RESIDENT SPACE OBJECT TRACKING FOR SPACE SITUATIONAL AWARENESS

PERUSHAN KUNALAKANTHA

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN EARTH AND SPACE SCIENCE

YORK UNIVERSITY

TORONTO, ONTARIO

AUGUST 2024

Abstract

This research presents several contributions aimed at improving the current state of Space Situational Awareness (SSA) using optical imagery. The first component involved the development of an image capture algorithm which was successfully used to acquire over 90000 optical images from the stratosphere, with hundreds of visually verified Resident Space Objects (RSOs). The second component involved the development of a novel RSO tracking algorithm which was able to detect 87% of the RSOs in an 878-image dataset at least once. The third component proposed an automated annotation framework and corresponding four-tool annotation suite to develop a 500-image dataset to train and test another RSO tracking algorithm. The final component involved the demonstration of a dual-purpose payload, performing RSO detection alongside an existing Attitude Determination (AD) algorithm which detected 11 unique RSOs in real-time during another stratospheric balloon mission.

Acknowledgements

I really struggled to write my acknowledgements page. I found it hard to fit my gratitude for all the amazing people that have supported me through this journey into just a single page. This thesis and my personal development would not have been possible without a long list of people, far greater than I can credit on this page. I hope I can meet you all again in person and truly tell each of you how grateful I am. But alas, I have to keep this short.

I would like to start by thanking my supervisor Dr. Regina Lee. I am overcome with emotion writing that simple sentence. I cannot begin to describe how much you have changed my life. Thank you so much for believing in me, being patient with me, and providing me with opportunities beyond what I could have ever imagined.

Thank you to Dr. Franz Newland for being an excellent role model. I am forever inspired by your wisdom, kindness, positivity, and selflessness.

Thank you to all my lab mates, great friends, and colleagues: Randa, Gabe, Vithurshan, Mike F., Mike S., Vidushi, Andrea, June, Sid, Ryan, Nick, Peter, Akash, Sriyan, Constantine, Antoine, Romain, Diane, David, Sergiy, Boris, Yan, Tom, Warren, Van, Lawrence, Hyunbin, Hashir, Ian, Marissa, Connor, Tim, Sabrina, Jack, Andrew, Punit, Aiden, Emilee, Mishal, Sogand, Shamil, Alex, Paul, and Matthew. I apologize if I missed anyone and know that your effect can never be forgotten. Thank you for the memories, and I wish the best for all of you.

Thank you to my parents, sister, and family for their support throughout the decades.

And to you. You know who you are. My greatest ambition.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Description
AD	Attitude Determination
BIRALES	Bistatic Radar for LEO Survey
BIRALET	Bistatic Radar for LEO Tracking
CAD	Computer-Aided Design
CASSIOPE	Cascade SmallSat and Ionospheric Polar Explorer
CCA	Connected Component Analysis
CNN	Convolutional Neural Network
COTS	Commercial-Off-the-Shelf
CSA	Canadian Space Agency
Dec	Declination
DNN	Deep Neural Network
FAI	Fast Auroral Imager
FEM	Finite Element Modeling
FN	False Negative
FOV	Field-of-View
FP	False Positive
FPGA	Field-Programmable Gate Array
FS	Full-Sequence

GEO	Geosynchronous Earth Orbit
GEODSS	Ground-Based Electro-Optical Deep Space Surveillance
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GT	Ground Truth
GUI	Graphical User Interface
ICP	Iterative Closest Point
IDF1	Identification F1
IDP	Identification Precision
IDR	Identification Recall
IDSW	ID Switch
IOD	Initial Orbit Determination
JPDA	Joint Probabilistic Data Association
LAP	Linear Assignment Problem
LEO	Low Earth Orbit
LIS	Lost-in-Space
LSTM	Long Short-Term Memory
MAL	Model Assisted Labelling
ML	Machine Learning
MODEST	Michigan Orbital Debris Survey Telescope
MOT	Multiple Object Tracking

MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
NASA	Nautical Aeronautics and Space Administration
NN	Neural Network
OBC	Onboard Computer
OS	Operating System
PDU	Power Distribution Unit
PF	Per-Frame
PSF	Point Spread Function
RA	Right Ascension
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
ROI	Region of Interest
RSO	Resident Space Object
RSONAR	Resident Space Object Near-space Astrometric Research
SBSS	Space Based Space Surveillance
sCMOS	scientific Complementary Metal–Oxide-Semiconductor
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip

SPEED	Spacecraft Pose Estimation Dataset
SPI	Serial Peripheral Interface
SSA	Space Situational Awareness
SSN	Space Surveillance Network
SSO	Sun-Synchronous Orbit
STARDUST	Star Tracker Attitude and RSO Detection for Unified Space Technologies
TAO	Tracking Any Object
TIRA	Tracking and Imaging Radar
TLE	Two-Line Element
TN	True Negative
TP	True Positive
VOT	Visual Object Tracking
WCS	World Coordinate System
YOLO	You Only Look Once

List of Symbols

Symbol	Description
$d_{Similarity}$	The similarity of two distances.
θ_{max}	The maximum permissible angle.

1. Introduction

1.1. Problem Statement

Many of the conveniences of the modern age that we take for granted are owed to space, and the numerous ways in which we take advantage of it. The satellites we launch into Earth orbit have been key in providing us with these services, such as high-speed communications, global internet, weather predictions, natural disaster monitoring, high-speed banking, navigation, and defence [1]. Given these essential services, it is without a doubt that the space environment around Earth is of great importance.

In recent years, space activity has exploded, as indicated by the rapid increase in total space objects since 2016, as observed in Figure 1.1. Companies invested in space are building and launching networks of satellites, called mega-constellations, into Low Earth Orbit (LEO). Starlink™, a satellite network developed by SpaceX, has over 5400 satellites in orbit alone, with a total planned size of 42000 satellites [2]. Other mega-constellations such as OneWeb, Iridium Next, and Globalstar are currently in construction or are beginning their launches [3]. It is estimated that by 2030, the population of active satellites will exceed 60000, a number many times greater than the 8300 active satellites in orbit right now [4].

These numbers become even larger when considering the other objects in space. Spent rocket stages, fragments, and inactive satellites also orbit Earth, though they are no longer useful for their initial purposes. Though these numbers increase as more satellites are launched into space, several events have occurred in history that have caused sharp increases in these numbers.

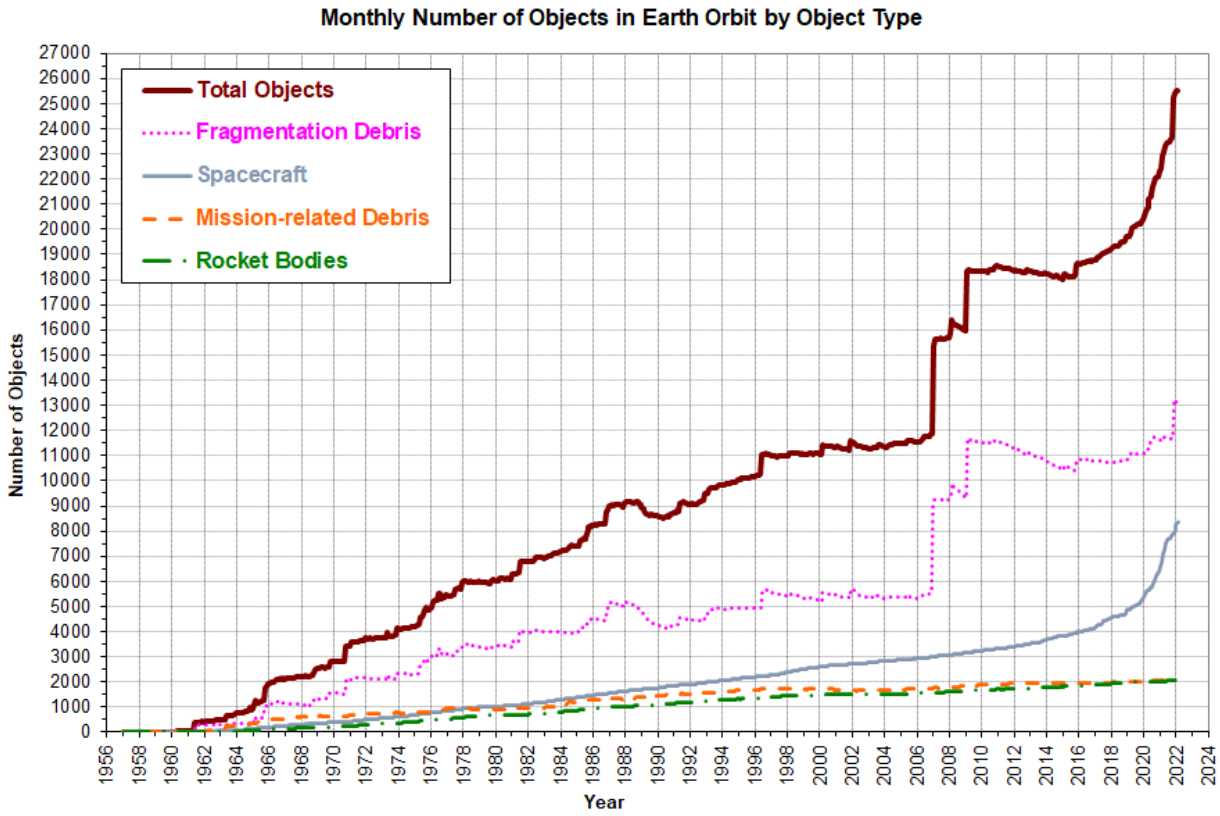

Figure 1.1. Number of objects greater than 10 cm in LEO, by year [5].

One such event was the unintentional collision in 2009 between an active communications satellite, Iridium 33, and an inactive military satellite, Kosmos 2251 [6]. This collision created over 2000 pieces of debris at least 10 cm in size, and many thousands more smaller pieces, which will remain in orbit for at least 100 years after the collision [6]. Figure 1.2 illustrates the debris predicted to have been generated by the collision, just three hours after the initial impact [7]. The consequences of this event can be observed in Figure 1.1, where there is a sharp increase in fragmentation debris in 2009 [5].

Another event generating significant debris was the anti-satellite missile test in 2007, where China launched a ballistic missile to destroy its own weather satellite, Fengyun-1C [8]. This

demonstration created more than 3000 pieces of debris, while scientists estimate that more than 32000 smaller pieces of debris remain untracked [8]. The extensive debris generated by this event can also be observed in Figure 1.1, as a sharp increase in the fragmentation debris in 2007 [5].

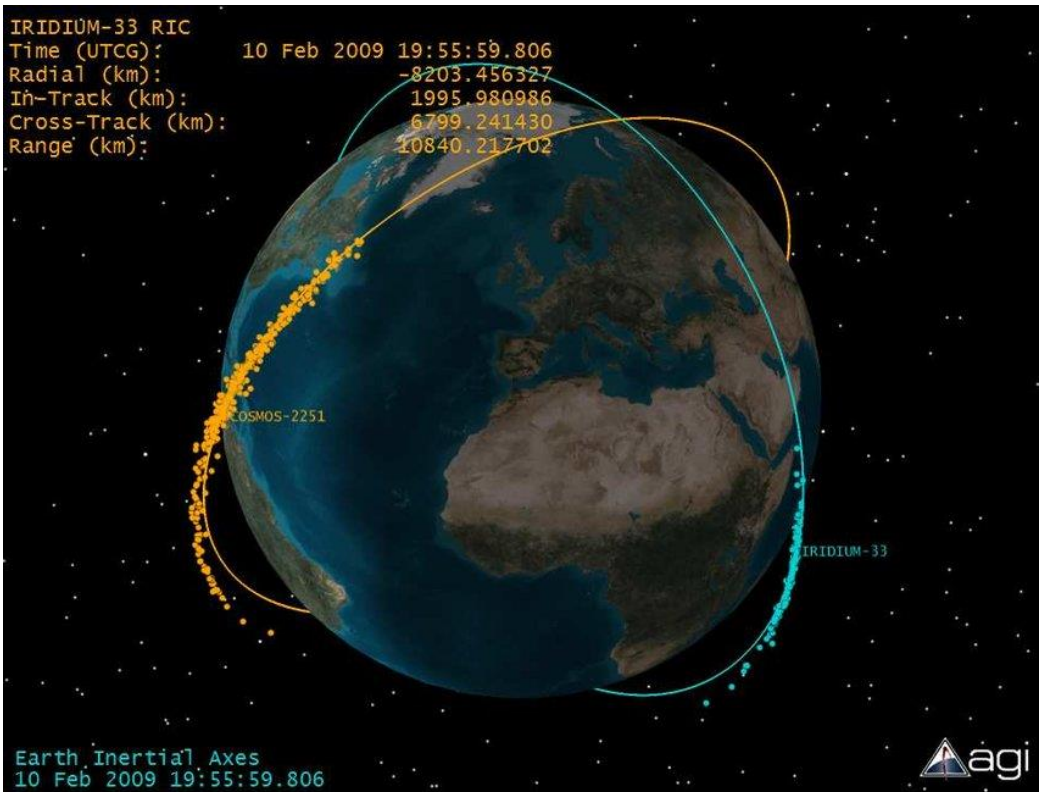

Figure 1.2. Simulation of the debris generated by the Iridium 33-Kosmos 2251 collision, three hours after impact [7].

These active satellites and inactive satellites and debris orbiting Earth are collectively termed RSOs. As of 2022, over 25000 RSOs greater than 10 cm exist in LEO alone [5], and the space environment is becoming occupied by these RSOs at an alarming rate. Given the massive increase in the expected number of satellites mentioned earlier, the population of the space environment will only continue to increase. Along with the increase in objects comes an increase

in the probability of collision between space objects. Given the incredible speeds at which debris travels, even millimeter-sized objects pose a threat to satellites and can cause devastating damage. Debris between the 1 cm and 10 cm range are the most concern, however, as they can severely cripple any spacecraft they hit and are too small and numerous to track [9]. These collisions can cause a cascading effect of collisions, known as Kessler Syndrome [10]. Such an event would create a dangerous and impermeable cloud of debris orbiting Earth, restricting the modern necessities provided by existing satellites and effectively preventing further satellite launches for many years.

In order to protect our current satellites and prevent such a catastrophe from occurring, operators of satellites need to have a well-defined understanding of the objects around them, including their orbital parameters, size, mass, and materials. The activities involved in collecting such information are termed SSA, and with good SSA, operators can perform collision avoidance maneuvers to proactively prevent collisions.

Many nations have developed their own systems and networks to provide them with SSA data. The United States, Europe, Russia, China, Australia, Canada, Iran, South Africa, South Korea, and Thailand all have some form of SSA capabilities [11]. Currently, the most comprehensive and open source SSA system of all these nations belongs to the United States and is known as the Space Surveillance Network (SSN) [11]. This network consists of a multitude of sensors, including terrestrial radar and optical sensors, as well as space-based sensors [11]. Using these instruments, measurements of satellites in space are made, which are then processed to identify the satellites and their orbital parameters. These identifications are included in a catalog and are updated as new detections are made.

Sharing SSA data is crucial to prevent the scenarios discussed above. Given the militaristic nature of data collection, SSA data is often shared in ways to foster strategic cooperation. The United States shares some of the information collected by the SSN through various programs, in tiers of data quality. The United States SSA Sharing Program exists to share otherwise restricted SSA information with close military allies, including Canada, Australia, and the United Kingdom, as well as other allies [11]. To prevent collisions, the United States also shares data with various organizations, launch providers, and satellite operators, though this data is less restricted and thus less comprehensive [11]. Finally, the lowest tier of data that the United States shares is made available to the public including researchers, through a website called Space-Track.org [11]. This data is in the form of Two-Line Element (TLE) sets, which is data representing the orbital parameters of RSOs.

While this data sharing program represents a monumental achievement in sharing SSA data to protect space assets, the program is not without its flaws. Firstly, many objects in the catalogue are lost with time. CelesTrak.org, a website that uses the information provided through Space-Track.org, maintains a list of RSOs that have been lost in the catalogue, but according to their last known orbital parameters, should not be [12]. Another example of this is a satellite that had been identified recently after being lost for 25 years [13]. These lost satellites could pose serious collision risks to operators without access to better SSA data. These same consequences can also come from the erroneous data in the catalogue, such as misidentifications and satellites that are not catalogued whatsoever [14]. Another flaw with the program is the amount of important data left out of the catalogue. As mentioned earlier, much of the data available publicly is limited. This not only applies to the number of RSOs covered, but also the availability of raw data from

sensors. This restricts the amount of processing that can be done using this data and limits the identified RSOs to what can be found by the database providers. Furthermore, the publicly available RSO catalogue may not even be suitable for conjunction assessment to avoid collisions. Space-Track themselves declare that the TLE generated for the public catalogue should not be used for conjunction assessment [15]. Finally, the catalogue available from Space-Track.org is just a single catalogue, and while many alternative catalogues exist for the public to use, there is not a single, unifying global framework to support SSA.

The long-term goal of this research is to develop such a unifying framework for open source, community-driven RSO identification, using data submitted from anyone globally. Figure 1.3 outlines the steps involved in such a framework, using optical imagery. The framework would begin with receiving raw data in the form of optical imagery. After verifying the integrity of the data, RSO detection algorithms would be used to find RSOs within the imagery. These detections in pixel coordinates would then be converted to an inertial frame of inference, producing detections in Right Ascension (RA) and Declination (Dec) coordinates. Further algorithms would then use these detections to determine the orbital parameters of the RSOs. Finally, matching algorithms would then correlate these orbital parameters to orbital parameters of known RSOs and return the best matches. In cases with poor matches, further detections could be used to improve the confidence of the match, or new identifiers could be instantiated for new RSO detections. This framework would allow anyone to contribute to and have access to SSA data, both raw data where available and TLE. This overcomes many of the limitations imposed by the publicly available catalogue as described above. Beyond these benefits, individuals could use the raw data to develop their own algorithms, which could be used to iteratively improve a catalogue

developed in this framework. Additionally, having a transparent, openly accessible catalogue could serve as a form of verification for other catalogues, and for the developed catalogue itself. Finally, more frequent detections of RSOs improve their positional accuracy, reducing the error in calculating conjunction events [16].

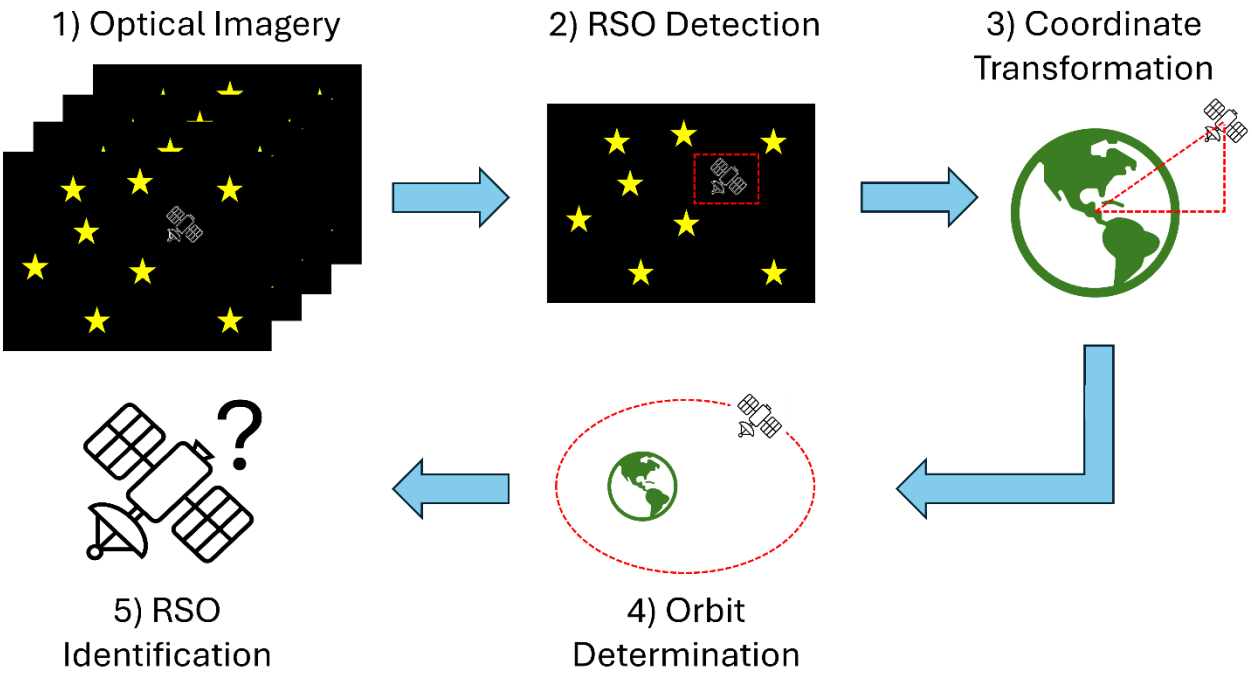

Figure 1.3. Steps involved in an open source, community driven RSO identification framework, using optical imagery.

