics Gems V. 164

[Van der Merwe & Wan 2001] Van der Merwe, R., & Wan, E.A., "The square-root unscented Kalman filter for state and parameter-estimation," . In: *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, Vol. 6, 2001, Pages: 3461–3464 vol.6. 45

[Van Verth & Bishop 2008] Van Verth, James M, & Bishop, Lars M, *Essential Mathematics for Games & Interactive Applications: A Programmer's Guide*, Taylor & Francis US, 2008. 162

[VanRossum & Drake 2010] VanRossum, Guido, & Drake, Fred L, *The Python Language Reference*, Python Software Foundation, 2010. 158

[Vaughan & Gerkey 2007] Vaughan, RichardT., & Gerkey, BrianP. "Reusable Robot Software and the Player/Stage Project," . In: *Software Engineering for Experimental Robotics*, edited by D. Brugali, Vol. 30 of *Springer Tracts in Advanced Robotics*, Pages: 267–289. Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-540-68951-5_16 2007. 39

[Vinh *et al.* 2011] Vinh, NguyenXuan, Chetty, Madhu, Coppel, Ross, & Wangikar, PramodP. "Dynamic Bayesian Network Modeling of Cyanobacterial Biological Processes via Gene Clustering," . In: *Neural Information Processing*, edited by B.-L. Lu, L. Zhang, & J. Kwok, Vol. 7062 of *Lecture Notes in Computer Science*, Pages: 97–106. Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-642-24955-6_12 2011. 55

[Volpe *et al.* 2000] Volpe, Richard, Nesnas, Issa A. D., Estlin, Tara, Mutz, Darren, Petras, Richard, & Das, Hari. "CLARAty: Coupled Layer Architecture for Robotic Autonomy," . Rapport technique, NASA - JET PROPULSION LABORATORY, 2000. 40

[Volpe 1999] Volpe, R., "Mars Rover Navigation Results using Sun Sensor Heading Determination," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, 1999, Pages: 460–467. 31

[Wang & Hussein 2011a] Wang, Y., & Hussein, I. I., "Bayesian-Based Decision Making for Object Search and Classification," *IEEE Transactions on Control Systems Technology*, Vol. 19, No. 6, 2011, Pages: 1639–1647. 250

[Wang & Hussein 2011b] Wang, Y., & Hussein, I. I., "Intentional Control for Planetary Rover SRR," . In: *American Control Conference (ACC), 2011*, June 29-July 1, 2011, Pages: 1280 – 1285. 252

[Wang *et al.* 2007] Wang, Zhongshi, Wu, Wei, Xu, Xinhe, & Xue, Dingyu, "Recognition and location of the internal corners of planar checkerboard calibration pattern image," *Applied Mathematics and Computation*, Vol. 185, No. 2, 2007, Pages: 894 – 906. http://www.sciencedirect.com/science/article/pii/S0096300306008071 Special Issue on Intelligent Computing Theory and Methodology. 274

[Wang *et al.* 2009] Wang, Xuebing, Ban, K., & Ishii, K., "Estimation of mobile robot ego-motion and obstacle depth detection by using optical flow," . In: *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, 2009, Pages: 1770–1775. 261, 265

[Wang 1997] Wang, Li-Xin, *A Course in Fuzzy Systems and Control*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997. 100

[Welch & Bishop 1995] Welch, Greg, & Bishop, Gary. "An Introduction to the Kalman Filter," . Rapport technique, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995. 41

[Wilcox & Jones 2000] Wilcox, B.H., & Jones, R.M., "The MUSES-CN nanorover mission and related technology," . In: *Aerospace Conference Proceedings, 2000 IEEE*, Vol. 7, 2000, Pages: 287–295 vol.7. 13

[Wilcox 1996] Wilcox, Brian. *Nanorovers for Planetary Exploration*. Jet Propulsion Laboratory, California Institute Of Technology,

http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/32248/1/95-1594.pdf 1996. 12