To start the development of this framework, algorithms need to be created to process optical imagery for RSO detection. However, there are challenges with developing such algorithms. Firstly, there is limited access to real, open source, labelled optical images, as mentioned before. This data is necessary for training and testing a robust algorithm to use in the above framework. Researchers often resort to synthetically generated data for RSO detection algorithm development and testing [17,18]. Moreover, the available data can be difficult to label and

process for RSO detection algorithms using existing detection techniques. These images can be distorted due to the presence of Earth's limb or the Milky Way, have uneven lighting conditions across a sequence of images and even within an image, and are low in resolution, making RSOs harder to find. An example of these images and their challenges can be seen in Figure 1.4. Lastly, to further support the RSO identification framework, these RSO detection algorithms should be able to run onboard the spacecraft in real-time, so that data can be collected and downlinked efficiently, consisting only of SSA data that is of interest. Real-time RSO detection algorithms would also allow operators to respond quickly to potential collision threats. These algorithms could be added to payloads as an additional function, or as dedicated satellites, providing fast response times to potential threats.

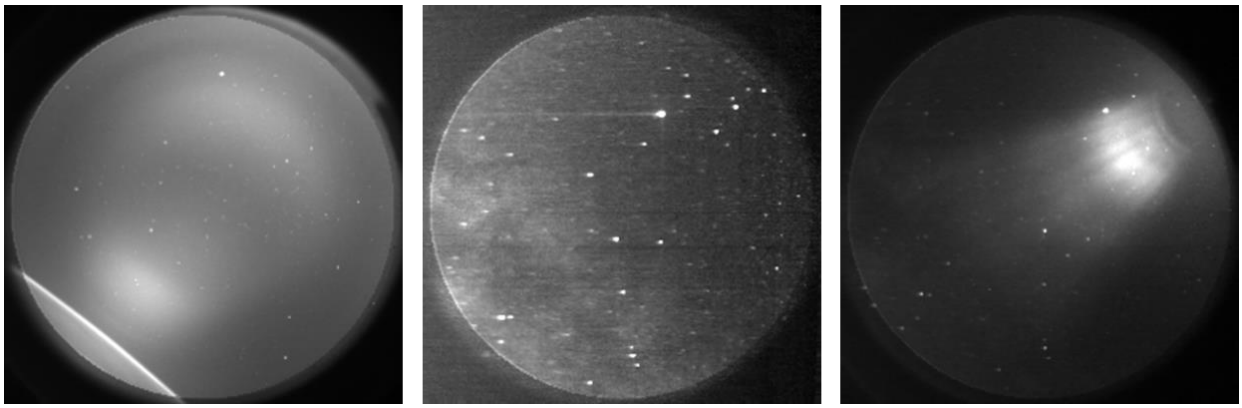

Figure 1.4. Examples of publicly available starfield images. These images can be challenging to process, given that they can contain the Earth's limb (left), the Milky Way (center), and have uneven lighting conditions (right).

1.2. Research Objectives

This thesis was developed with an emphasis on engineering research objectives. The primary objective of this research is to develop RSO detection algorithms to improve SSA capabilities.

However, as mentioned earlier, the availability of data for algorithm development, and the necessity for these algorithms to run in real-time involve additional challenges. Therefore, to develop these RSO detection algorithms, the following research objectives were identified:

- To collect and label optical RSO imagery in a variety of imaging conditions to develop preliminary datasets for RSO tracking algorithm development.
- To develop RSO tracking algorithms to detect RSOs within the data.
- To demonstrate a dual-purpose payload concept for SSA, involving an RSO detection algorithm running in real-time as a secondary payload function.

Given the lack of readily available SSA data, the first step involves obtaining this data. This data can be collected from the ground, using ground-based telescopes paired with a computer for storage of images and camera parameter manipulation. Alternatively, this data can be collected above ground, using platforms such as stratospheric balloons and spacecraft to overcome unwanted atmospheric effects. This thesis will focus on data collected from stratospheric and space-based platforms. The next step involves labelling the collected data to serve as truth data when developing algorithms. The precise and efficient labelling of this data can be challenging, given the small and numerous objects within each of the images. After developing a labelled dataset, RSO tracking algorithms can be developed and tested using the images, using metrics such as the precision and recall of the algorithm to determine its performance. Lastly, after developing a robust RSO tracking algorithm, the last step involves creating a dedicated payload to track RSOs in real-time, to serve as a technology demonstration.

1.3. Contributions

The following are contributions made to the improvement of SSA done through this research.

- The development of the software of a payload designed to autonomously image RSOs from the stratosphere, processing GPS data in real-time to adapt image capture to the flight conditions.
- An overview of existing algorithms for detecting RSOs within various optical imagery.
- An extensive review of the imagery from the Fast Auroral Imager (FAI) instrument onboard the Cascade SmallSat and Ionospheric Polar Explorer (CASSIOPE) satellite, collected during 2023, extracting sequences suitable for RSO detection, focusing on criteria such as scene illumination characteristics and RSO motion and brightness variation.
- The curation of a novel 878-image RSO dataset made from annotating the RSOs in the extracted sequences, providing bounding box coordinates, sizes, and unique identifiers for every RSO in every sequence.
- The development of a novel algorithm combining multiple analytical techniques to process the images for RSOs.
- The development of a four-tool labelling suite to efficiently label images containing RSOs, automating much of the labelling process and producing labels in the form of bounding box coordinates and size for three classes of objects, including stars, RSOs, and noise.
- The curation of a novel 500-image data using the developed labelling tool suite, made from annotating the stars, RSOs, and noise in the images, producing both masks and text files as truth data, to serve as a foundation for a benchmark dataset.

- The proposal of a combination of metrics to effectively and fairly compare RSO detection and/or tracking algorithms.
- The development of an RSO tracking algorithm using a track-by-detection framework, making use of Machine Learning (ML) to detect objects and multiple custom analytical techniques to track objects.
- The development of a flight-version RSO detection algorithm constructed using components of previously developed RSO tracking algorithms and enhancements for real-time operation.
- The demonstration of a dual-purpose payload onboard a stratospheric balloon platform, performing RSO detection, running alongside an existing AD algorithm.

1.4 Thesis Outline

This thesis has been developed as a combination of several peer-reviewed research papers which have been submitted or are in the process of being submitted. In Chapter 2, the stratospheric optical RSO data collection payload is described and evaluated. In Chapter 3, the analytical RSO detection algorithm is presented and evaluated, alongside the 878-image dataset. In Chapter 4, the data labelling tool suite, 500-image preliminary benchmark dataset, as well as the ML-based track-by-detection RSO tracking algorithm are described. In Chapter 5, the stratospheric dual-purpose AD and RSO detection payload is described and evaluated. In Chapter 6, a summary of the previous chapters is provided, alongside with a discussion of future work.

2. Stratospheric Night Sky Imaging Payload for Space Situational Awareness (SSA)

This chapter is based on a paper [19] describing a payload, namely Resident Space Object Near-space Astrometric Research (RSONAR), which sought to autonomously acquire images of the night sky from a stratospheric balloon platform for SSA. This mission and payload were developed given the lack of real, high-quality, readily available optical images, necessary for the development of RSO tracking algorithms. Additionally, the payload and mission served as technology demonstration missions to prove that stratospheric ballon platforms are suitable for SSA purposes. Despite hardware failures during the mission, over 93000 images of the night sky were captured, with 100s of RSOs visible through manual inspection of the images. Image processing algorithms developed by the coauthors of the paper were able to detect 100s of RSOs as well, in preliminary findings. Further research is being conducted in the laboratory to investigate the data's quality, as well as using these images for other RSO detection algorithms.

I developed the software that was used to autonomously control the camera, including boot-up operations, sensor processing for mode selection, camera parameter setting, image acquisition, and payload status data saving. I was involved in the assembly of the payload, the planning of the mission, payload-gondola integration, payload health monitoring during the mission, and post-mission operations. I was a member of the crew that carried out these tasks to successfully oversee the payload's launch on a stratospheric balloon from Timmins, Ontario, in August 2022. I prepared all figures, and tables, and wrote the manuscript. Andrea Vallecillo Baires and Siddharth Dave provided me with their image processing results for the results section. Gabriel

Chianelli provided me with information about the electrical system of the payload. Ryan Clark, Siddharth Dave, and Regina S. K. Lee created the metrics and documentation for the design of the mission. All coauthors provided feedback and revisions for the manuscript.

2.1. Introduction

The space environment has seen a rapid increase in the number of RSOs, ranging from tiny flecks of paint to entire satellites. In recent years, the space industry has trended toward mega-constellations of smaller satellites, further increasing the population of the space environment. RSOs can reach orbital speeds of over 7 km/s, traveling fast enough to cause devastating damage in collision events. Though once believed to be unlikely, the rapidly increasing RSO population is increasing the chances of in-orbit collisions. Such an event has already been realized in February 2009, when Iridium-33, an American communications satellite, collided with Kosmos2251, a defunct Russian military satellite, generating almost 2000 fragments with diameters of at least 10 cm, and thousands more smaller pieces [6].

With the threat of future in-orbit collisions growing as the number of RSOs increases, satellites must be tasked with collision avoidance maneuvers to prevent impacts. To develop such strategies, operators must first be aware of nearby RSOs, having detailed information such as the type of object and orbital parameters. SSA refers to various programs and initiatives to advance such knowledge and can be accomplished by imaging RSOs and applying image processing algorithms to extract the required information about the objects.

In addition to optical imaging, radar systems have been used extensively for SSA activities. One example of such a system is the Italian Bistatic Radar for LEO Tracking (BIRALET) system

[20]. This radar system's transmitter works in the P-band, operating at 410–415 MHz, able to capture LEO RSOs in beam parking mode or tracking mode. Another example is the recently developed CHEIA SST Radar, a Romanian system leveraging existing parabolic antennas, aimed at strengthening the European Union Space Surveillance and Tracking (EU SST) effort [21]. This C-band radar system operates in the 3.6–6.4 GHz range, also able to track RSOs.

Many techniques have been developed to collect and process radar data, obtaining information critical to SSA. At Fraunhofer Institute for High Frequency Physics and Radar Techniques FHR, a team of scientists operate a radar system, the Tracking and Imaging Radar (TIRA) [22]. Information such as orbital elements, motion parameters, target shape, and mass and material properties have been gathered from this system. An Initial Orbit Determination (IOD) algorithm has been developed to determine the state vector and orbital track of satellites using the Italian Bistatic Radar for LEO Survey (BIRALES) system [23]. The algorithm is being further developed to improve catalog correlation. Using radar data alongside optical data, a concept referred to as data fusion, is a promising technique for improving the information gained from RSO detections [24].

In terms of optical systems, traditionally, SSA activities typically focus on imaging RSOs from ground-based telescopes, such as the National Aeronautics and Space Administration's (NASA) Michigan Orbital Debris Survey Telescope (MODEST) [25]. Such a large-scale, narrow Field-of-View (FOV) telescope can take high-resolution images of RSOs in Geosynchronous Earth Orbit (GEO) [26]. The narrow FOV telescopes have a large integration time, for example, 5 s is typical, integrated with an active tracking methodology. The light from an object is accumulated in a narrow spatial region of the image, improving the sensitivity of the instrument. Active

tracking is suitable for GEO objects, which move relatively slowly and more predictably with respect to a ground-based observer, making them easier to keep in the FOV of a telescope. However, the tracking requirements for a LEO observation introduce complications in the tracking system, since RSOs in LEO tend to move much faster than RSOs in GEO and have more uncertainty in their trajectories due to effects such as atmospheric drag. Furthermore, the narrow FOV would limit the number of RSOs that can be seen in a single image. Coupled with large design and operating costs, a need exists for a low-cost alternative with space-based operation, short integration time, and a wide FOV imager. Such an imager would also be useful in LEO RSO detections [27,28]. The imaging strategy proposed in this chapter is designed to survey many RSOs with a wide FOV, that is cost-effective and suitable for both ground and space applications [29].

In comparison to narrow FOV imagers, RSO detection and tracking using a wide FOV imager has been studied with limited results in the past. Clark et al. developed a simulator to realistically simulate images taken by low-resolution, space-based imagers [30]. Clemens demonstrated the feasibility of RSO detection from a wide FOV imager using images from the CASSIOPE satellite's FAI instrument [31]. Dave et al. then examined an in-orbit RSO orbit estimation method using star tracker cameras [32]. Sease et al. predicted RSO motion from a simulated star tracker imager, demonstrating that angular rates can be extracted from streaked imagery without the use of star catalogs, and even extend image processing to detect RSOs [33]. Spiller et al. also examined RSO detection from star-tracker images using simulated datasets [34]. Wide FOV telescopes have also been considered for ground-based RSO imaging and subsequent image

processing, as demonstrated by Hasenohr [35], and extended to wide FOV ground-based arrays by Fitzgerald et al. [36].

In this chapter, we present the overview of the RSONAR payload with a wide FOV star tracker-like camera with relatively low resolution, built with Commercial-Off-the-Shelf (COTS) components. The payload was flown on a stratospheric balloon as part of the Canadian Space Agency's (CSA) stratospheric balloon program, STRATOS. There have been numerous remote sensing and technology demonstration missions onboard stratospheric balloons, carrying out experiments at suborbital altitudes as test missions for orbital demonstrations. Examples of such experiments include University of Alberta's Ex-Alta 2 mission, designed to monitor wildfires with a multispectral imager [37], and University of Minnesota's SOCRATES payload, studying x-rays and gamma rays [38]. To the best of our knowledge, the SSA mission described in this chapter was the first mission of this nature with the ambitious goal to image night sky star fields to detect, track and identify RSOs in low-Earth orbits. We aimed to demonstrate the feasibility of using wide FOV, commercial-grade cameras for RSO detection as a first step toward a CubeSat mission The next step in this research is to improve the payload (both hardware design and algorithm development) for space-based RSO observations from LEO, to advance SSA capacities by providing space-based RSO detection data to complement ground-based observations.

2.2. Mission Concept

The RSONAR mission was a technology demonstration of a commercial-grade star tracker for SSA applications. The main research objective was to take images of RSOs from the stratosphere

(approximately 40 km in altitude) using a low-resolution, wide FOV star-tracker-like camera. The developed system used a 29.7-degree FOV, contrasting typical systems using 1-degree or smaller FOVs. The mission objective was to progress the technology readiness level of a flight payload that incorporates a commercial-grade star-tracker-like camera. The payload demonstrated in flight that a dual-purpose functionality to image RSOs in addition to the star tracker's primary purpose of determining spacecraft attitude is achievable. Mission success criteria were defined along three sequential objectives: RSO detections per hour (minimum 1 per hour, ideal 10 per hour), RSO tracking accuracy (minimum 15 arcseconds, ideal 1.5 arcseconds) [39], and RSO identification (minimum 80%, ideal 95%) [40].

The RSONAR payload consists of COTS hardware. This includes the camera and lens, the System-on-Chip (SoC) development board, the Global Position System (GPS) sensor, cabling, harnesses, and connectors. The survival of these components would confirm that COTS hardware is suitable for use in payloads for high-altitude missions.

The payload interfaces with CSA's gondola, designed to operate at an altitude of approximately 40 km, coast for 4 to 8 h, and descend. The duration of the flight is typically about 13 h, for which the payload operates during the night portion of the flight. During the flight, the payload takes images of the sky, varying the camera's resolution, exposure, and delay depending on the altitude of the balloon, by using altitude values obtained from an onboard GPS to change camera parameters. Operating the payload in this way ensures the largest number of high-quality images to be taken during optimal observation conditions, while the balloon is coasting at the maximum altitude.

2.3. Technology Description

2.3.1. Electronics

The Xilinx PYNQ-Z1 SoC development board (Figure 2.1), sourced from Digilent, located in Pullman, USA, was used as the Onboard Computer (OBC). The system consists of a Zynq-7000 SoC, which itself consists of Field-Programmable Gate Array (FPGA) fabric and a dual-core ARM Cortex-A9 processor [41]. The processor hosts the operating system, which contains the image capture application and driver management, while the FPGA fabric is used to control the pins used for the GPS module. During testing, the board's ethernet and USB ports were used for debugging purposes. The Arduino headers and PMODA pins were used to interface with other sensors and subsystems. The PYNQ-Z1 board was provided with a Linux-based Operating System (OS), which was used to simplify software design. The captured images were stored in the same partition as the OS. Extensive power-out testing showed that there was no need to mount the OS into its own read-only partition.

A ZED-F9P Global Navigation Satellite System (GNSS) module [42], integrated into a SparkFun GPS-RTK2 board, was used to receive GPS signals, providing information such as time, altitude, longitude, and latitude. This module is connected to the PYNQ-Z1 board via its Arduino headers.

An Innodisk industrial-grade 512 GB microSD card [43] was used for onboard storage, as well as to hold the operating system. The microSD card was chosen for its operating and survival temperatures, capable of functioning at −40 °C, which the system could be subjected to in a stratospheric balloon mission.

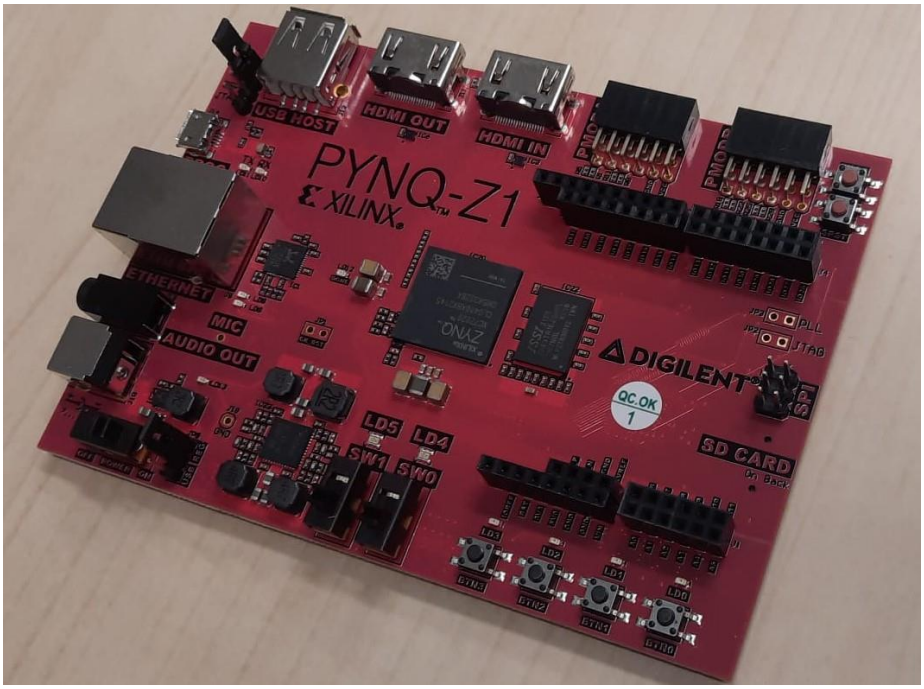

Figure 2.1. Xilinx PYNQ-Z1 FPGA development board.

2.3.2. Optics

The pco.panda 4.2, sourced from PCO Imaging, located in Kelheim, Germany, a scientific Complementary Metal-Oxide Semiconductor (sCMOS) camera [44] was chosen for this mission. This compact monochrome camera is similar in FOV, resolution, and exposure time to a typical star tracker used for AD on satellites. The camera features a maximum resolution of 2048 × 2048 pixels at a variety of exposure times.

Paired with the camera is a wide FOV lens, given the requirement for RSO detection. The lens used is the ZEISS Dimension 2/25, sourced from Carl Zeiss Industrielle Messtechnik GmbH, located in Oberkochen, Germany, having an aperture of 5.7 cm [45]. Combined with the pco.panda 4.2 camera, a full-angle FOV of 29.7 degrees is obtained.

2.3.3. Structure

The payload structure design was based on a CubeSat form factor, a class of miniature satellite with dimensions of 10 cm × 10 cm × 10 cm [46]. A 2U-form factor was implemented, with total dimensions of 10 cm × 10 cm × 20 cm. However, additional triangular prism segments were added to the payload to host a secondary payload for the mission and to provide a 45-degree elevation for viewing geometry. Figure 2.2 provides a Computer-Aided Design (CAD) model of the RSONAR payload with these additional segments, from multiple perspectives.

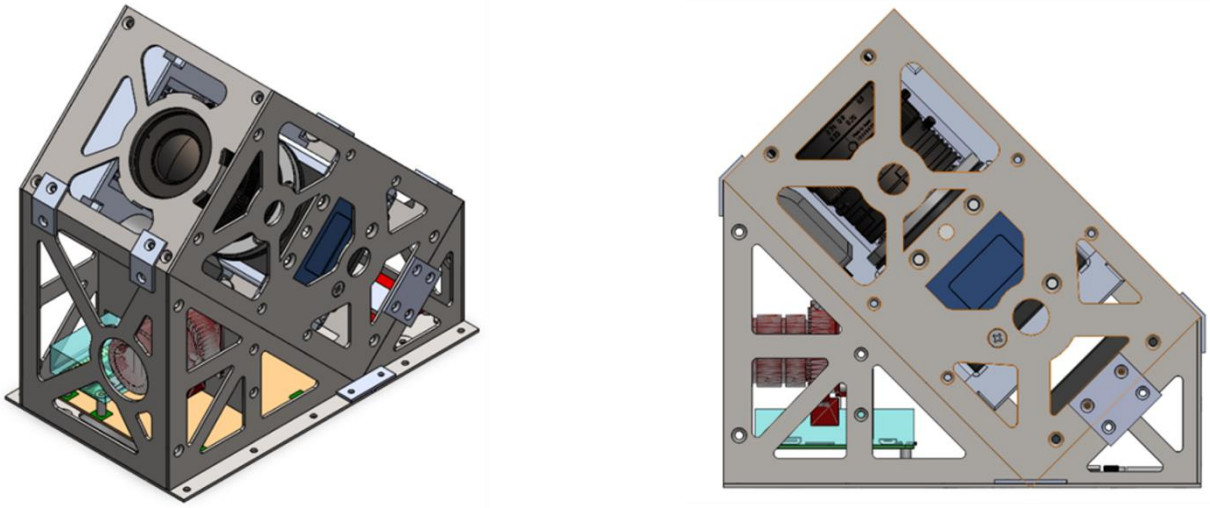

Figure 2.2. RSONAR payload CAD model outlining the structure and some electronics within the payload from the isometric view (left) and the side view (right). Note that triangular prism-like segments are added to the plain 2U segment.

The payload itself was mounted to the gondola using a metal interfacing plate. The payload was screwed onto this plate using 10 stainless-steel bolts, nuts, and washers. The assembly was then attached to the gondola using five C-clamps. Figure 2.3 illustrates the payload and interface plate mounted on the gondola for flight.

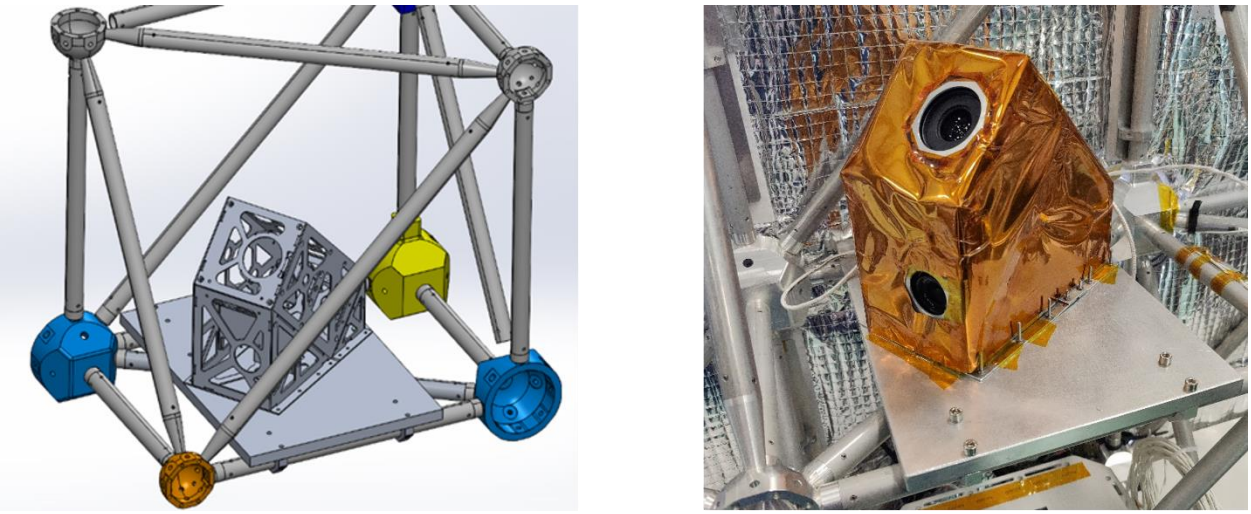

Figure 2.3. RSONAR payload fastened to the gondola. (a) CAD model depiction of the mounting scheme; (b) Actual integration of the payload to the gondola.

The chassis is primarily fabricated from aluminum 6061-T6, while the fasteners are made of AISI 304. The payload's mass (2.1 kg) falls well under the maximum mass listed by the mission requirements. The interface plate, not included in the mass requirement, has a mass of 2.5 kg alone. Table 2.1 shows the masses of the payload components.

Table 2.1. Components of the RSONAR payload and their corresponding estimated masses.

Item	**Mass (g)**
Fasteners (screws, nuts, washers)	80
Aluminum chassis	780
PCB and breakout boards	230
Optics and antenna	955
Wires and cables	100
Total	**2145**

To conform to safety requirements, a simulation was performed to examine the payload and interface plate under a variety of load cases, alongside calculations. For all load cases, neither the payload nor the interface plate was expected to detach, given that the stresses were well below the yield strength of the materials. The Finite Element Modeling (FEM) analysis indicated that the highest stress response was at the corner bolts (not on the payload structure itself) where minimum risk is expected.

The payload does not feature an active thermal control (heating or cooling) system, as the electronics were shown to produce enough heat such that this is not a concern, while the upper temperatures are within the limit. A passive thermal control approach was adapted using an aluminum polyamide covering (shown in Figure 2.3b) as a thermal blanket and reflector, allowing the payload to retain heat during the night and reflect some solar radiation during the day to keep the payload cool. Radiation strike mitigation strategies were not used for this payload.

2.3.4. Communications

To simplify the design, communications to the payload during flight were not implemented. Instead, an autonomous algorithm and onboard storage were used to manage bootup and shutdown, camera settings, payload health, and data storage. In future flights, the National Centre for Space Studies' (French: Centre national d'études spatiales, CNES) communication subsystem, PASTIS, will be used for telemetry and command, using the Ethernet IP network protocol.

2.3.5. Software

The Xilinx PYNQ Z1 FPGA board was programmed with a headless version of Ubuntu, which makes software development and troubleshooting more convenient. The application driving the camera control was written in C++. Existing header files and functions were used to rapidly develop the application. The application makes use of a GPS receiver to determine the time, altitude, and variation over time, which indicates how stable the payload is and thus what quality could be expected from the images. Therefore, the application was designed to take more pictures at a higher quality during steadier flights.

To achieve this high-quality imaging, a mode scheme was programmed. Each mode corresponds to a mode that was expected to be seen during the flight: ascend, descend, coast, and DnD (dawn and dusk). A safe mode was also programmed to account for potential software errors during flight. To enter each mode, a set of altitude values over a period of time from the GPS are analyzed. If various conditions are met corresponding to that mode, the mode is selected, and images are taken for a short duration, after which the condition checking using the GPS values is repeated.

Ascend mode was created to account for the balloon take-off and other altitude increases, during which time imaging would not be ideal. The resolution is reduced to 512 × 512 pixels, and 20, 10, and 10 images are captured with exposure times of 100 ms, 500 ms, and 1000 ms, respectively. A 10 min delay was added at the end of this mode to further optimize storage used in ascend mode.

Descend mode is similar to ascend mode, created to account for the balloon descent at the end of the mission. The settings are the same as ascend mode. This mode was programmed separately in case changes are desired between ascend mode and descend mode. This was also the mode intended to be used when the balloon is in a powered state but still on the ground.

Coast mode is to account for the condition where the balloon is floating in the stratosphere at a steady altitude. This would be an ideal case for image taking, and thus the image resolution is increased to 1024 × 1024 pixels, and 20, 10, 10, and 10 images are captured with exposure times of 100 ms, 500 ms, 1000 ms, and 5000 ms, respectively. This was done to get a variety of data in this mode. A 10 min delay was also added at the end of this mode.

DnD mode is for image taking during the ideal observation time—the hours preceding dawn and following dusk, during which the light reflected off satellites would be greatest from the perspective of the camera. The resolution in this mode is varied between 2048 × 2048 pixels and 1024 × 1024 pixels, capturing 20 images for each of those resolutions, while the exposure time is set at 100 ms. A delay of just 15 s was added to the end of this mode. The mode was designed such that the most pictures would be taken in this mode.

Lastly, a safe mode was programmed for the event that the GPS data could not be read, invalid, or if a system error is detected. This mode was developed to be as simple as possible, and thus a single resolution and exposure time were used for this mode. The resolution is set to 2048 × 2048 pixels, capturing 10 images with an exposure time of 100 ms. The highest resolution was used to maximize the information that could be acquired. A short exposure time was used over a long exposure time because the short exposure images could be stacked in post-processing to create

streaked images for RSO streak detection algorithms, while they could be used individually for RSO point-tracking algorithms. A delay of 20 s was added to the end of this mode.

The software is to be executed on bootup, restarting in the event of a power outage during the mission to ensure that the Linux kernel can properly shut down and reboot. A power-down signal is issued by the team through the CSA's on-ground Power Distribution Unit (PDU) interface at the end of the mission to safely shut down the payload. Figure 2.4 outlines the camera operation from bootup.

2.4. Implementation

Figure 2.5 below outlines the connections between the major components of the payload. The Xilinx PYNQ Z1 FPGA board connects to the pco.panda 4.2 camera using a USB-A to USB-C cable, which offers sufficient power to the camera as well as data transfer of images. The GPS receiver module connects to the FPGA board via its Arduino headers. The custom sensor board connects to the FPGA board via the PMODA pins. The SD card hosts the operating system, which in turn holds the images, hardware drivers, and sensor interfaces. Power is delivered via the onboard power jack.

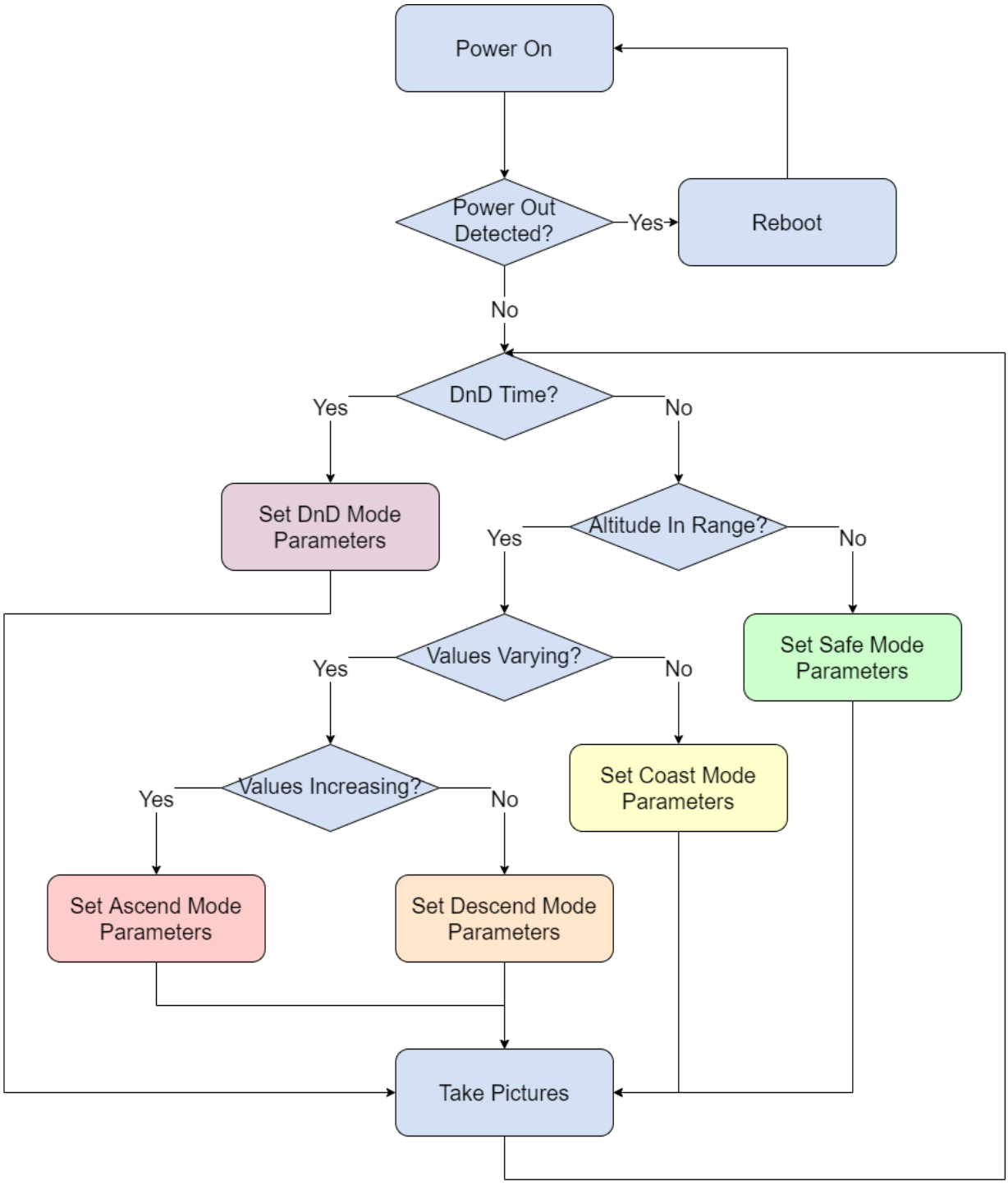

Figure 2.4. Block diagram outlining the autonomous algorithm used to power on the payload, check the altitude and time, and take images with pre-defined camera parameters corresponding to the mode.

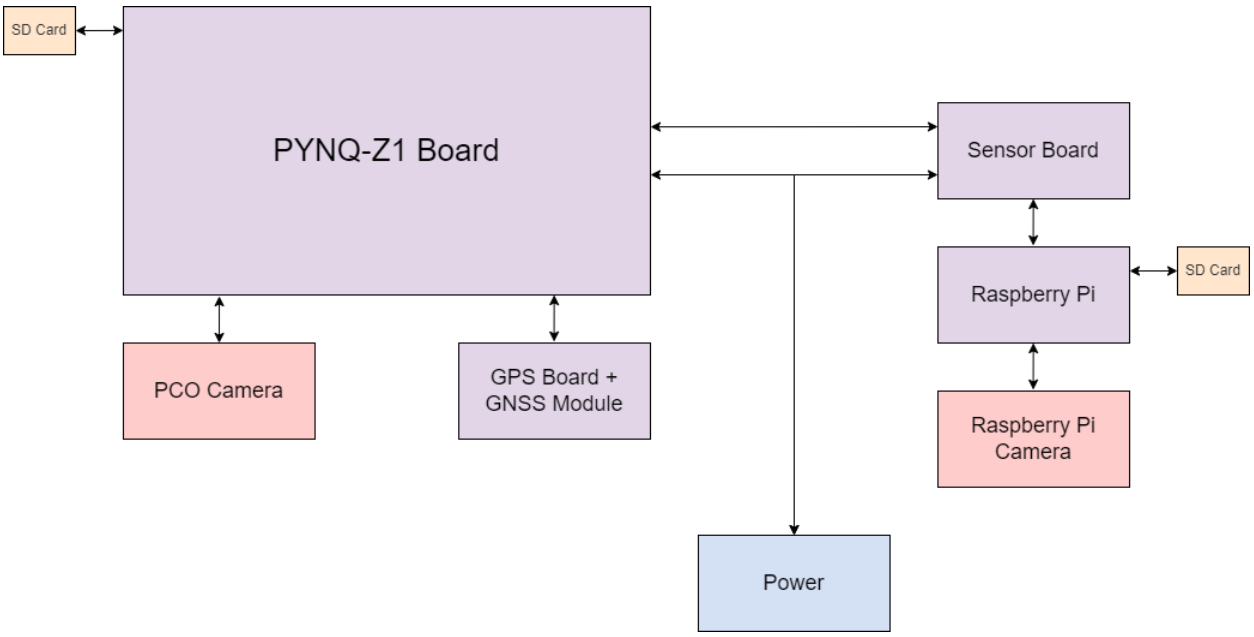

Figure 2.5. Diagram outlining the connections between the electronics in the RSONAR payload.

Power from the gondola was provided from the PDU, connected via a female PT02E-12-3S connector. The RSONAR payload itself implements the same female connector, thus a male-to-male connector was chosen, ensuring the correct pinouts for the MIL-DTL-26482 standard. Within the payload, the power is routed through two fuses, two tantalum capacitors, and a DC-DC converter, as outlined in Figure 2.6. The fuses are rated at 3.2 A, derated based on a study recommending fuses to be derated by 50% for vacuum environments [47]. The capacitors are rated at 100 uF and 220 uF, respectively. The DC-DC converter converts incoming power at a different voltage to a steady 12 V output. The PYB20-Q48-S12-DIN from CUI Inc is used to achieve the power regulation, and convert an incoming voltage in the range of 18 V to 75 V down to a voltage of 12 V, and supplying a current of 1.667 A. All components in PDU meet a minimum temperature specification of −40 °C. Figure 2.6 below outlines the power connectors and components.

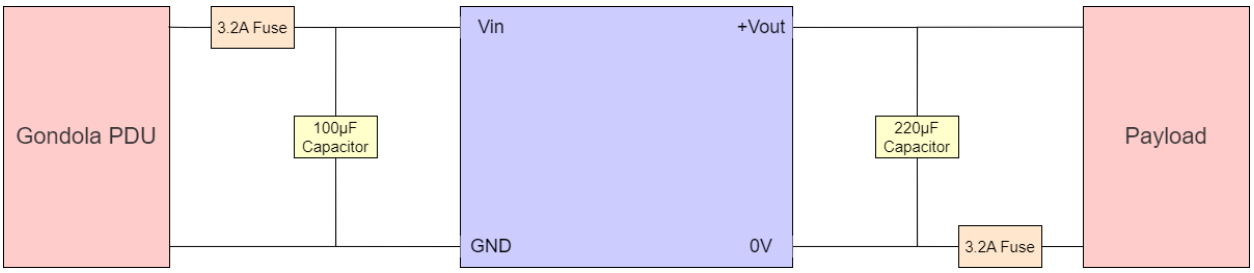

Figure 2.6. Diagram outlining the electrical components used on the power distribution unit.

2.5. STRATO-SCIENCE 2022 Campaign Overview

The STRATOS stratospheric balloon program aims to provide Earth and space science and engineering research community the opportunity to test newly developed payloads, collect data from a near-space environment, and train next-generation scientists and engineers. During the campaign in August 2022, named STRATO-SCIENCE 2022, four balloons were launched from the Timmins Stratospheric Balloon Base, carrying a variety of payloads. Various flight profiles were available, reaching as high as 40 km in altitude [48]. Figure 2.7 depicts the balloon and gondola carrying the RSONAR payload.

During integration on-site, GPS data collection from the payload configuration proved difficult, as there was no access to a clear GPS signal with the thermal blanket in place. Consequently, in the absence of a reliable GPS signal, the payload was reconfigured to operate exclusively in safe mode. Safe mode was modified to capture 27 images at a 1024 × 1024 resolution, using the same 100 ms exposure time. The resolution was decreased so that the number of images captured per set could be increased, and so that the delay between sets could be decreased to 4 s. These parameters were chosen to maximize microSD card usage.

Figure 2.7. Pictured in the foreground is a smaller balloon used to keep the attached gondola steady upon launch. In the background is the larger balloon being inflated to launch. The larger balloon expanded even further as it ascended into the stratosphere.

The RSONAR payload was manually switched on 30 min before launch. The balloon carrying the payload launched at 11:37 PM local time on 21 August 2022, from the Timmins Stratospheric Balloon Base at the Victor M. Power Airport. The payload's current draw, transmitted from the balloon in real-time, was used as the method of monitoring the status of the payload since expected current draws were previously determined in testing. For the duration of the flight, the current draw was monitored, and no abnormal behavior was detected.

The payload was switched off manually at 8:30 AM local time the next day. The balloon continued to fly into the day, collecting valuable near-space data for all other payloads, and landed at 1:10 PM local time.

2.6. Results

2.6.1. Imaging Modes

Given that the payload was forced into safe mode due to an unreliable GPS signal, the algorithm's mode-switching function could not be observed. However, given the altitudes and changes in altitudes observed through post-mission data, the algorithm would have correctly changed modes to account for the balloon's ascent, coast, and descent. This was tested by running the algorithm on the GPS data provided by the CSA, creating a script to feed the GPS data to the algorithm at the expected frequency and verifying that the algorithm determined the correct mode to switch to. For future missions, the GPS will be relocated outside of the payload and clamped to the gondola chassis where a clear signal can be received.