[Wu & Butz 2005] Wu, Dan, & Butz, Cory. "On the Complexity of Probabilistic Inference in Singly Connected Bayesian Networks," . In: *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, edited by D. Slezak, G. Wang, M. Szczuka, I. Düntsch, & Y. Yao, Vol. 3641 of *Lecture Notes in Computer Science*, Pages: 581–590. Springer Berlin Heidelberg,
http://dx.doi.org/10.1007/11548669_60 2005. 225

[Wu *et al.* 2008] Wu, Changchang, Clipp, Brian, Li, Xiaowei, Frahm, Jan-Michael, & Pollefeys, Marc, "3D model matching with viewpoint-invariant patches (VIP)," . In: *Computer Society Conference on Computer Vision and Pattern Recognition*, 2008, Page: 1–8. 280

[Wurm *et al.* 2010] Wurm, Kai M., Hornung, Armin, Bennewitz, Maren, Stachniss, Cyrill, & Burgard, Wolfram, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," . In: *In Proceedings of the ICRA 2010 workshop*, 2010. 280

[Xiong *et al.* 2006] Xiong, K, Zhang, H. Y., & Chan, C. W., "Performance evaluation of UKF-based nonlinear filtering," *Automatica*, Vol. 42, 2006, Pages: 261–270. 181

[Yehezkel & Lerner 2006] Yehezkel, Raanan, & Lerner, Boaz. "Bayesian Network Structure Learning by Recursive Autonomy Identification," . In: *Structural, Syntactic, and Statistical Pattern Recognition*, edited by D.-Y. Yeung, J. Kwok, A. Fred, F. Roli, & D. Ridder, Vol. 4109 of *Lecture Notes in Computer Science*, Pages: 154–162. Springer Berlin Heidelberg,
http://dx.doi.org/10.1007/11815921_16 2006. 243

[Yoshida & Hamano 2002] Yoshida, Kazuya, & Hamano, Hiroshi, "Motion dynamics and control of a planetary rover with slip-based traction model," . In: *Proc. SPIE*, Vol. 4715, 2002, Pages: 275–286. 14

[Zadeh 1965] Zadeh, Lotfi A, "Fuzzy sets," *Information and control*, Vol. 8, No. 3, 1965, Pages: 338–353. 56

[Zadeh 1975] Zadeh, L.A., "The concept of a linguistic variable and its application to approximate reasoning—I," *Information Sciences*, Vol. 8, No. 3, 1975, http://www.sciencedirect.com/science/article/pii/0020025575900365 Pages: 199 – 249. 100

[Zadeh 1999] Zadeh, L. A., "Fuzzy Sets As a Basis for a Theory of Possibility," *Fuzzy Sets Syst.*, Vol. 100, April, 1999, http://dl.acm.org/citation.cfm?id=310817.310820 Pages: 9–34. 56

[Zaloga 2003] Zaloga, Steven, *M4 (76mm) Sherman Medium Tank 1943-65*, Vol. 73, Osprey Publishing, 2003. 68

[Zerigui *et al.* 2007] Zerigui, A., Wu, Xiang, quan Deng, Zong, & Yu, Kejie, "Common hardware design For wheels mobile rover," . In: *Robotics and Biomimetics, 2007. ROBIO 2007*, December, 2007, Pages: 712 –716. 132

[Zhuang *et al.* 1988] Zhuang, Xinhua, Huang, Thomas S., Ahuja, Narendra, & Haralick, Robert M., "A simplified linear optic flow-motion algorithm," *Computer Vision, Graphics, and Image Processing*, Vol. 42, No. 3, 1988, http://www.sciencedirect.com/science/article/pii/S0734189X88800434 Pages: 334 – 344. 263

[Zou & Feng 2009] Zou, Cunlu, & Feng, Jianfeng, "Granger causality vs. dynamic Bayesian network inference: a comparative study," *BMC Bioinformatics*, Vol. 10, No. 1, 2009, http://www.biomedcentral.com/1471-2105/10/122 Page: 122. 55

List of Publications

Micro-rover articles published or accepted in refereed journals:

1. Mark A. Post, Junquan Li, Regina Lee, "A Low-Cost Photodiode Sun Sensor for CubeSat and Planetary Microrover". International Journal of Aerospace Engineering, vol. 2013, Article ID 549080, 9 pages, 2013. doi:10.1155/2013/549080.