2.6.2. Preliminary Image Processing

Over 93,000 images were collected during the active flight by the RSONAR payload. Several algorithms have been developed to detect RSOs from images and return statistics. Detailed analysis of the RSOs detected in the images, both theoretically possible and experimentally determined, will be a part of future work. Preliminary analysis is given below.

2.6.2.1. RSO Streak Detection

One of the algorithms is an RSO streak detection algorithm, which aims to scan large image databases and find sequences that may contain RSOs through bulk processing. The algorithm performs bulk processing to preserve time and computational resources. Bulks of sequences are stacked by weighted addition to create a single image. These images are stacked without compensating for star drift, given the minimal movement of stars in these sequences. Image processing techniques such as Wiener filtering, Canny edge detection, and thresholding are applied to the image to reduce noise and detect objects. Smaller objects such as stars are removed by a size threshold and only larger objects such as RSO streaks are left behind. Images that contain a potential object of interest are then considered to be ‘Priority 1′ whereas images with just stars are considered ‘Priority 2′. The algorithm can also detect large obstructions such as Earth’s limb and classifies those images as ‘Priority 3′. Overall, the goal of this algorithm is to reduce the need for manual inspection of databases, which can be tedious and time-consuming. Instead, the classification algorithm provides a snapshot of which image sequences may contain an object of interest.

Before processing the image set, the first 5400 images were discarded, since the balloon was ascending during this time, causing even stars to streak in the images, resulting in many false positives. From the remaining images, the first 12,130 images were processed as a preliminary

test of the algorithm. Since images were captured with a 100 ms exposure time, multiple images in the sequence needed to be stacked together into a single image so that RSOs would appear as streaks. The reduced image set was stacked into sequences of 27 images. Processing the images with the streak detection algorithm returned 51 sequences (with 27 images per sequence) with RSO streaks in them, with an accuracy of 90%. The algorithm was executed again, this time stacking 9 images at a time instead of 27 images to form a single image. This resulted in 69 sequences (with 9 images per sequence) with RSO streaks in them and an accuracy of 98%. Information such as the length of a streak and the brightness of a streak will be implemented in the algorithm in the future. Further analysis such as known and unknown RSO identification (by correlating detections with an RSO catalog, mission objective three) is currently being conducted by developing an IOD algorithm. Figure 2.8 shows an example of the resulting image created by stacking images together. Stars appear as stationary light sources, while RSOs appear as streaks in the image. Multiple streaks are visible, corresponding to multiple RSOs.

2.6.2.2. RSO Point Detection

Another algorithm performs RSO detection and tracking by considering the movement of the RSO points through the images, temporally [49]. The algorithm consists of a Convolutional Neural Network (CNN) combined with a graph-based Multiple Object Tracking (MOT) algorithm. The CNN takes images as input and classifies them as containing detections (RSOs or stars) or not containing detections. The images classified as detections are then used as inputs to the next algorithm, which uses a k-partite graph layout (where k is the total number of images in a sequence, and each detection is a node in the graph). This algorithm distinguishes RSOs from stars by groupings of angular motion, since stars move differently from RSOs.

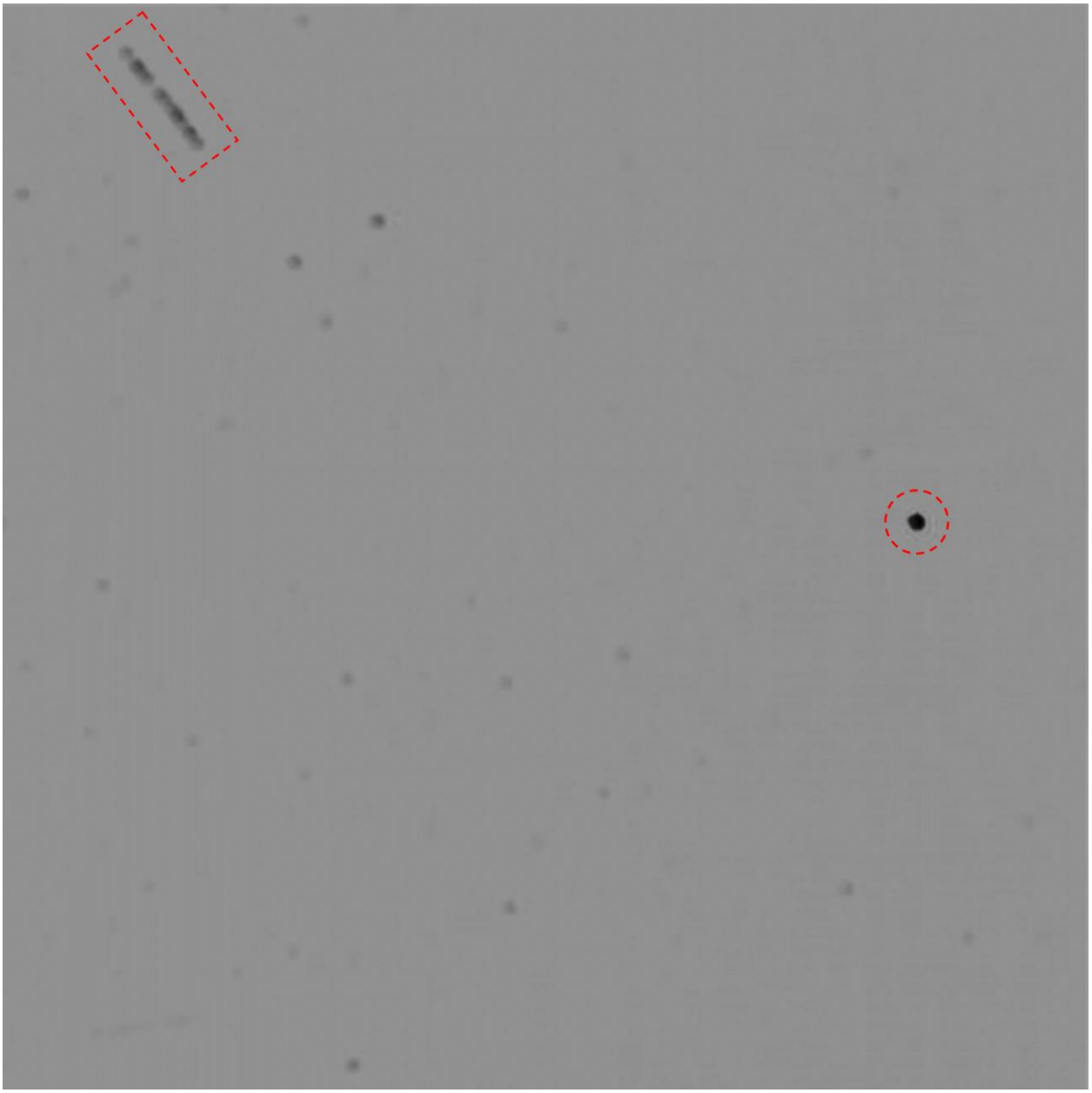

Figure 2.8. Example of an image created from stacking a sequence of images, inverted for easier viewing. Stars appear as black dots (example contained in dashed red circle), while RSOs appear as black streaks (example contained in dashed red box).

Preliminary processing of the RSONAR images using this algorithm has found over 500 RSO detections throughout the images. The performance of the algorithm is still being evaluated, and

metrics will be provided in the future. Further analysis such as RSO tracking accuracy and RSO identification, mission objectives two and three, respectively, is being conducted, and will also be shown.

The number of RSO detections over the course of the flight has proven a successful mission. Further processing is being carried out to extract data from these images, such as attitude estimation, orbit determination, and optical property estimation.

2.6.3. Image Delays

The images taken during the mission were timestamped down to the millisecond. Given that the exposure time was consistently 100 ms, it was determined that there was a significant delay between each of the pictures. Ideally, the timestamps would increase in intervals of 100 ms, but are in intervals closer to 400 ms, indicating delays of around 300 ms. Preliminary investigation has shown that the microSD card used for the mission appears to be the primary cause of the delays. Though chosen for its thermal resilience, the microSD card is of UHS Speed Class 1 (U1), which appears to have been too slow for the images. Testing with a UHS Speed Class 3 (U3) card yielded much smaller delays. The team has not found a microSD card with the same or better operating temperature as the one that was used. Given that the microSD card is a critical component, the delays must be managed in a different way, or a faster, limited operating temperature microSD card must be rigorously thermal tested for use. In the future, we plan to use image compression algorithms or different imaging rates and sizes to reduce the data rate.

2.6.4. Thermal Environment

The PCO camera used in the RSONAR payload had three different temperature sensors at various locations within it. These temperature sensors recorded the PCO camera's internal temperature for the duration that the payload was powered, from 11:00 PM local time to 8:30 AM local time the next day. The gondola itself had a temperature sensor recording the environment temperature for the duration of the flight, from 11:37 PM local time to 1:10 PM local time the next day. Figure 2.9 shows the three internal temperatures of the PCO camera plotted with the environmental temperature recorded by the gondola.

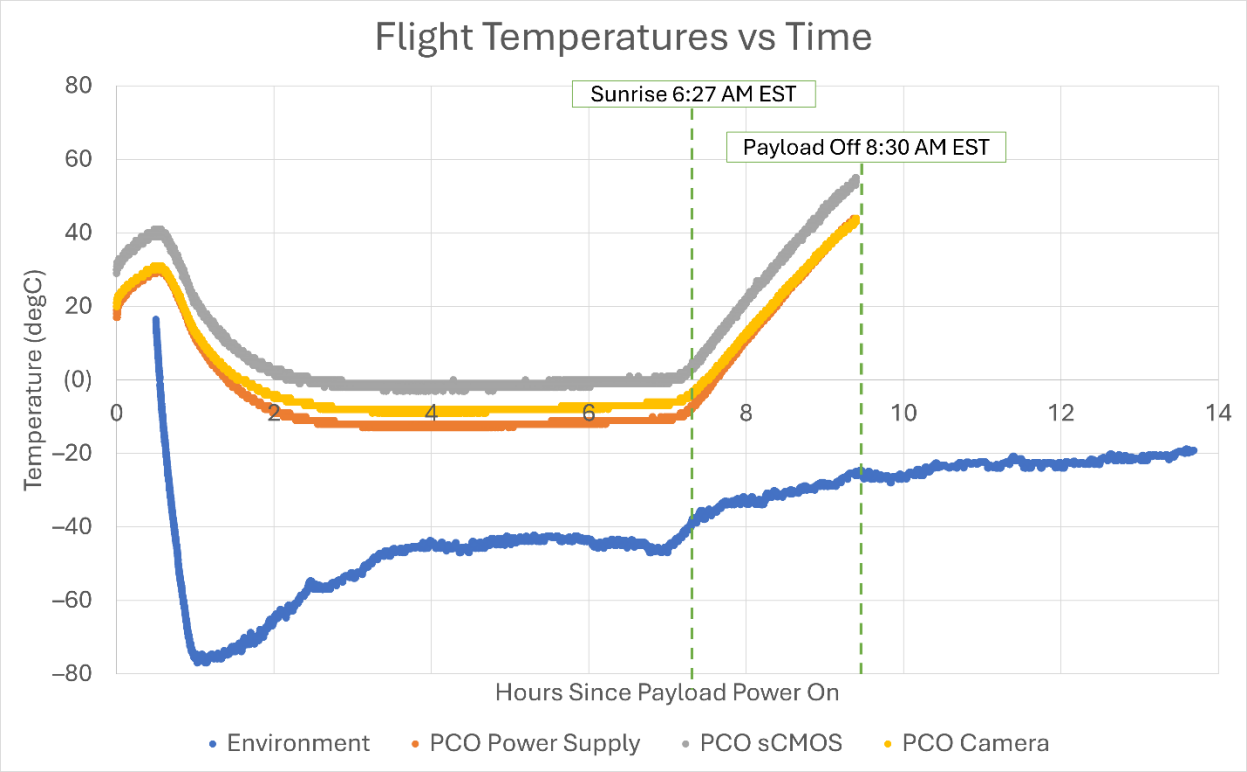

Figure 2.9. Internal temperatures of the PCO camera and environmental temperature, since the payload was powered. Environmental temperature logging continued until 1:10 PM local time.

This work is based on observations with the CNES temperature sensor under a balloon operated by CNES, within STRATO-SCIENCE 2022 and in the framework of the CNES/CSA Agreement.

During payload development, the cold temperatures were the biggest concern, given that most of the components had minimum operating temperatures of −40 °C. This minimum temperature rating was not low enough to account for the almost -80-degree Celsius temperature expected during the ascent through the troposphere. However, all the components of the RSONAR payload survived the mission, and post-mission testing has shown no thermal damage observed in the components.

During the mission itself, the coldest environment temperature observed was −77 °C, while the coldest temperature observed in the PCO camera was −13 °C, well above the camera's minimum operating temperature. Furthermore, the temperature of the PCO camera decreased at a much smaller rate than the environmental temperature.

However, the maximum temperature range appeared to be an area of larger concern. After sunrise, the PCO camera temperature began to increase at a much larger rate than the environment. At the time that the payload was powered off, the environmental temperature was −30 °C, while the hottest temperature observed in the PCO camera was 55 °C. It appears that this temperature would have kept increasing, given the positive rate of change of the temperature when the payload was shut off. This temperature may have passed the 60-degree Celsius maximum operating temperature, which may have caused damage to the components.

From the temperature data, it can be determined that passive heating (the thermal blanket and heat generated by the PCO camera itself) was enough to keep the PCO camera well above its minimum operational temperature during the mission. Given that the components survived with no thermal damage, it can be determined that powering off the payload shortly after sunrise and

thus removing the heat generated by the PCO camera, was enough to keep the PCO camera below its maximum operating temperature during the mission.

Given that the temperature sensors were integrated into the camera electronics, they stopped recording the temperature after the camera was shut off. In the future, we will make use of external temperature sensors coupled to a separately powered circuit, such that the sensors can continue to record the temperature even after the camera is shut off. The environmental temperature data provided by the CSA can be used as a temperature profile for thermal testing on the ground, which can provide a way to validate the system for future work.

2.7. Conclusions

The risk of in-orbit collisions is an ever-growing threat, and the need for SSA continues to grow with it. Optical imaging remains a viable method of detecting RSOs, but ground-based telescopes remain expensive, have large integration times, and limited FOVs.

The RSONAR mission successfully demonstrated the feasibility of using a star-tracker-like, wide FOV camera for imaging RSOs. Over 93,000 images were captured, with preliminary analysis showing over 500 RSOs detected within them. The payload was developed entirely using COTS components. These factors present a cheaper, sub-orbital, rapid imaging alternative to typical ground-based solutions for increasing SSA, demonstrating key technologies to extend the payload to a space-based mission.

Though the cold temperatures were thought to be the biggest concern, the heat generated by the COTS components, as well as the thermal insulation from the aluminum polyamide covering, helped the payload maintain temperatures well within operating specifications. The upper-

temperature range was the bigger concern, with device temperatures nearly reaching the maximum operating temperatures before the payload was issued a shutdown signal from the ground.

Given the promising preliminary results, we plan to advance the RSONAR payload for a space mission. However, further changes need to be made to overcome some of the issues faced, and to ensure the design is ready for the challenges of a low earth orbit environment. We aim to use a radiation-hardened version of the Zynq-7000 SoC, combined with space-grade sensors and custom pins to ensure the hardware is ready for the space environment.

For future work, we will demonstrate increased onboard image processing to detect RSOs in real-time. This involves keeping only images containing RSOs to reduce the amount of data stored and downlinked, as well as the time spent processing images for RSOs on the ground. Furthermore, real-time image processing for RSO detections in future stratospheric missions will serve as a technology demonstration for real-time RSO detection in space-based missions. This is highly desirable because it is the first step in enabling autonomous, real-time action, such as RSO tracking, changing camera parameters for better RSO imaging, and performing satellite maneuvers, without needing a human in the loop. The FPGA environment will be better leveraged to accomplish this by offloading processing-heavy tasks to the FPGA fabric, and sequential tasks and sensor-interfacing to the integrated ARM processor.

3. Resident Space Object (RSO) Tracking in Space-Based, Low Resolution, Non-Constant-Attitude Imagery

This chapter is based on a paper currently under review describing an RSO tracking algorithm designed to track RSOs within space-based, low resolution, non-constant-attitude imagery. This imagery was acquired from the FAI instrument, an optical instrument onboard the CASSIOPE satellite. The availability of this imagery provided a unique opportunity to use imagery from existing space-based payloads to improve SSA. However, many challenges were posed by these images, including their low resolution, high noise, presence of artifacts such as Earth's limb, and the non-constant attitude of the host satellite, causing both the background stars and RSOs to move. These challenges prompted the development of a novel algorithm, combining multiple image processing techniques to reliably detect RSOs. The algorithm consists of a custom thresholding scheme to separate the background from the objects within the images, followed by a simultaneous pose and correspondence algorithm to identify the rotation and translation of the background stars between images, simultaneously correcting the motion and removing the stars from the images. Lastly, a tracking algorithm based on a linear motion model was used to track the RSOs within the images. The algorithm achieved 79% precision and 71% recall on a hand-labelled 878-image dataset. Within these images 87% of unique RSOs were detected at least once, as they passed through the imager's FOV. Along with these results, additional research involving the brightness and centroiding accuracy of the objects was conducted. These results proved that the developed algorithm is a promising method for detecting RSOs within imagery

from payloads onboard existing satellites, even in cases with reduced attitude control, during slew maneuvers, or otherwise non-inertial pointing scenarios.

I developed the RSO tracking algorithm outlined within this chapter. I developed the 878-image dataset, selecting image sequences based on a variety of criteria, and using CVAT.ai to label the RSOs within them. I prepared all figures, and tables, and wrote the manuscript. Paul Harrison and Matthew Driedger gave me helpful tips during the development of the algorithm. Randa Qashoa provided me with her expertise on the image processing pipeline for SSA. Gabriel Chianelli gave me feedback on the studies carried out to examine the centroiding accuracy of the system. Vithurshan Suthakar provided me with his insight on the study carried out to analyze the brightness of the objects detected by the system. Regina Lee provided me with her expertise and direction in carrying out this research. All coauthors provided feedback and revisions for the manuscript.

3.1. Introduction

The near-Earth space environment is experiencing a large and sudden increase in the number of RSOs, which can be attributed to the increase in space-based activities. Satellite launches, anti-satellite missile testing, and even in-orbit collisions contribute to the RSO population. The Iridium 33-Cosmos 2251 collision in 2009 [50] and the Fengyun-1C destruction are just a few examples of such activities contributing to the generation of space debris [51]. The space community continues to work on mitigating the risks associated with space debris by improving collision avoidance strategies, promoting responsible satellite disposal practices, and increasing international cooperation to address this growing problem [52]. In order to implement these

strategies, operators need detailed knowledge of the space environment to be able to task satellites with collision avoidance maneuvers. Knowledge such as object velocity, orbital parameters, and object identity are all necessary. The collection of this knowledge is an activity referred to as SSA. A first step in achieving SSA can be through the use of optical imagery, from which space objects can be detected and then identified [53].

Traditionally, SSA using optical imagery is conducted on the ground, using powerful, long-exposure, narrow FOV telescopes. An example of such a telescope was NASA's MODEST telescope, which used a 1.3-degree by 1.3-degree FOV to capture images of orbital debris in GEO using a five second exposure time [25]. These ground-based telescopes are used to track the debris, following their motion through the sky, meaning that the RSOs appear as point sources in the image. However, in LEO, complications arise in using a similar system to track RSOs [28,29]. Instead, RSOs are usually detected from imagers operating in stare mode, where RSOs pass through the FOV of the imager. These RSOs must then be tracked within sequences of images using algorithms. In this case, tracking refers to the process of identifying unique RSOs and maintaining their identities throughout images. A wide FOV system is preferred in this case to maximize RSO quantity in the FOV. Many streak detection methods have been used to identify star or RSO streaks in long-exposure astronomical images and can be categorized into several classes [54]. These classes include simple methods such as source detection, computer vision methods such as derivative-based edge detection, Point Spread Function (PSF) template matching algorithms, and even machine learning algorithms [55,56,57,58].

However, short exposure imagery (as in the data considered in this research), captured in the order of 10 to 100 milliseconds, requires the use of more sophisticated detection methods. In this

type of imagery, stars and RSOs will both appear as point sources of light in images, making the task of differentiating RSOs from stars more complicated. Instead, information from multiple sequential images across time can be used. Furthermore, tracking must be performed to uniquely detect RSOs. On the ground, stars will remain roughly stationary for short periods of time, while RSOs will move through the FOV. Simple source detection can be used in this case, but a different shape requirement must be used to detect RSOs. However, in space-based imagery, the stationary star assumption may not be feasible, especially in cases with non-inertial pointing spacecraft. In this imagery, the background stars move in addition to the RSOs, making the RSO detection and tracking problem even more challenging. Given space-based hardware constraints and the space environment, this imagery may be of lower quality than what powerful ground-based systems can provide as well.

Nevertheless, multiple methods and pipelines have been developed to process short-exposure astronomical imagery. Frame differencing has been used to detect RSOs in imagery where stars do not move significantly in short time frames [59]. Consequently, this algorithm does not perform as well when the imager is moving relative to the stars, which causes stars to move in the images. Another commonly used method is image stacking, a technique which involves combinations of subsequent astronomical images. This method has been successfully used to both increase the signal-to-noise ratio of detections, as well as detect geostationary satellites [60]. This method works well to detect these satellites because stars will tend to streak after stacking, while geostationary satellites will remain in the same location. However, this method may not be sufficient in detecting satellites that appear to be moving in images, such as ground-based observations of LEO satellites or satellites as observed from a moving, space-based

platform. Machine learning methods have also been used to detect RSOs. In [49], a CNN is first used to classify images as containing detections (of both stars and RSOs), and then classified as stars, RSOs or noise during the tracking process. A graph-based tracking method is used to match objects between frames, which uses an objective function consisting of the minimum of a set of four cost functions. The tracks determined by this algorithm are then classified into stars, RSOs, and noise.

While most SSA images rely on narrow FOV, high resolution images from ground-based telescopes, [61] outlines several benefits to wide FOV imagers like star trackers. These advantages include their low cost, ability to image more of the sky per frame, faster integration time, and lack of atmospheric restrictions over their expensive, ground-limited counterparts.

On the other hand, detecting RSOs in low-resolution imagery imposes a great deal of challenges. While most object detection algorithms are designed for high resolution images where features such as lines, corners, and facets are visible in the images, these methods are not applicable in low-resolution images where a star or an RSO is represented with a very small number of lit pixels. Furthermore, these low-resolution images contain more noise, making real objects harder to distinguish from the noise.

In this chapter we outline a method for RSO detection and tracking in low resolution, non-constant-attitude imagery. In Section 3.2, the full SSA pipeline is discussed. In Section 3.3, the dataset used for this research will be outlined. In Section 3.4, the algorithm itself will be described. In Section 3.5, the metrics used to evaluate the algorithm's performance will be introduced, followed by the results. In Section 3.6, the results are discussed, and advantages and

limitations of the algorithm are described. In Section 3.7, improvements that are planned and currently being performed are discussed.

3.2. Space Situational Awareness Pipeline

While the focus of this research is to present and evaluate an RSO tracking algorithm, it is important to discuss the complete processing pipeline in detecting RSOs from unresolved optical imagery for SSA. Firstly, as outlined in this research, RSOs need to be detected within a sequence of images, producing detections in pixel coordinates relative to the imager's frame of reference. Then, coordinate transformations need to be considered to transform the RSO detections in the images to an inertial frame of reference, such as the celestial coordinate system, alongside the accuracy of this transformation. After doing this, the detected RSOs need to be identified. This can be done by performing IOD to get an estimate of orbital parameters, from which a matching algorithm can be used to correlate the detection to the positions of satellites as reported by their corresponding ephemeris data or TLE data. It is important to note that this pipeline does not consider photometric information from RSO detections, such as their brightness. However, such information can be used to enhance RSO identification, or to gain further knowledge about the RSO, such as its attitude.

3.2.1. RSO Detection and Tracking

The SSA pipeline begins by detecting RSOs within a sequence of images. As highlighted in Section 3.1, various algorithms are used for this purpose, depending on characteristics such as the imager's attitude, the exposure time, the resolution, the Signal-to-Noise Ratio (SNR), and artifacts in the images. The ideal RSO detection algorithm will be able to track RSOs across

images, maintaining unique identities for each RSO over time. The algorithm's output will produce pixel coordinates of RSO detections.

3.2.2. Celestial Coordinate System Transformation

Each RSO detection made by the algorithm needs to be converted from image coordinates to coordinates in an inertial frame of reference, such as the celestial coordinate system. This coordinate system consists of the RA and Dec of a celestial object, which is based on an inertial frame centered at the center of the Earth. To do this, an image first needs to be plate solved, a process that identifies the stars in an image using a known star catalogue and returns the astrometric calibration of that image, consisting of information such as the RA/Dec of the center of the image, the orientation of the image with respect to the celestial coordinate system, and the pixel scale [62]. Since the FAI instrument's image plane rotates and translates over time (due to the host spacecraft's non-constant attitude), the orientation of the images will change over time as well. However, as will be mentioned below, this movement can be negated using the RSO detection algorithm in this research. This reduces the frequency at which the images need to be plate solved.

With the center pixel RA/Dec, orientation of the imager, and pixel scale known, each RSO detection can be converted to RA/Dec coordinates by first orienting the image such that the positive y-axis aligns with North (the direction of increasing Dec). Then, the location of each RSO in the image can be calculated as pixel offsets in the x-axis and y-axis independently, from the center pixel location. These pixel offsets can then be converted to arcsecond offsets by using the pixel scale. Finally, these offsets can then be added to the center pixel's RA/Dec, following

the convention of increasing Dec in the positive y-direction, and increasing RA in the positive x-direction, to get each RSO's RA/Dec. These transformations are often captured in a World Coordinate System (WCS) file returned by tools such as [62] but need to be done manually as outlined above if using a custom AD algorithm. Figure 3.1 visually depicts this transformation.

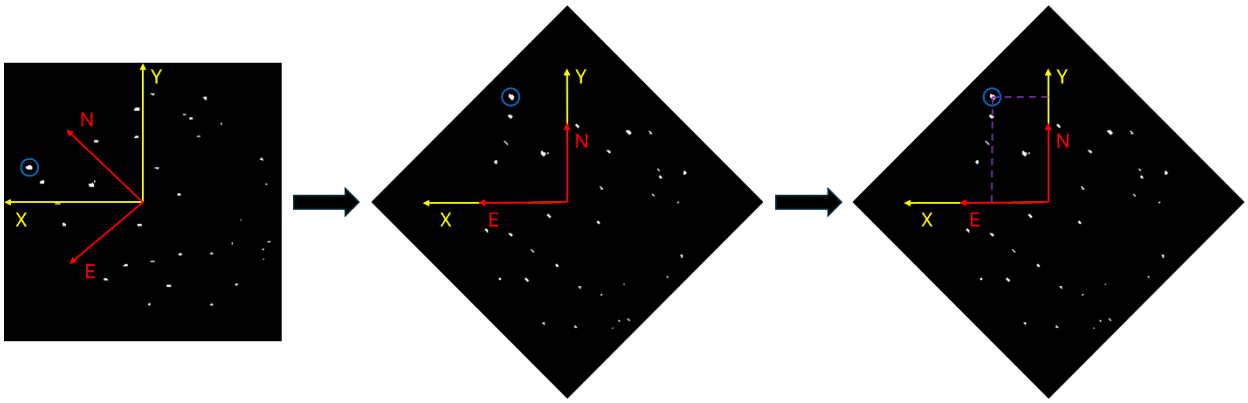

Figure 3.1. Illustration depicting the transformation performed to align the image axes (yellow) to the celestial coordinate system axes (red), to then determine the x and y pixel offsets (purple) to a desired RSO (blue) to determine the RSO's RA/Dec.

3.2.3. Initial Orbit Determination

To accurately identify the RSOs in the images, further information is needed about the RSO, such as its orbital parameters. These parameters can be extracted using an angles-only IOD algorithm such as the Gauss or Laplace method [63], which seeks to use multiple observations of the same RSO (in RA/Dec) to find its orbital parameters. Angles-only IOD algorithms are useful in that information such as the range to the detected RSO is not required. However, the host spacecraft's positional information needs to be known as well.

3.2.4. RSO Identification

With the RSO's orbital parameters known, the RSO can be identified. This can be done by correlating the detected RSOs to known RSO data. For this, the time at the instant of imaging needs to be known, along with the corresponding RSO detections to be matched. Existing RSOs need to then be propagated to this time by using existing information about them. Such RSO information can be in the form of TLE data, ephemeris data, or some other system, coming from databases such as Space Track and CelesTrak, or from internal databases generated from high-confidence observations. A matching algorithm can consider the orbital parameters acquired by the steps above, alongside metrics such as the closest RSO, and use them to find the best matching RSO. RSOs detected in the FOV with poor matches may be uncatalogued RSOs, for which repeated measurements should be conducted and orbital estimations refined to confirm this hypothesis. Figure 3.2 graphically depicts this SSA pipeline.

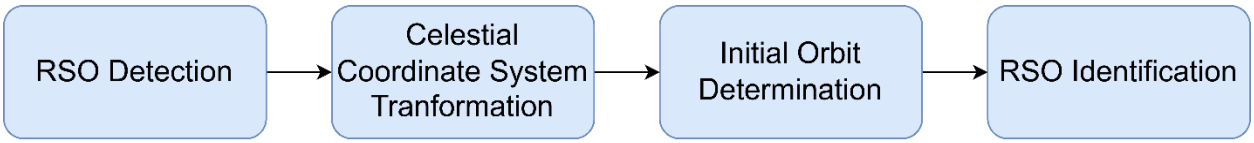

Figure 3.2. Block diagram outlining the steps in the SSA pipeline for identifying RSOs from unresolved optical imagery.

3.3. Dataset Used in the Study

The dataset used for this research consists of low resolution, short exposure imagery captured by the FAI, an instrument onboard the CASSIOPE satellite [64]. Though developed and routinely tasked to image the aurora, RSOs appear in the FOV of the imager, allowing the use of the FAI imagery for SSA purposes. Furthermore, this instrument has comparable optical properties to a conventional star tracker, as described below and in Table 3.1.

Table 3.1. FAI instrument properties.

Instrument Property	**Property Value**
FOV	26º full angle
Focal Length	68.9 mm
Focal Length with Reducer	13.78 mm
f-number	4.0
Pixel Size	26 µm by 26 µm
Pixel Scale	388 arcsec/pixel
Resolution	256 by 256 pixels
Exposure Time	100 ms

The images have a resolution of 256 by 256 pixels, and an exposure time of 100 milliseconds. Given the low resolution, short exposure time, and low angular rate of the host spacecraft, both stars and RSOs appear as point sources of light with limited pixels per source. As of December 17, 2021, the CASSIOPE spacecraft is spin-stabilized using torque rods, given the failure of its momentum wheels [65]. This caused a variety of challenges that needed to be accounted for during image processing. For example, the stars and RSOs both moved through image sequences, which meant that traditional algorithms such as frame differencing could not be used to identify RSOs, since these algorithms rely on a static background to identify RSOs. Additionally, spin stabilization caused RSOs in the FOV to follow a curved trajectory. This meant that a simple linear motion model could not be applied directly to identify RSOs. Furthermore, the changing attitude of the imager caused the illumination conditions of each

image to vary spatially, in addition to the temporal illumination change. This variable background illumination, both within a single image and across multiple images, prevented the use of simple thresholding techniques to process the images. Using a single threshold value would result in both over and under-thresholded areas within a given image while inter-image variability would cause similar over and under-thresholding throughout image sequences. An algorithm that can consistently detect and track RSOs within these images needs to account for all these challenges. Figure 3.3 shows a variety of images captured by the FAI instrument and demonstrates some of the challenges in processing these images. More details on the FAI images can be found in [20].

3.4. Tracking Algorithm Overview

An algorithm was developed to detect and track RSOs within FAI imagery. Several assumptions were made to simplify the algorithm's design. Firstly, it was assumed that the dominant point sources of light within the images are stars. Secondly, it was assumed that the background stars appeared to move rigidly within the imager's FOV throughout subsequent images. These two assumptions facilitated the use of the Iterative Closest Point (ICP) method to find and correct the effect of the host spacecraft's attitude on the images. Thirdly, it was assumed that RSOs travel approximately linearly through images, after accounting for the spacecraft's attitude with respect to the background stars. Finally, it was assumed that RSOs travel the same distance between each image. These two assumptions were used to design a method to distinguish RSOs from the remaining objects after processing the images. The algorithm was developed in Python, using OpenCV, scikit-learn, and scikit-image among other common Python libraries.

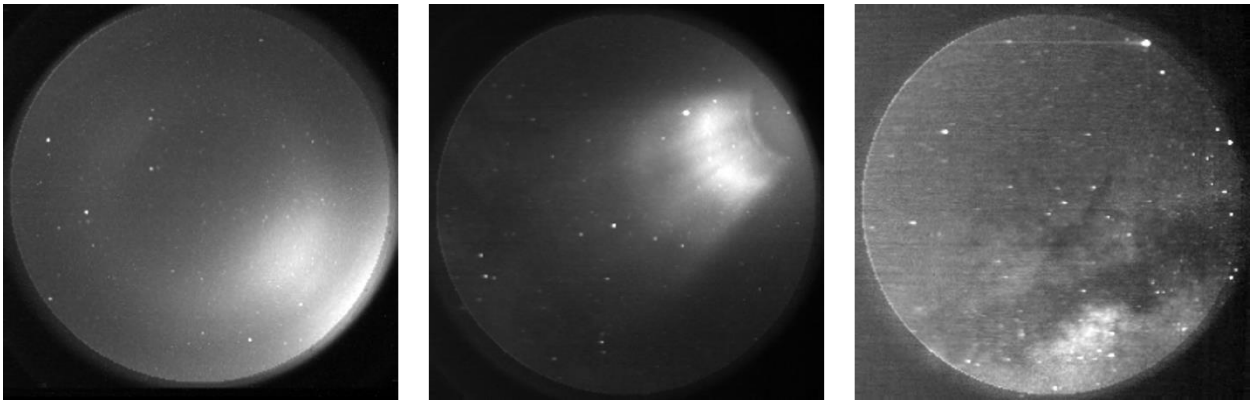

Figure 3.3. Several examples of astronomical images taken by FAI. Stars and RSOs appear as point sources of light with limited features. The illumination conditions vary significantly in the images, posing a challenge for simple thresholding methods.

The algorithm takes in three images at a time, in a sliding window fashion. In other words, the algorithm maintains three images in memory, replacing the last image with a new image, rolling the window forward. The algorithm also takes in certain parameters as inputs, such as the threshold value to use. As outputs, the algorithm produces text files containing the locations, pixel sizes, and other telemetry regarding the detected RSOs. Optionally, the algorithm can also generate processed output images, consisting of the RSOs detected within the current frame as well as the predicted locations of previously detected RSOs which have disappeared from the current frame.

The algorithm consists of five main steps as outlined in Figure 3.4, namely (a) preprocessing, (b) ICP star removal, (c) three-frame RSO association and tracking, (d) position estimation, and (e) RSO status and archival.

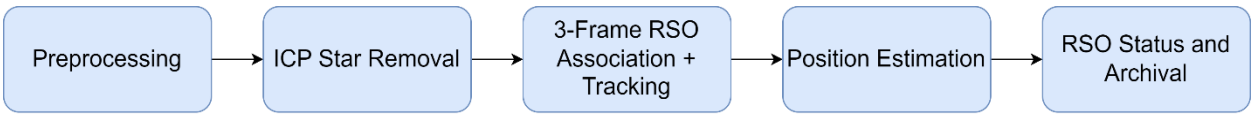

Figure 3.4. Block diagram outlining RSO tracking algorithm steps.

3.4.1. Preprocessing

The preprocessing step consists of several processing techniques to generate a list of data such as centroid coordinates corresponding to objects (stars, RSOs, and noise) from each raw image.

3.4.1.1. Image Reading and Cropping

Firstly, three images are read into memory. Given that the algorithm uses a sliding window of three images at a time, the preprocessing must be performed on all three images for just the first iteration. Later iterations only need to read in and perform preprocessing on the third (new) image. The images contain additional information in the first 72 and last 12 rows of each image, which are cropped from the images before processing begins.

3.4.1.2. Circular Region of Interest (ROI) Extraction

Next, the remaining pixels outside of the circular FOV of the imager are set to zero to prevent false detections due to noise, and to prevent these areas from interfering with further processing. This is done by keeping all the pixels in the image within a radius of 120 pixels from the center of the image. This radius value is left as a parameter that can be adjusted.

3.4.1.3. Windowed Multi-Otsu Thresholding

After this, thresholding is performed to binarize the images, leaving the background as the minimum value and the stars and RSOs as the maximum value. While well-established methods

such as SExtractor exist for extracting sources from astronomical images, a custom algorithm was developed for simplicity and specifically for thresholding images for SSA, where steps such as background map construction, faint galaxy deblending, and star-galaxy separation are unnecessary [66]. Given the varying spatial illumination in the image, a local thresholding method is used to threshold each image in sections or windows. For the images used in this research, a user-defined window size of 32 by 32 pixels was used to threshold each image, window by window, which produced acceptable thresholding results. In cases where there is significant spatial inhomogeneity, it is recommended to decrease the window size further, and vice versa.

To account for the varying temporal illumination, histogram-based thresholding was used. Specifically, Otsu's method was adapted, which seeks to find a threshold value which separates pixel values in an image into background and foreground classes [67]. The idea behind this algorithm is that most pixels in an image will belong to one of those specific classes, forming two peaks in the pixel histogram of the image. In the case of these images, those two classes would be the background and the star and RSO signals. A threshold that best separates these two peaks could then be found, determined for each image, to dynamically produce an optimal threshold for each image. However, analysis of the histograms of FAI images revealed that there existed a third class of pixels within the images, with average values greater than the background but smaller than the star and RSO signals. This class appeared to belong to the illumination effects, such as lens flare. For this reason, multi-Otsu thresholding was used instead, which returned a threshold that separated the first two classes from the third class, allowing the stars and RSOs to pass the threshold without any background effects. However, an experimentally

determined constant needed to be added on top of the returned threshold value, likely due to overlap between the pixel values of illumination effects and star and RSO signals. A user-defined constant of 35 was used, found by manually adding to the returned threshold on a subset of images until a desirable threshold was produced. This constant is recommended to be increased or decreased depending on the results of the threshold returned by the multi-Otsu thresholding algorithm. This threshold is calculated and applied to individual segments of each image, as mentioned previously. An example result of this windowed multi-Otsu thresholding is shown in Figure 3.5, for one image.

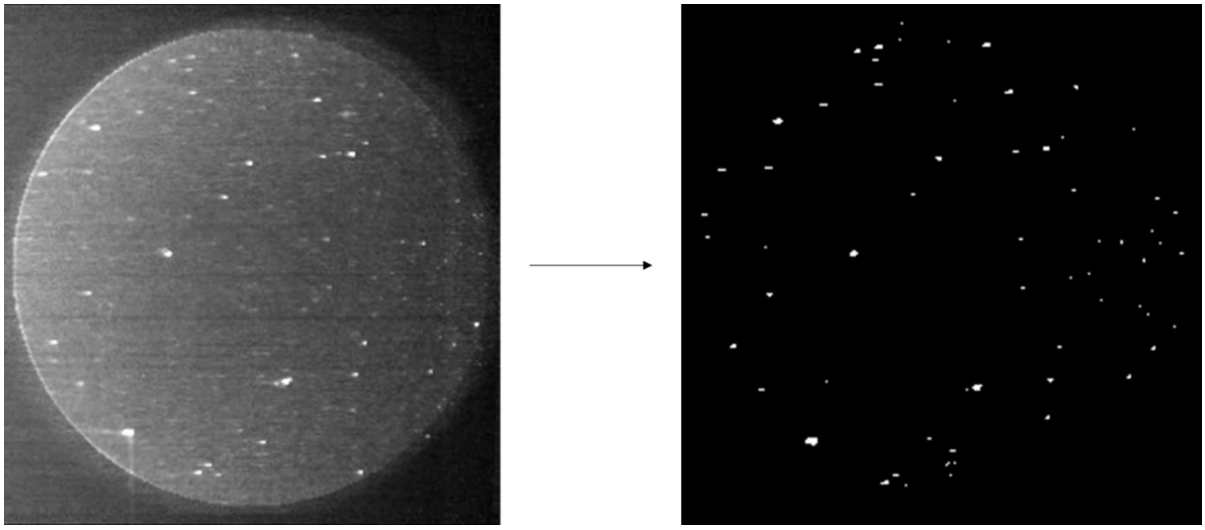

Figure 3.5. The original image (left) and the image after removing pixels out of the imager's FOV and thresholding (right).