2. M.A. Post, R. Lee. "Lessons learned from the York University Rover Team (YURT) at the University Rover Challenge 2008–2009". Acta Astronautica. Accepted Aug. 28 2010, doi:10.1016/j.actaastro.2010.08.037.

Nano-satellite articles published or accepted in refereed journals:

1. M. Post, J. Li, R. Lee. "Design and Construction of a Magnetic Field Simulator for CubeSat Attitude Control Testing". Journal of Instrumentation, Automation, and Systems. Vol. 1. Accepted for publication May 9, 2014.

2. J. Li, M. A. Post, T. Wright, R. Lee. "Design of Attitude Control Systems for CubeSat-class Nanosatellite". Journal of Control Science and Engineering, Special Issue on Embedded Control Systems. Journal of Control Science and Engineering, Volume 2013 (2013), Article ID 657182.

3. Junquan Li, Mark Post, and Regina Lee. "Real-Time Nonlinear Attitude Control System for Nanosatellite Applications". Journal of Guidance, Control, and Dynamics, (2013). Accessed October 29, 2013, doi: http://arc.aiaa.org/doi/abs/10.2514/1.59023.

Unrelated articles published or accepted in refereed journals:

1. Z.H. Zhu, M.A. Post, S.A. Meguid. "The Potential of Ultrasonic Non-Destructive Measurement of Residual Stresses by Modal Frequency Spacing using Leaky Lamb Waves". Journal of Experimental Mechanics. Accepted Dec. 2011, Published Online Jan. 06 2012, doi:10.1007/s11340-011-9585-x.

2. Z.H.Zhu, M.A.Post, and P.C.Xu. “Stress evaluation using Ultrasonic Interference Spectrum of Leaky Lamb Waves”. Journal of Experimental Mechanics. Accepted July 12 2010, Published Online Aug. 06 2010, doi:10.1007/s11340-010-9391-x.

3. M. Post, Z. H. Zhu, G. Clarkson. “Characterization of fibre orientation in FRP using pitch-catch ultrasonic technique”. International Journal for Multiscale Materials Modeling. Vol.1, Number 1, January-June 2010 pp53-61.

Articles in preparation for refereed journals:

1. M. Post, B. Quine. “The Beaver Micro-Rover, a Canadian 6kg Autonomous Platform for Exploration”. Canadian Aeronautics and Space Journal. In preparation, expected publication in 2015.

2. Mark A. Post, Junquan Li, Regina Lee, Brendan M. Quine. ”Fixed-Point Implementation and Comparison of Sigma-Point Kalman Filters for Embedded Applications”. Automatica. In preparation.

3. J. Li, M. Post, R. Lee. “FPGA Implementation of Adaptive Type-2 Fuzzy Controller for Satellite Attitude Control Systems”. Mechatronics. In preparation.

4. M.A. Post, B. Quine, R. Lee, J. Li. ”Micro-Rover Map Building with Probabilistic Range Sensor Model”. IEEE Transactions on Control Systems Technology, currently under requested revision.

Micro-rover articles published in refereed conference proceedings:

1. Mark A. Post, Brendan M. Quine, Regina Lee. “Bayesian Decision Making for Planetary Micro-Rovers”. AIAA Infotech@Aerospace 2012. 19 - 21 Jun 2012, Hyatt Regency Orange County, Garden Grove, California.

2. Jordan Bailey, Shailja Sahani, Mark A. Post, Regina Lee, Michael Daly. ”York University Rover Team: Space Education Beyond The Classroom, Developing Lasting Institutions

For Hands-On Education". CASI ASTRO 2012. April 24-26, 2012, Quebec City, Quebec, Canada.

3. Mark A. Post, Brendan M. Quine, Regina Lee. "Beaver Micro-Rover Development for the Northern Light Mars Lander". CASI ASTRO 2012. April 24-26, 2012, Quebec City, Quebec, Canada.