3.4.1.4. Object Detection

Next, Connected Component Analysis (CCA) is performed on each binary image to individually segment and label groups of connected pixels. This offers a way to separate point sources within the image into distinct objects [68]. For each of these objects, in each image, properties are

extracted such as the total pixel area, height, width, and centroid coordinates of the bounding box around the object. The centroid coordinates of the objects are especially useful in further processing steps.

3.4.1.5. Object Filtration

Finally, the preprocessing step concludes by filtering the objects found in the previous step by imposing maximum and minimum limits on each object's properties. Firstly, a minimum pixel area is imposed to remove small objects from the images, which are likely a result of noise such as hot pixels and faint stars. In this research, objects corresponding to pixel areas smaller than two pixels are discarded. Next, a maximum pixel area is imposed to remove large objects from the images, which are likely due to surviving illumination effects such as lens flare. Objects greater than 81 pixels in area are discarded in this research. After this, maximum width and height thresholds were imposed to remove streak-like artifacts from the images, usually caused by light streaking from bright stars and RSOs, as well as radiation strikes. A value of 15 pixels is used as the maximum threshold for an object's height and width. After this filtration step, the data is ready to be processed for star removal and RSO detection.

3.4.2. ICP Star Removal

The next step involves removing the points corresponding to stars from the lists corresponding to the three images by using the ICP algorithm, which seeks to find a rigid transformation that fits a set of source points to a set of reference points, as well as the point correspondences [69,70]. This is done by iteratively matching the point sets and minimizing an error metric, in this case the Euclidean distances between the point matches, until a threshold is reached. The method

proves useful in that it does not require an initial estimation of the transformation or the point correspondences. Though commonly used for lidar applications to reconstruct 3D surfaces from different scans, a 2D implementation was used for this RSO detection algorithm [71]. As mentioned previously, it is assumed that the stars between images appear to transform rigidly This is because stars appear fixed due to their large distances away from near-Earth observers, and only appear to move between images due to the host spacecraft's own attitude. The other relevant assumption here is that there are significantly more stars than RSOs in the images. Since the algorithm returns the point correspondences, and the stars are often the dominant object in the images, the ICP algorithm tends to match the star transformation between frames. Thus, the returned list of matching points tends to be the star matches between the images. With three point sets (corresponding to the three images), the idea is to keep the second image's point set as the reference point set, and then match the point sets from the other two images to the second image point set. For this, ICP is applied twice, once to match the points from the first image to the second image, and then the third image to the second image. The two operations return the points that were matched (assumed to be stars) and all points from the first image and third image transformed by the identified rigid transformation.

After having identified the stars in the images, they can be removed from all the images. The matched points from the first image and second image are searched for within the original points from the first image and second image and removed. The same operation is done between the third image and the second image. Figure 3.6 shows an example of the points that are left over from all the images after this, stacked onto a single image. A star identification method for star removal would be less restricted and would return information about the RA/Dec, necessary for

orbit determination. However, the algorithm is intended to be independent of AD (which performs star identification) and using ICP for star removal has the additional benefit of removing the spacecraft's rotation from the RSOs, making them easier to detect with the approximate linear motion model. Ideally, after this step, all that remains are RSOs, since RSOs have motion different from the stars within the images. However, there are leftover stars that did not get matched in the ICP algorithm and noise. Furthermore, the points corresponding to RSOs have not been matched between the three images, since the ICP algorithm only returns the star matches. Lastly, the RSOs have not been checked for matches in previous images. The next step seeks to address these concerns.

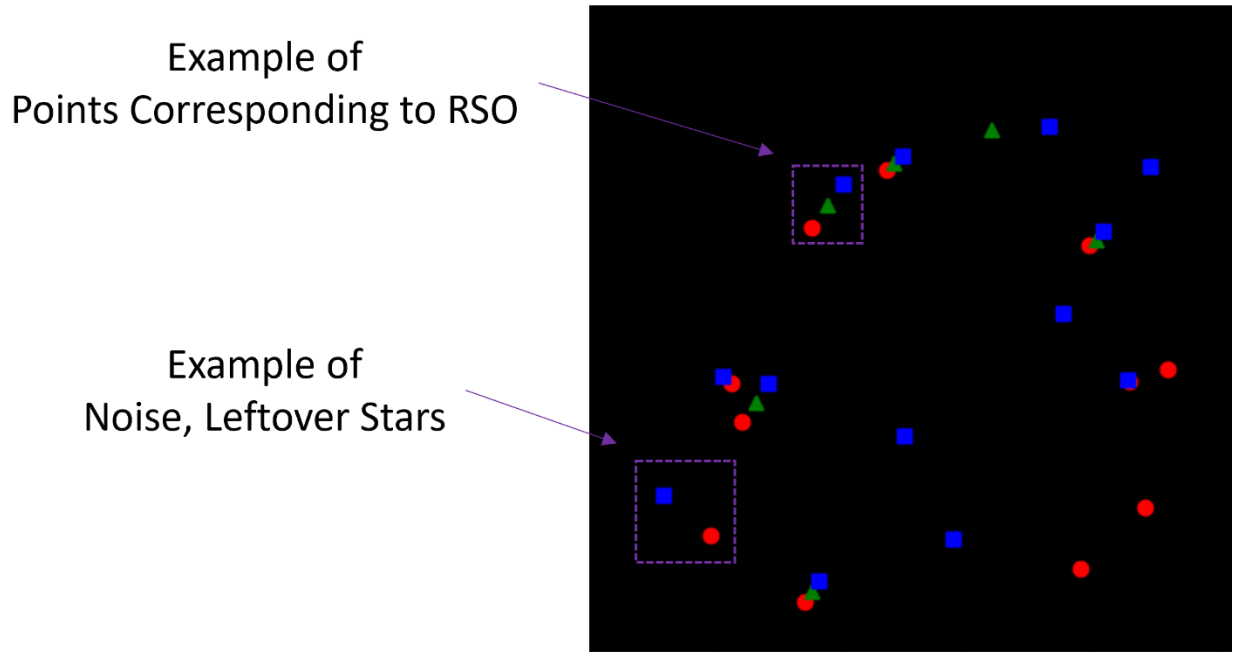

Figure 3.6. The points left over after removing the stars from the three images, plotted against a single background. Red (circle) corresponds to the first image, green (triangle) corresponds to the second image, and blue (square) corresponds to the third image in the rolling window.

3.4.3. 3-Frame RSO Association and Tracking

The next step uses an algorithm to determine which of the remaining points correspond to RSOs. This is done by using the assumption that RSOs will travel the same distance between frames, and that their motion will be roughly linear and in one direction. A nested for loop is used to iterate through unique combinations of points, where each point must correspond to a different image, and not already be matched to an RSO. The first step in the loop is to calculate the distance, d_1, between the first point (first image) and the second point (second image), as well as the distance d_2 between the second point and the third point (third image). Next, after ensuring that neither distance is close to zero (to avoid divide-by-zero errors), the similarity of the distances, $d_{similarity}$, is calculated using Equation 3.1. If this similarity is above an experimentally determined threshold (0.5 is used), the next check is performed, which determines how linear the motion is between the points, and if they are in the same direction. This is done by determining the vectors d_1 and d_2, and calculating the angle between them, using Equation 3.2.

$$d_{similarity} = 1 - \left| \frac{d_1 - d_2}{\max(d_1, d_2)} \right| \quad (3.1)$$

$$\theta = \cos^{-1}\left(\frac{\overrightarrow{d_1} \cdot \overrightarrow{d_2}}{|\overrightarrow{d_1}||\overrightarrow{d_2}|} \right) \quad (3.2)$$

To ensure that the motion of an RSO is roughly linear, the calculated angle must be limited to a maximum value. It was found that some RSOs with large distance similarities may still have large angles between their corresponding vectors. Similarly, some RSOs may have small distance similarities but also have small angles. An appropriate maximum angle, θ_{max}, was determined for all distance similarities by plotting the distance similarities and their corresponding angles for

both the remaining points and RSOs, as illustrated in Figure 3.7. The line that separated the RSOs from the remaining points was then identified, as shown in Equation 3.3, and used to calculate θ_{max} for a given $d_{similarity}$ to classify a set of three points as an RSO or not.

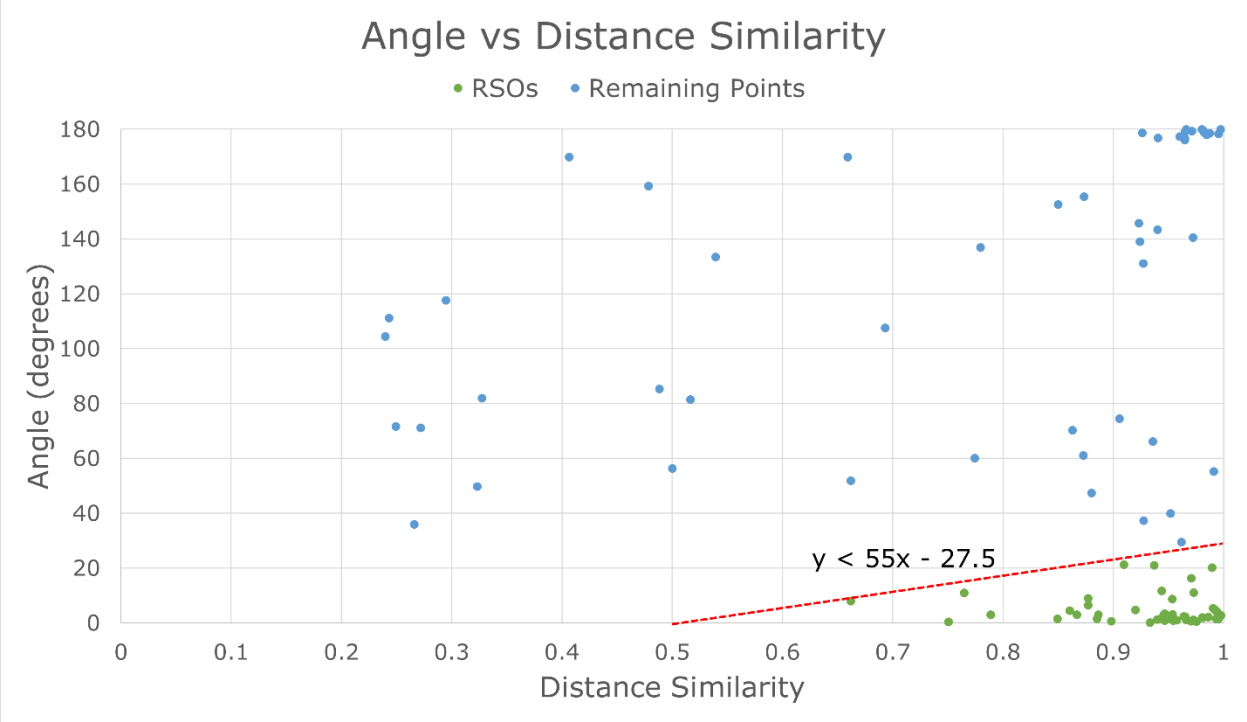

Figure 3.7. Plot of angles against distance similarities of the remaining points and RSOs at this step. A line can be found that separates each class.

$$\theta_{max} = 55d_{similarity} - 27.5 \quad (3.3)$$

Once these conditions have been met, the set of three points are associated to a single RSO detection and are removed from the list of points to loop through. The next step involves associating the detected RSO to previous detections. Since a sliding window is used, the current iteration's points in the second image will correspond to the previous iteration's points in the third image. The same can be said about the first image and second image, respectively. Using

this information, the current RSO detection's points can be compared to previous RSO detections' points to see if there is a match. If there is a match, the RSO detection is associated with it. Otherwise, a new label is assigned to the detection. Figure 3.8 illustrates the results of the 3-frame RSO association algorithm, showing how the RSOs are picked out from the remaining points.

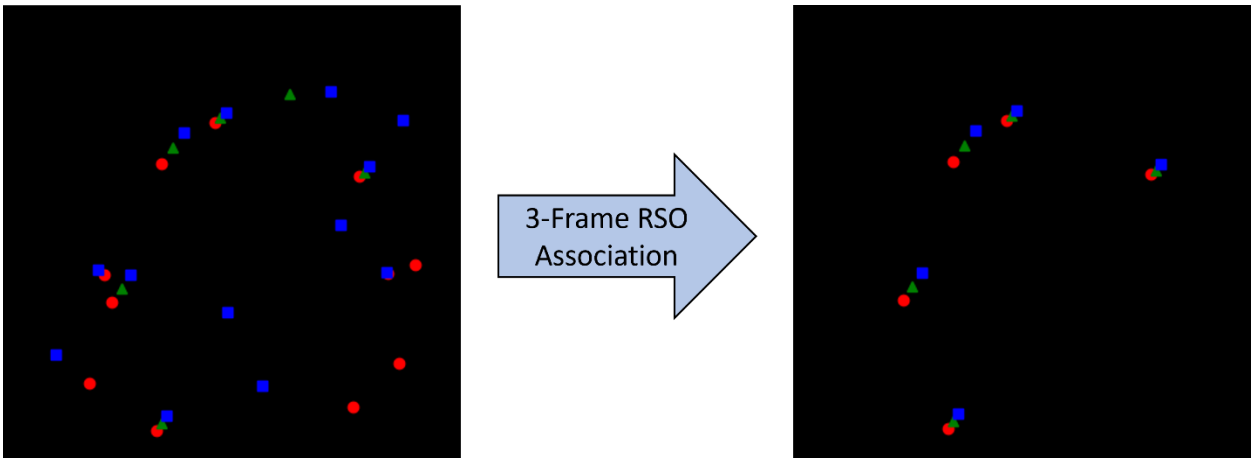

Figure 3.8. The 3-frame RSO association algorithm selects the RSOs (right) from the remaining points (left). Red (circle) corresponds to the first image, green (triangle) corresponds to the second image, and blue (square) corresponds to the third image in the rolling window.

3.4.4. Position Estimation

The second to last step (d) is used to account for dropped RSO detections from previous frames by estimating their positions using previous data. This step starts with looping through all the RSOs from the last iteration that were not detected in the current iteration. If the position of that RSO was not already estimated in the previous iteration (to prevent multiple position estimations of false positive detections), its position is estimated in the current iteration. This is done by calculating the x and y distance travelled by the RSO between the two previous frames and

adding it to the x and y coordinates of that RSO in the previous frame. Additional checks are added as well, such as ensuring that the state that is estimated is within the FOV of the imager.

3.4.5. RSO Status and Archival

The last step (e) involves the generation of text files to report on the status of RSOs, and to archive dropped detections. RSOs that are detected in the current iteration are reported, including information such as their x and y pixel location, pixel size, and a Boolean to indicate whether the RSO had been estimated in the current iteration. RSOs that were in the previous iteration, but are no longer detected in the current iteration, have their information archived in a separate text file. Optionally, figures are generated to provide a visual understanding of the RSO detections. Figure 3.9 is an example of RSOs detected in three frames, represented by bounding boxes with unique identification numbers.

3.5. Results

3.5.1. RSO Detection

For the current study, we used a total of 878 images from the FAI instrument, collected over the current year, 2023. RSOs in each image were annotated manually and 2191 RSO detections, corresponding to 75 unique RSOs, were identified during the annotation, in total. As noted, CASSIOPE is currently operated in spin-stabilization, adding complexity not only to the RSO detection algorithm development, but to the annotation process as well. Slow moving RSOs could be confused with stars, given that the stars were also moving in the images. Furthermore,

RSOs were harder to visually detect due to the varying illumination in the images, making it easy to lose track of an RSO or identify it in the first place.

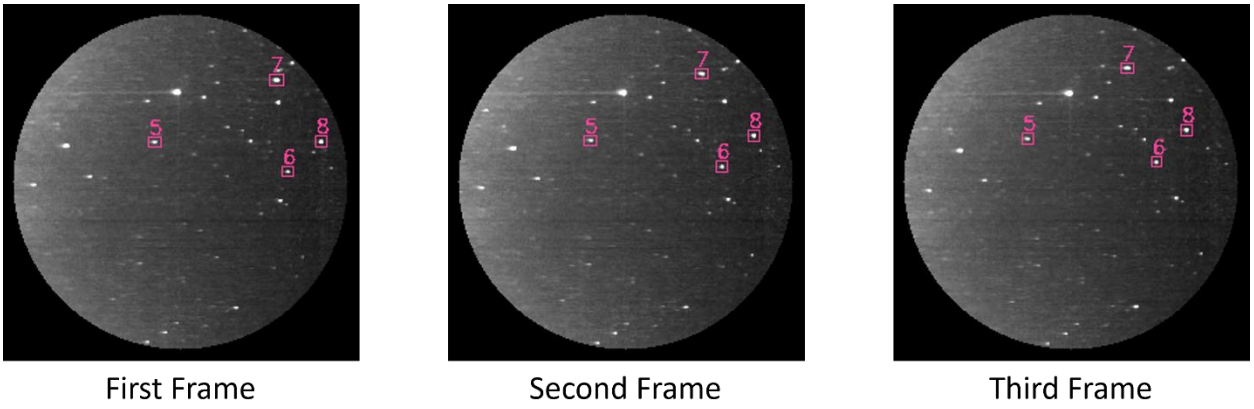

Figure 3.9. Example of figures generated by RSO detection algorithm. Detected RSOs are captured in bounding boxes and given a unique identification number.

To quantify the performance of the developed algorithm, the precision and recall are used, calculated using Equation 3.4 and Equation 3.5 respectively. These metrics use the True Positives (TPs), False Positives (FPs), and False Negatives (FNs).

$$Precision = \frac{TP}{TP + FP} \quad (3.4)$$

$$Recall = \frac{TP}{TP + FN} \quad (3.5)$$

There are several caveats with respect to the results which must be considered. Firstly, RSO detections were determined visually from the images, and therefore the annotations may be inaccurate, especially in challenging images. Furthermore, RSOs at the edge of the circular FOV are ignored, given that the algorithm reduces the FOV to avoid lighting issues at the perimeter of the FOV. Additionally, image sequences with drastic lighting changes were ignored. Finally, the

True Negatives (TNs) were ignored, given the difficulty and ambiguity in calculating this quantity.

Given that the algorithm uses three sequential frames to detect RSOs, the number of RSOs reported at each iteration may be different than the number of RSOs reported in total, after viewing an entire sequence. For example, if an RSO were to appear halfway through a sequence, it would take three iterations (corresponding to three images) before that object was classified as an RSO. However, the classification process detects the RSO in the previous two iterations as well. Therefore, to capture this caveat in the results, the "Per-Frame (PF)" results are quantified to show the performance at every iteration, and the "Full-Sequence (FS)" results are quantified to show the performance at the end of an entire sequence of images. Table 3.2 shows these results.

Table 3.2. RSO detection algorithm performance, Per-Frame (PF) and Full-Sequence (FS).

Test Number	**Date**	**Number of Images**	**Precision (PF)**	**Recall (PF)**	**Precision (FS)**	**Recall (FS)**
1	2023-01-16	41	95%	85%	90%	90%
2	2023-01-21	76	88%	87%	77%	93%
3	2023-01-25	53	91%	68%	89%	75%
4	2023-03-31	139	90%	64%	83%	71%
5	2023-05-31	128	96%	64%	92%	72%
6	2023-06-03	91	68%	46%	57%	52%
7	2023-06-20	39	100%	52%	100%	56%
8	2023-07-19	114	92%	81%	86%	91%
9	2023-08-04	24	94%	89%	86%	100%

10	2023-08-04	75	84%	44%	72%	48%
11	2023-08-05	35	90%	81%	84%	90%
12	2023-08-05	63	88%	76%	80%	91%
Total		**878**	**87%**	**63%**	**79%**	**71%**

Another metric to consider when evaluating the performance of the algorithm is to measure how many RSOs were detected at least once during the RSO's crossing of the imager's FOV. To capture these results, the total number of unique RSOs, along with the number of detections and missed detections is presented in Table 3.3.

Table 3.3. RSO detection algorithm performance. RSOs detected at least once are considered as detections.

Test Number	**Date**	**Number of Images**	**Unique RSOs**	**Detected RSOs**	**Missed RSOs**	**Percentage Detected**
1	2023-01-16	41	1	1	0	100%
2	2023-01-21	76	3	3	0	100%
3	2023-01-25	53	5	5	0	100%
4	2023-03-31	139	13	12	1	92%
5	2023-05-31	128	17	13	4	76%
6	2023-06-03	91	11	9	2	82%
7	2023-06-20	39	2	1	1	50%
8	2023-07-19	114	8	8	0	100%

9	2023-08-04	24	2	2	0	100%
10	2023-08-04	75	6	4	2	67%
11	2023-08-05	35	3	3	0	100%
12	2023-08-05	63	4	4	0	100%
Total		**878**	**75**	**65**	**10**	**87%**

3.5.2. Detection Accuracy

To quantify the positional accuracy of the detections, the centroids of objects detected by the algorithm are compared to truth data. While an algorithm as outlined in Section 3.4.3. is unavailable in this study to use as truth data for RSO detections, the background stars can be used instead, given that the algorithm also detects stars as captured by the ICP algorithm in the star removal step. To serve as truth data, images can be plate solved using a tool such as the one outlined in [62]. However, tools such as these may contain their own centroiding errors, but for this study, these are assumed to be negligible. Further studies conducted for the rest of the SSA pipeline will evaluate the impact of centroiding errors on RSO identification. Since these stars are centroided in the same way as the RSOs, and since both objects are similar in appearance (point sources of light), the stars' centroids can be used to determine the algorithm's centroiding accuracy. While the focus of this research is on RSO detection and metrics associated with RSO

detection, centroid accuracy is an important consideration for the rest of the SSA pipeline and to identify improvements needed.

Using [62], two images from 11 of the 12 sequences were plate solved to both identify the stars in the images and their true positions. Sequence 7 was unable to be plate solved, given the challenging lighting conditions (and lack of sources detected). While [62] was unable to plate solve the original images, they were able to plate solve the thresholded images, and so these were used. Given the rapid exposure time relative to the changing attitude, only two images per sequence were used, one from the start and one from the end, since many of the same stars would appear in subsequent images. Nevertheless, there were some duplicate stars, which were included in the results since they are centroided again in subsequent images, hence not counting as duplicated results. Table 3.4 presents the results from this testing, showing the number of stars used in each sequence and the average centroid difference in pixels, from the true pixel location of the stars.

Table 3.4. Centroiding accuracy of the algorithm for the FAI images.

Test Number	**Date**	**Number of Stars**	**Average Centroid Difference in X (Pixels)**	**Average Centroid Difference in Y (Pixels)**
1	2023-01-16	82	0.65	0.75
2	2023-01-21	59	0.45	0.62
3	2023-01-25	50	0.49	0.63
4	2023-03-31	49	0.53	0.50
5	2023-05-31	44	0.58	0.59
6	2023-06-03	57	0.61	0.79
7	2023-06-20	N/A	N/A	N/A
8	2023-07-19	68	0.82	0.81
9	2023-08-04	82	0.91	0.79
10	2023-08-04	65	0.65	0.77
11	2023-08-05	72	0.60	0.65
12	2023-08-05	52	0.80	0.91
Total		**680**	**0.66**	**0.72**

3.5.3. Detection Brightness

Another result examined in this study is the brightness of the detections in the images, quantified by their visual magnitudes in the visible band. RSO truth data would be especially difficult to use

in this case, since the visual magnitude of RSOs can vary as their attitudes change, and the assumptions that need to be made about their shapes when calculating their visual magnitudes. Therefore, the stars are used again to determine the average, minimum, and maximum visual magnitude of the objects detected.

The same two images were used for each sequence as in Section 3.5.2. The tool outlined in [62], used to plate solve the images in the previous section, also returned the RA/Dec of the detected stars, in each of the images. With these values, the corresponding stars' visual magnitudes could be queried, using the tool outlined in [72]. Table 3.5 shows the results from this analysis, depicting the number of stars with visual magnitude information available for each sequence, the magnitude of the brightest star, the magnitude of the faintest star, and the average magnitude of the stars.

Table 3.5. Visual magnitude results of stars detected by algorithm for the FAI images.

Test Number	Date	Number of Stars	Brightest Visual Magnitude	Faintest Visual Magnitude	Average Visual Magnitude
1	2023-01-16	75	2.98	7.14	5.12
2	2023-01-21	57	2.89	6.17	4.90
3	2023-01-25	46	2.89	6.57	4.95
4	2023-03-31	40	0.97	8.64	5.06
5	2023-05-31	43	0.91	7.74	4.01
6	2023-06-03	55	0.91	7.74	4.43
7	2023-06-20	N/A	N/A	N/A	N/A
8	2023-07-19	67	2.07	7.44	5.17
9	2023-08-04	73	2.89	7.14	5.52
10	2023-08-04	66	3.08	7.31	5.13
11	2023-08-05	65	2.89	7.36	5.19
12	2023-08-05	51	2.89	7.23	5.25
Total		**638**	**0.91**	**8.64**	**5.02**

3.6. Discussion

Overall, the algorithm maintains good precision due to its robustness to FPs but struggles with recall due to its high FNs. These FNs can be attributed to several reasons. Firstly, the algorithm

was built on the assumption that RSOs travel roughly linearly as viewed through the imager's FOV. In some cases, RSOs did not travel in this fashion, instead following different motion patterns. This was especially true for faint and slow-moving RSOs. However, the algorithm handled most of these cases using Equation 3.3. Slow moving RSOs generally had high distance similarities, and so larger angles would be permitted by Equation 3.3, allowing the algorithm to capture some slow-moving, curving RSOs. Despite this, there were a few cases of RSOs which curved heavily and had low distance similarities, which were not detected by the algorithm consistently. Another cause of high FNs is the extremely faint RSOs. These RSOs were very hard to distinguish even while annotating the data, given the changing illumination conditions and noise in the images. FPs, though fewer, were also of concern, and were mostly due to the challenging lighting conditions, which caused the thresholding to interpret the lit pixels caused by the lighting as objects. Additionally, FPs were more prevalent in images with more noise, given that these noise objects would not be removed by the ICP star removal algorithm, and may consequently be incorrectly classified as RSOs. FPs were also more numerous in images containing a larger number of stars as well, as the likelihood of these objects erroneously being missed by the ICP star removal algorithm increased, as well as the likelihood of them then being incorrectly classified as RSOs. The algorithm was mostly robust to occlusions, given that the position estimation algorithm estimated the position of RSOs that should have appeared in a particular frame, but were missing, overcoming the effect of occlusion.

Unsurprisingly, the precision is greater in the per-frame performance than in the full-sequence performance, while the recall is greater in the full-sequence performance than in the per-frame performance. This is because as mentioned, the per-frame performance considers each frame at a

time, while the full-sequence performance considers all previous frames. This means that there will be more FPs and TPs reported in the full-sequence result, since it considers the previous two frames that created each detection.

Though the algorithm's recall for the per-frame and full-sequence tests was 63% and 71% respectively, the algorithm captured 87% of all RSOs at least once, through their crossings of the imager's FOV. This indicates that while there remains improvement to be made for detection consistency, the algorithm does quite well in capturing RSOs when considering each RSOs full transit.

While the results presented above depend on the algorithm's performance alone (since the truth data is based on the FAI images), the accuracy and brightness results depend both on the algorithm's performance and the limits imposed by the FAI instrument itself.

In terms of accuracy, the RSO detection algorithm is able to centroid stars in these FAI images within one pixel of their true locations; 0.66 pixels in the x-direction, and 0.72 pixels in the y-direction, which is relatively good given the low resolution of the images and smearing of the detections due to the host imager's attitude. Using the Euclidean distance formula, this corresponds to an overall pixel difference of 0.98 pixels. Additionally, sequences with brighter stars were detected more accurately, which makes sense given that brighter stars appear larger in the images, which can be centroided with more accuracy. The opposite is true as well, given that faint stars can correspond to just one pixel, reducing the accuracy with which these stars can be detected. Given the pixel scale of 388 arcseconds, the centroid accuracy corresponds to 379 arcseconds. This could result in detected RSOs being kilometers away from their true positions. Using algorithms to determine the orbital parameters of the RSO and iteratively improving these

parameters can help match the RSO to known RSOs, effectively identifying the RSO that was detected. Furthermore, more robust centroiding methods can be pursued to improve detection accuracy. This centroiding error can also be reduced by averaging a detection over multiple frames.

In terms of brightness, the algorithm detects stars with an average visual magnitude of about 5.02 within these images and can detect stars as faint as 8.64 in visual magnitude. With current satellite constellations having magnitudes between 4 and 6 [73], the system would be capable of seeing the majority of these RSOs appearing within its FOV, assuming that the algorithm itself is able to detect these RSOs. This presents a significant opportunity for detecting and monitoring a large number of satellites, validating their detections and identifying anomalies.

3.7. Future Work

As mentioned in the previous section, the algorithm's FNs could be improved. With respect to the highly curving RSOs, an improvement can be made by incorporating a more robust RSO motion model, replacing what was experimentally determined and captured in Equation 3.3. We are currently investigating the approach of fitting quadratic functions to established tracks to classify RSOs, while assuming circular motion for stars. With respect to the extremely faint RSOs, both the algorithm and annotation process could be improved to help detect these RSOs by using a more advanced light source detection technique and by visually enhancing the data to be labelled, respectively. Adjustments could be made to improve the algorithm's robustness to FPs, and consequently the precision. To improve the FPs identified in the results, the algorithm could be improved by refining the state estimation algorithm to add a confidence score for RSO

detections based on metrics such as its motion history. This confidence score could then be used to limit which detections are found to be RSOs, decreasing the number of FPs. This could also decrease the number of FNs, since a high-confidence track could be estimated more than once. Additionally, the thresholding process could be improved or changed entirely to prevent lit pixels from challenging illumination conditions from being detected as objects in further processing steps. These improvements are currently being researched to enhance the algorithm.

To improve the centroiding accuracy of the algorithm (to subsequently improve the positional accuracy of RSO detections made by the algorithm), a centroiding method which more accurately considers the pixel intensity and shape of the detected objects is being considered. A CNN-based algorithm is also being investigated since the shapes of the detections can be learned and used to separate detections from the background. This would mean that even challenging images such as in Figure 3.10 could be processed for RSO detection, expanding the number of RSOs that can be detected, as well as the versatility of the RSO detection algorithm and usefulness of the FAI images for RSO detection.

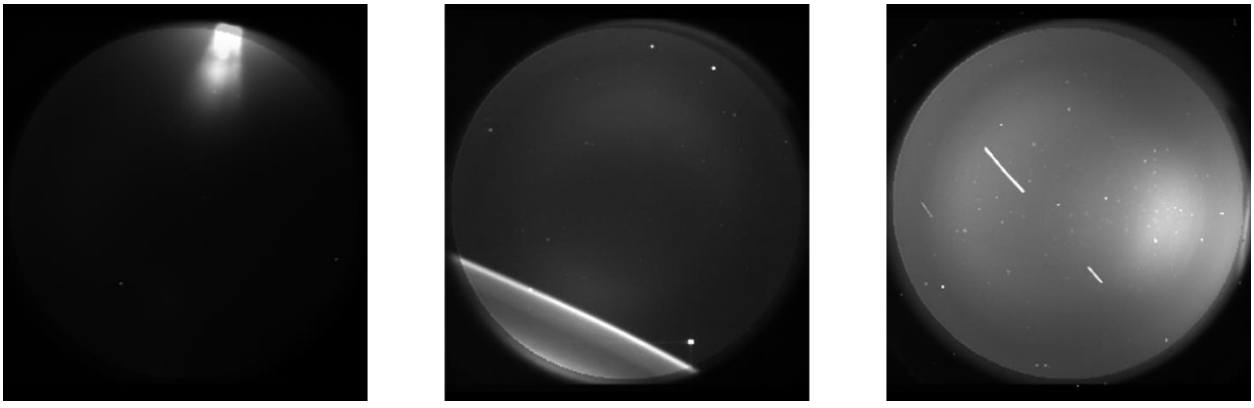

Figure 3.10. Three different examples of challenging images from the FAI. RSOs are still present within these images but are harder to detect.

Though the algorithm was only tested on FAI images, it is still applicable to other situations in which the background stars may be moving as well, as described in previous sections. Since the ICP algorithm can robustly determine rigid transformations, the algorithm can be used in scenarios such as an imager on a stratospheric balloon platform. During unstable conditions such as the launch of the balloon and high wind, the background stars will appear to move with respect to the imager's FOV due to excessive motion [19]. Another situation in which the background stars will move is during a spacecraft's slew operation. The algorithm could theoretically be used to correct the rotation and translation of these stars and detect RSOs even in these challenging conditions, and in the latter case, RSO detection can be performed as a secondary task. The algorithm's effectiveness in these situations is currently being investigated.

3.8. Conclusions

In this research, an RSO tracking algorithm was developed to track RSOs in low exposure, low resolution, wide FOV star field imagery, where the background stars appear to move, due to spacecraft attitude motion. The algorithm was built using rules-based methods, making use of computer vision techniques for object detection, ICP for star removal, and linear motion modelling for RSO classification. The algorithm was tested on real FAI imagery, as they are publicly available and share specifications similar to commercial star tracker imagery. The algorithm yielded a precision of 87% and 79% in the per-frame results and full-sequence results, respectively, while yielding a recall of 63% and 71% in those respective result categories. Overall, the algorithm detected 87% of RSOs at least once, during each RSO's transit through the imager's FOV. The stars detected in these sequences were centroided with an accuracy of 0.66 pixels in the x-axis, and 0.72 pixels in the y-axis, and were on average 5.02 in visual

magnitude. The algorithm's recall could be improved by incorporating better motion modelling and faint RSO detection, while the precision could be improved by adding further robustness to challenging illumination conditions. The centroiding algorithm is being improved to consider the shape and pixel intensity of the detected objects, while CNN-based detection is being considered to increase the number of RSOs that can be detected in these images. These improvements are currently being investigated, alongside the algorithm's performance on night sky images taken during unstable conditions on a stratospheric balloon.

While the process was challenging, the capability to detect RSOs from starfield images taken on a moving, space-based platform provides an opportunity to examine the feasibility of SSA operations during slew maneuvers, collision avoidance operations, harsh lighting conditions, or a satellite that cannot maintain stability during its observation period. For example, any satellite with a functioning star tracker can serve as an SSA instrument to provide useful information, even under the conditions mentioned above.

4. Towards a Benchmark Starfield Dataset and Automated Data Labelling for Unresolved Resident Space Object (RSO) Tracking Algorithm Development

This chapter is based on a paper currently in preparation describing multiple fundamental components of RSO tracking algorithm development, including a framework for efficiently labelling RSO imagery, a preliminary dataset, a paradigm for RSO tracking algorithm development, and a sample RSO tracking algorithm. Given the lack of a benchmark dataset, algorithms developed for RSO tracking often used their own data, which was often not detailed enough, shared, or coordinated, making algorithm comparison difficult. This ultimately made the overall progress in the field of RSO tracking ambiguous, since algorithms developed on different datasets are not directly comparable. A benchmark dataset and iterations of such a dataset are difficult to produce, given the laborious nature of labelling small, visually similar objects, which are the stars, RSOs, and noise in these images. For this reason, an automated labelling tool suite was proposed and developed in this research to efficiently produce rich annotations in multiple formats. Using these tools, a preliminary benchmark dataset was created and described as well. Finally, an RSO tracking algorithm paradigm was suggested for future algorithm development, along with a sample algorithm presented using this framework and corresponding preliminary results. A preliminary 500-image benchmark dataset was developed using the proposed tools, with more classes and data formats than was developed in previous research. This highlights the potential that this labelling framework has, as well as the progress made towards a final benchmark dataset. An RSO tracking algorithm was also successfully developed using the

proposed tracking algorithm paradigm and successfully trained on the preliminary dataset, demonstrating the potential of both the paradigm and dataset for future algorithm development.

I developed the annotation tools described in this chapter, along with the proposal of the automated labelling framework. I created the 500-image preliminary benchmark dataset using these tools, carefully selecting image sequences to include a variety of imaging conditions. I developed the RSO tracking algorithm in this study, using a custom implementation of a CNN for the detector, a custom implementation of an assignment algorithm for the first tracker, and another custom algorithm for the second tracker. I prepared all the figures and tables and wrote the manuscript. Regina Lee provided me with her expertise and direction in carrying out this research. All coauthors provided feedback and revisions for the manuscript.

4.1. Introduction

SSA is a topic of increasing importance in recent years, given the sudden and exponential growth of RSOs [5]. These RSOs consist of active objects and debris, such as weather satellites and rocket bodies. Small objects, in the order of 1 cm to 10 cm, such as fragments of debris, are of most concern, given their large number and difficulty in detection [9]. Currently, the United States SSN provides most of the publicly available SSA data on these objects, as a catalogue [15]. However, there are limitations to this data, such as the lack of data for objects smaller than 10 cm and the numerous objects that have been lost in the catalogue [15,12]. Better data is required to ensure that the characteristics of more RSOs are well understood, necessary to avoid catastrophic space-based collisions [10]. One way to improve SSA is to leverage space-based optical sensors to capture images, and then identify the RSOs within them. Using imagery from

such space-based sensors has a variety of advantages, including better coverage and imaging windows than ground-based telescopes. However, this is a complicated, multistep process, which begins by detecting and tracking the RSOs within the images.

Given the large distances between RSO observations and the host sensor, the RSOs will usually appear as unresolved sources of light. In the case of long-exposure imagery in the order of seconds, a streaking effect will occur, given the movement captured during the exposure time. This movement may be due to the observed RSO's movement relative to the host imager, the host imager's movement relative to the observed RSO, or a combination of both. Examples of recent research that developed algorithms to process images of this kind can be found in [74] and [75]. In the case of short-exposure imagery in the order of 100s of milliseconds or less, the RSOs in the images will tend to appear as point sources of light. Star trackers, optical devices used to determine the attitude of a spacecraft from stars detected in their imagery, use short exposure times [76]. Given that many space-based payloads are equipped with star trackers, detecting and tracking RSOs within this kind of imagery is a promising and efficient solution to bolster existing SSA systems.

However, the point-source appearance of RSOs means that algorithms using single-image feature detection methods will struggle to detect and track RSOs across images. Methods such as source detection, derivative-based edge detection, template matching algorithms, and even spatial-feature-based ML algorithms may struggle to tell apart RSOs from stars and noise in an image [55,56,57,58]. For this reason, methods that incorporate temporal information are preferred. In Chapter 3, an analytical algorithm was developed to detect RSOs in star-tracker-like imagery from the FAI instrument, an optical imager onboard the CASSIOPE satellite. This method

processed sequences of images in a rolling window fashion, then applied a linear motion model to track RSOs. In [49], a CNN was used to detect all sources in a sequence of images, including stars, RSOs, and noise, then a graph-based MOT algorithm was used to track objects, and finally tracks were classified as stars, RSOs, or noise. Lastly, in [77], a Faster Recurrent Convolutional Neural Network (Faster R-CNN) was combined with various preprocessing techniques to detect RSOs.

Regardless of the method, algorithm development requires quality, labelled data. To measure algorithm improvement over time, a benchmark dataset is also needed, where algorithm developers can compare their algorithm's performance to other algorithms on the same dataset. Datasets in fields such as autonomous vehicle development, surveillance, and even particle tracking are often followed by challenges. The datasets are often hosted online, where researchers can access them to develop their algorithms. A competition is then held, where the algorithms developed by various researchers are tested against a held-out dataset on a variety of metrics. Examples of such benchmark object tracking datasets or associated challenges include KITTI, Visual Object Tracking (VOT), TrackingNet, MOTChallenge, UAV123, Tracking Any Object (TAO), and Particle Tracking Challenge [78,79,80,81,82,83,84]. Datasets such as VOT and its associated challenges have gone through many iterations. Datasets are usually expanded and updated, new metrics are introduced, and competitions are organized again [85]. Over time, algorithm performance on each evaluation metric usually increases for a particular version of the dataset, as researchers develop better methods [81]. The idea is that such improvements translate to better algorithms developed for real-world applications.

In terms of RSO tracking, such a benchmark dataset, composed of real, space-based, unresolved RSO observations, is not known to exist. This may be due to the difficulty in acquiring or labelling such imagery, or the effort and coordination involved in developing a benchmark. In [86] and [87], algorithms were developed to detect spacecraft, and their corresponding datasets were introduced. While these algorithms achieve relatively good results on these datasets, the algorithms and datasets are not representative of the unresolved RSO tracking problem that this research is concerned with. In [86] and [87], some of the data was synthetically generated and individual satellite components were visible, which may not be accurate representations of the RSOs imaged by a star-tracker-like sensor. Furthermore, these datasets and algorithms are concerned with object detection alone, and do not consider the corresponding identities of the objects in subsequent images, using tracking annotations. In [88] and [89], algorithms and datasets similar to [86] and [87] were introduced, with notable distinctions from [86] and [87]. These datasets are concerned with the problem of autonomous spacecraft pose estimation, where it is desired not only to detect spacecraft in images, but to determine their attitudes as well. In [88], the Spacecraft Pose Estimation Dataset (SPEED) was provided, consisting of spacecraft detection annotations as well as spacecraft attitude annotations. Metrics for an algorithm developed in this work are also provided. In [89], the SPEED+ dataset is provided, making use of a robotic simulation testbed for mockup spacecraft images to improve upon the SPEED dataset. This dataset sought to improve the robustness of developed algorithms and their performance drop between synthetically generated and real data (referred to as the “domain gap” issue). Again, while these datasets address the key problem of autonomous spacecraft pose

estimation, they do not contain the images or annotations necessary to develop algorithms for unresolved RSO tracking in space-based imagery.