4. M. A. Post, R. Lee, and B. Quine. "Modular Design for Space Engineering Research Platforms". 8th International Conference on Informatics in Control, Automation and Robotics (ICINCO). Noordwijkerhout, Netherlands, July 29, 2011.

5. Mark Post, Dr. Regina Lee. "Lessons Learned from the York University Rover Team (YURT) and the University Rover Competition 2008". 60th International Astronautical Congress (IAC2009). Daejeon, October, 2009.

Nano-satellite articles published in refereed conference proceedings:

1. J. Li, M. A. Post, R. Lee. "A Novel Adaptive Unscented Kalman Filter Attitude Estimation and Control System for a 3U Nanosatellite". 12th Biannual European Control Conference, July 17-19 2013, Zurich, Switzerland.

2. J. Li, M. A. Post, R. Lee. "Cubesat Attitude Control Systems with Magnetic Torquers and Flywheel". 23rd AAS/AIAA Spaceflight Mechanics Meeting. February 10- 14, 2013, Kauai, Hawaii.

3. M. A. Post, J. Li, R. Lee. "Nanosatellite Sun Sensor Attitude Determination using Low-Cost Hardware". 23rd AAS/AIAA Spaceflight Mechanics Meeting. February 10-14, 2013, Kauai, Hawaii.

4. Mark A. Post, Junquan Li, Regina Lee. "Nanosatellite Air Bearing Tests of Fault-Tolerant Sliding-Mode Attitude Control with Unscented Kalman Filter". AIAA Guidance, Navigation, and Control Conference. Minneapolis, Minnesota, August 13-16, 2012.

5. Junquan Li, Mark A. Post, Regina Lee. ”Nanosatellite Attitude Air Bearing System using Variable Structure Control”. IEEE 25th Annual Canadian Conference on Electrical and Computer Engineering. Montreal, Canada, April 29-May 2, 2012.

6. Junquan Li, Mark A. Post, Regina Lee. ”Real Time Fault Tolerant Nonlinear Attitude Control System for Nanosatellite Applications”. AIAA Infotech@Aerospace 2012 Conference. Garden Grove, California, June 19-21, 2012.

7. Regina Lee, Hugh Chesser, Mathew Cannata, Mark A. Post, K. D. Kumar. ”Modular Attitude Control System Design For CubeSat Application”. CASI ASTRO 2012, April 24-26, 2012. Quebec City, Quebec, Canada.

8. Junquan Li, Mark A. Post, Regina Lee, K. D. Kumar. “Nanosatellite Nonlinear Attitude Control Testing on an Air Bearing System”. CASI ASTRO 2012. April 24-26, 2012, Quebec City, Quebec, Canada.

Unrelated articles published in refereed conference proceedings:

1. M. Post, Z. H. Zhu. “Non-Destructive Measurement of Stress using Ultrasonic Leaky Lamb Waves”. 22nd Canadian Congress of Applied Mechanics (CANCAM). Halifax, NS, June, 2009.

2. M. Post, G. Zhu, G. Clarkson. “Ultrasonic Amplitude Fitting of of Fibre Orientation in Fibre-Reinforced Polymers”. 22nd Canadian Congress of Applied Mechanics (CANCAM). Halifax, NS, June, 2009.

3. J. O. Parra, P.-C. Xu and M. A. Post. “Experimental ultrasonic microseismograms using scaled realistic earth and borehole models”. Expanded Abstracts, BG P1.1, SEG 76th Annual Meeting. New Orleans, 2006.

4. Steve Mann, Ryan Janzen, Mark Post. “Hydraulophone Design Considerations: Absement, Displacement, and Velocity-Sensitive Music Keyboard in which each key is a water jet”. MM-2006: ACM Multimedia. Santa Barbara, California, Oct. 23-27, 2006.

Articles in preparation for refereed conferences:

1. M. A. Post, J. Li, B. Quine. “Planetary Micro-Rover Operations on Mars using a Bayesian Framework for Inference and Control”. Oral Presentation. 65th International Astronautical Congress, Toronto, Canada, September 29-October 3, 2014.