As mentioned previously, algorithms have been developed for the specific problem of unresolved RSO tracking in short exposure space-based imagery. However, each of the datasets used in these studies had their limitations. The dataset in Chapter 3 only included annotations for RSOs, while the annotations were limited to bounding boxes, which can miss pixel-level details. Furthermore, the annotation process and attributes are not described in detail. In [49], synthetic and simulated images were used to attempt to supplement the lack of availability of real, annotated images. Furthermore, the annotation attributes and annotation process for the real FAI images are also not described. Lastly, the study mentions that detection accuracy could be increased provided examples of real, annotated data are added to algorithm training. While the dataset in [77] describes the image quantity used, the specific sequences used and annotation attributes are similarly unclear, as well as the annotation process. This study also discusses the need for an expanded training dataset.

The differences between the datasets used in each study means that the developed algorithms cannot be compared exactly. Combined with this problem, the discussed dataset limitations highlight the need for a well-developed, benchmark dataset. The difficulty in producing such a benchmark dataset lies in the time-consuming, laborious effort of labelling starfield imagery. Current tools to annotate images lack many of the specific functions required to label these images.

This chapter seeks to introduce a labelling pipeline and corresponding labelling tools to efficiently annotate starfield imagery. This chapter also provides a preliminary 500-image

starfield dataset developed using the proposed labelling tools, with significant advantages over the previously developed datasets. Finally, this chapter provides an RSO tracking algorithm development framework, using the track-by-detection paradigm. A sample tracking algorithm composed of a CNN and custom tracker to test the developed dataset is also provided, along with preliminary qualitative results. Section 4.2 discusses annotation details related to starfield images, including the shortcomings of existing tools, challenges, and annotation attributes. Section 4.3 discusses the annotation tools developed in this study and proposes a framework of tools to streamline the annotation process. Section 4.4 describes the preliminary dataset developed, including details of the host imager. In Section 4.5, the RSO tracking algorithm framework is described. In Section 4.6, the sample RSO tracking algorithm is discussed, with the preliminary qualitative results given in Section 4.7. Section 4.8 and Section 4.9 cover the future work and conclusions of this research, respectively.

4.2. Starfield Image Labelling

4.2.1. Existing Annotation Tools

There currently exist many data annotation tools to annotate data of all kinds. V7, Encord, CVAT, and Labelbox are all examples of feature-rich, well-developed data annotation tools [90,91,92,93]. Every one of these annotation tools has the capability to label multiple forms of data, including general images, videos, documents, and even medical data. These data annotation tools are often web-hosted, allowing users to submit their data online to take advantage of tools built into these websites. All these annotation tools also incorporate automation through leveraging foundation models to assist users in labelling their data, which is known as Model

Assisted Labelling (MAL). This makes the task of labelling images with objects such as people, animals, and vehicles efficient. These tools also offer a labelling workflow, allowing multiple collaborators to label the same dataset and functions to organize annotation tasks. While free, demonstration-level access is given to use these tools, full use of the advertised features is provided through various pricing schemes, in tiers based on user needs. CVAT even offers a labelling service to have data labelled by professionals [92].

4.2.2. Challenges Labelling SSA Imagery

While these data annotation tools serve their purpose and offer efficient general labelling methods, they have their shortcomings. The rich features offered by these tools are unnecessary for the task of labelling objects in unresolved space imagery. Furthermore, these tools offer limited features in their free versions, if available at all. Finally, a web-based labelling tool may not be suitable for labelling SSA imagery, given the sensitivity that may be associated with this data.

There are also specific challenges with labelling SSA imagery, which annotation tools need to address. The objects in this type of imagery are often very small, sometimes just a single pixel in size. These objects are often very faint as well, and some scenes can be crowded with hundreds of objects. The various objects within the images, such as the stars and RSOs, tend to look almost identical, and consequently the motion of the objects needs to be considered to distinguish them. Finally, the illumination conditions in these images tend to change as well, both within a single image and throughout a sequence of images.

Annotation tools need to address these challenges, producing detailed annotations efficiently. Pixel-level annotations are desired to be able to capture these small objects, while bounding-box-style annotations are also preferred for algorithm compatibility. Furthermore, these tools need to be open source and simple, while also providing automation to speed up labelling. Finally, these tools should consider users' privacy needs, offering the ability to have self-hosted labelling.

4.2.3. Annotation Format, Classes, and Attributes

To test, train, and validate RSO tracking algorithms, the annotated data should have several different formats. As mentioned above, having pixel-level annotations would be beneficial to capture small details, and are often used in CNNs performing semantic segmentation, such as U-Net [94]. To accomplish this, the annotated data should include masks, images where each pixel represents the class of that pixel in the corresponding real image. Figure 4.1 is an example of an image and a corresponding mask. Masks can help algorithms learn the shapes and features of objects in these images. Another important form of annotations is bounding box classes and coordinates. CNNs such as You Only Look Once (YOLO) instead use these bounding box annotations to learn shapes and features of the objects [95].

These bounding box annotations should consist of several attributes. Standard YOLO format consists of an object's class, represented by a positive integer, the object's center x-coordinate, the object's center y coordinate, the object's width, and the object's height [96]. These coordinates and dimensions are often normalized to a scale of zero to one. Each image can have multiple objects, and each object is a new line with these attributes in a text file corresponding to each image.

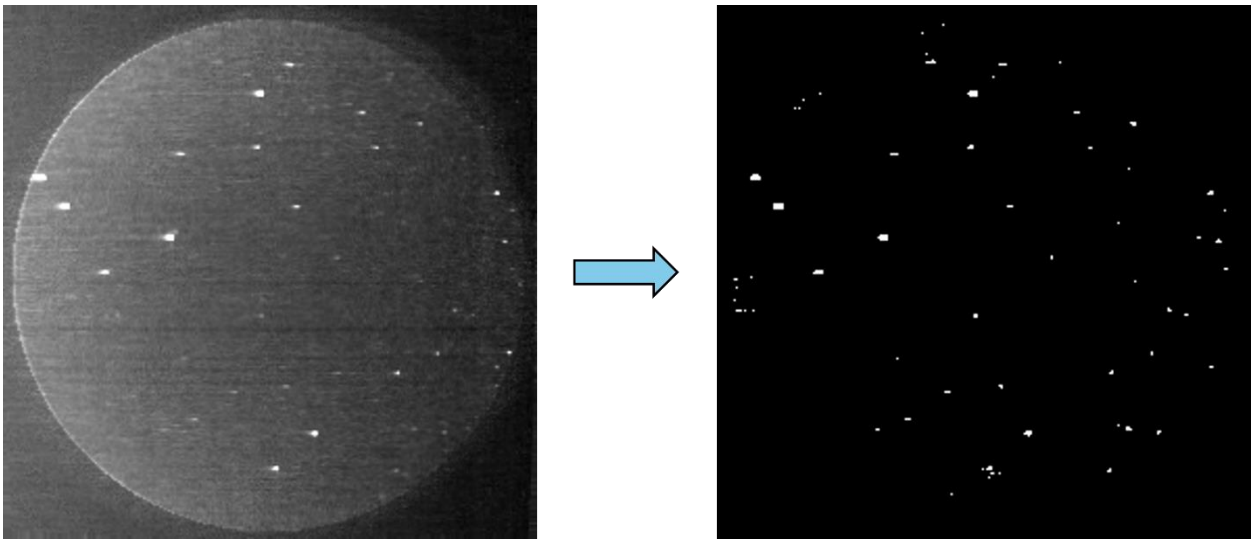

Figure 4.1. An example of a starfield image and its corresponding mask. The mask consists of two classes, objects (white pixels, or high values) and background (black pixels, or low values).

While CNNs such as U-Net and YOLO can serve as the first step in a tracking algorithm pipeline, they do not use tracking-related attributes, since they are object detection models. To test a full tracking algorithm in terms of its ability to track objects throughout a sequence, a unique object identifier is needed as an additional attribute in the truth data. Each object in an image sequence should carry this unique identifier throughout the image sequence. An example of such an annotation protocol that uses a unique identifier is provided in [97].

4.3. Annotation Tools

To generate annotations described in the above format, several prototype tools were developed in Python. While these tools were used to successfully and efficiently generate a preliminary benchmark dataset, they are meant to serve as a demonstration for a framework of labelling. The idea for these tools was to use previously developed RSO tracking algorithms to automate some of the labelling process (similar to MAL), while developing an intuitive, easy-to-use Graphical

User Interface (GUI) with only essential functions. The tools were developed as a four-module suite, consisting of automated mask generation, mask revision, automated object classification, and object class revision. The module system was developed so that individual tools could be entirely replaced with improved tools, without affecting the rest of the tool suite. Over time, as RSO tracking algorithms improve, these tools will be improved or swapped out entirely by integrating the algorithms themselves into the tools. For the automated tools developed in this study, components of the RSO tracking algorithm developed in Chapter 3 were built into them. Figure 4.2 depicts the labelling flow using the labelling tool suite developed.

4.3.1. Automated Mask Generation

As mentioned, masks are important for helping semantic segmentation CNNs like U-Net understand the features of the data. To start developing these masks, the first tool in the suite was created to automatically generate masks. This tool takes the original images and produces masks for them. The tool incorporates the windowed multi-Otsu thresholding technique described in Chapter 3 to threshold the images. The window (slice) size and offset are each built into a GUI as sliding buttons, alongside the thresholding result. This is displayed to the user, and the thresholding result can be seen in real-time. Additional buttons such as “Save” and “Next” are incorporated to save the mask and proceed to the next image, respectively. A “Save Blank” button is incorporated in the case that an ideal mask cannot be generated for an image, in which case a blank mask is created to manually develop using the next tool. Figure 4.3 is an example of an input image, and the GUI a user sees for developing a mask for that image.

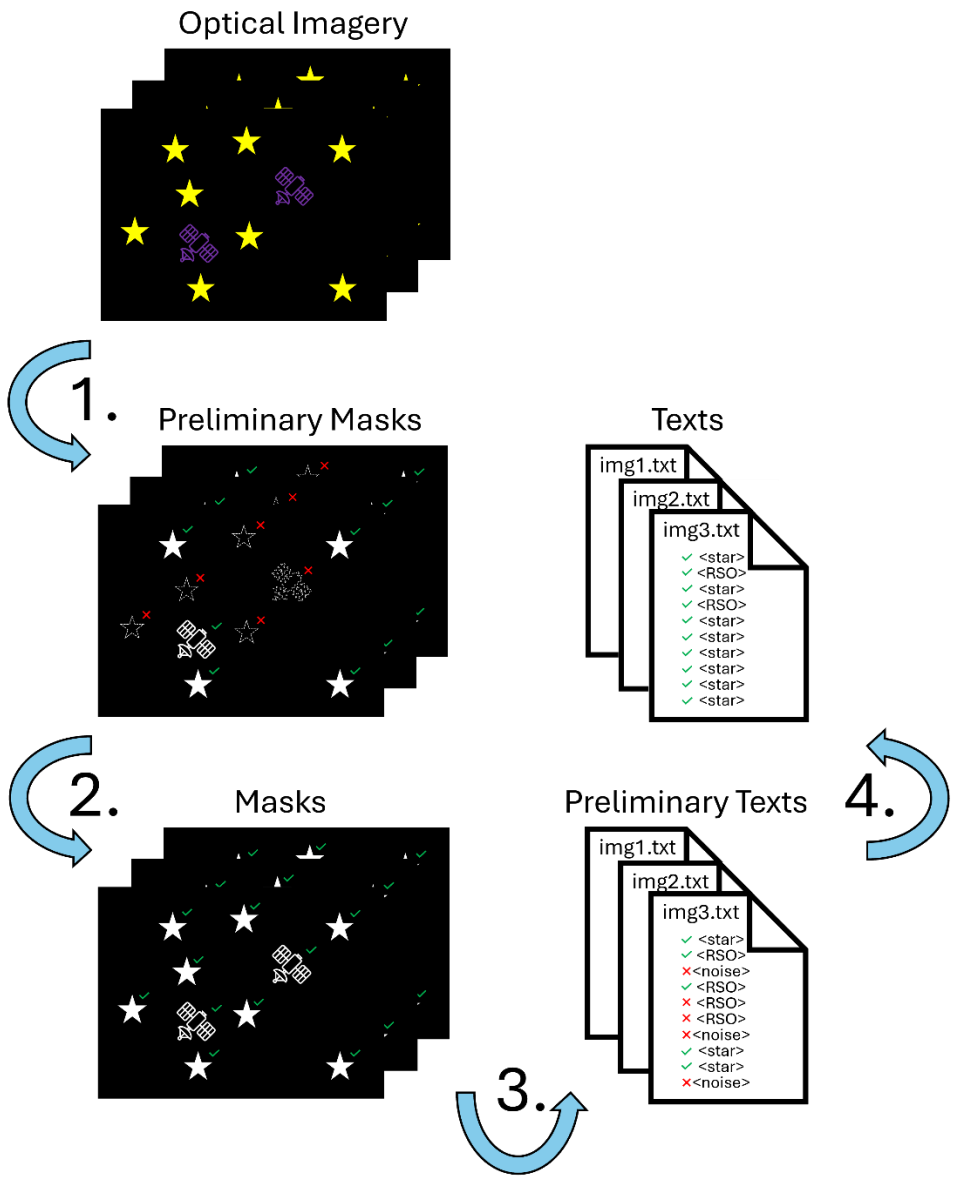

Figure 4.2. Illustration of the labelling flow using the developed 4-tool labelling suite. A combination of automation and manual revision is used to generate the masks and text files.

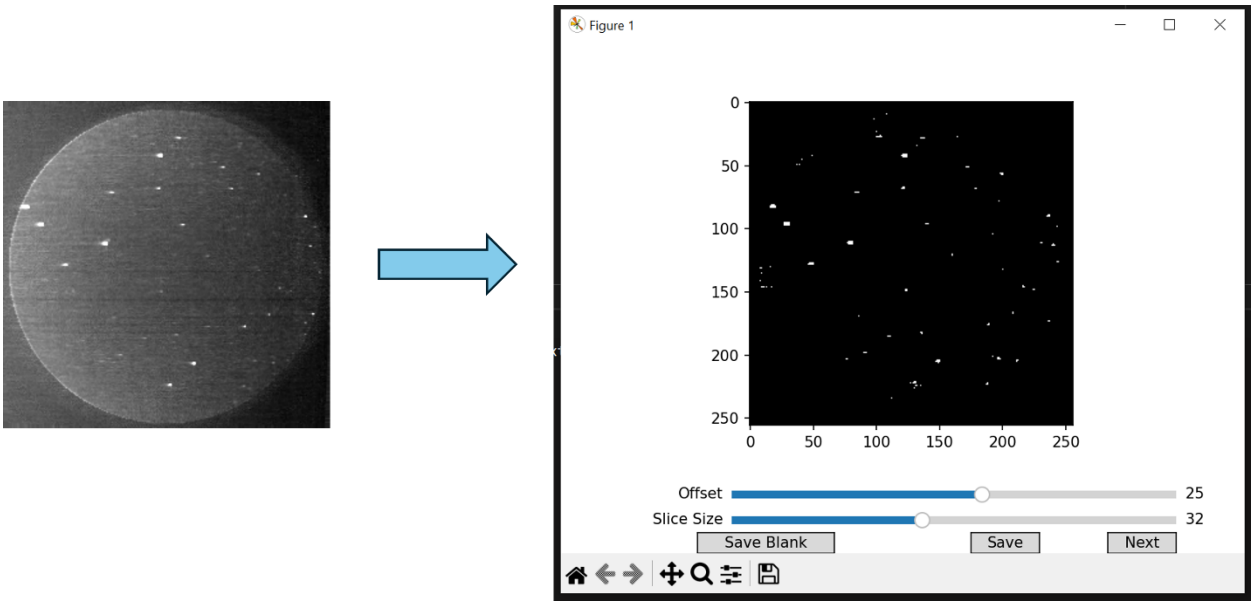

Figure 4.3. An example of an input image and the automated mask generation tool's GUI for developing the respective mask.

4.3.2. Mask Revision

The masks generated by tuning parameters in the previous step may not capture all the objects in an image or may erroneously define pixels that are not objects as being so. A tool was needed to address this issue in an efficient, user-friendly way. Manual mask revision accomplished this by providing a means of manual mask revision. This tool takes in the masks developed in the previous step, and enables users to produce hand-corrected, polished masks. This tool was designed to take the mask and corresponding original image and create a composite image that the user could view, letting them understand which pixels should be added and removed from the mask. This composite image overlays the mask on to the original image, where the objects in the mask, denoted by high pixel values (255 in this case) become translucent colours, and the background in the mask, denoted by low pixel values (0 in this case) become completely

transparent. This composite image lets users see colours representing the objects in the masks superimposed on the actual objects in the image. Users can then visually verify if the objects in the mask indeed correspond to real objects, and whether some objects are missed. By holding down right-click on their mouse, users can remove pixels from the mask for incorrectly detected objects, and by holding down left-click users can add pixels to the mask for missed objects. This is done in a brushing fashion for efficient labelling. The GUI consists of this composite and function, the file path of the current image, and keyboard shortcuts to traverse and save images. Figure 4.4 illustrates this tool's GUI, showing examples of objects that were missed and incorrectly identified in the previous step. A shade of pink was used as the colour to denote objects in the mask.

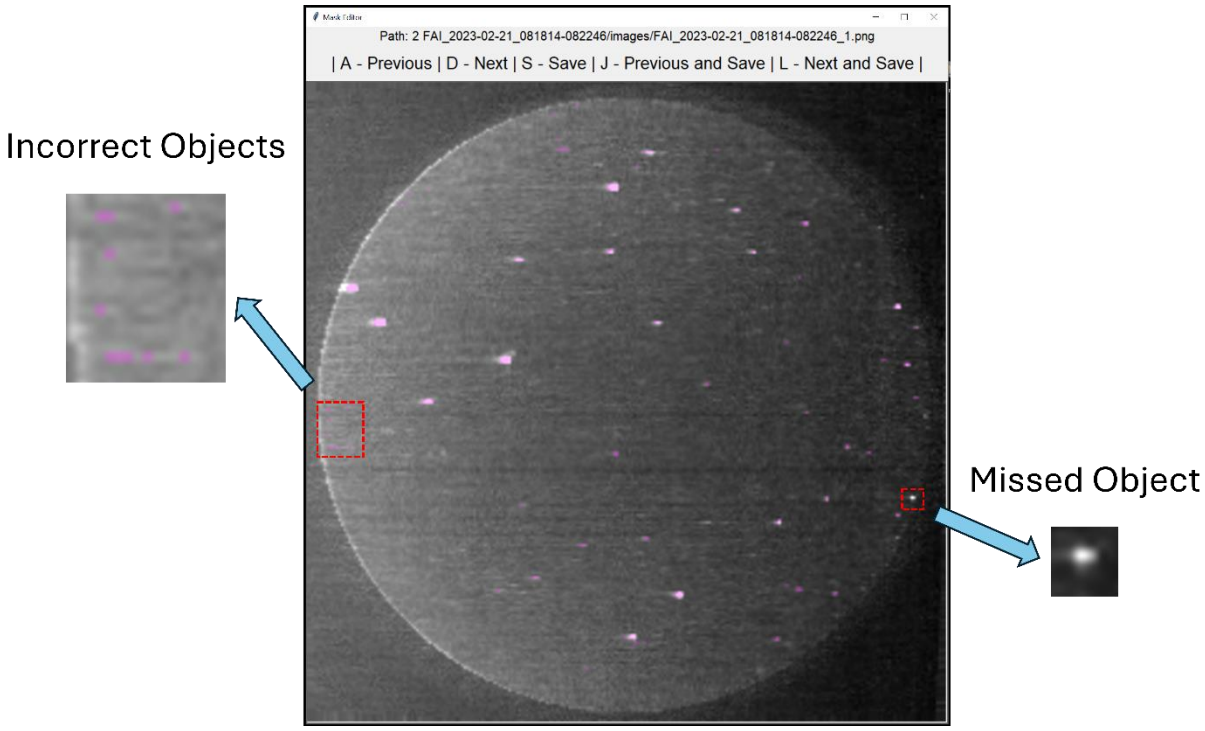

Figure 4.4. An example of the mask revision tool's GUI. Examples of a missed object and incorrectly detected objects are pointed out.

4.3.3. Automated Object Classification

To make distinctions between the various objects in SSA imagery, algorithms need truth data which also contains the classes corresponding to the objects. The next tool in the tool suite seeks to accomplish this by automatically classifying objects in the images. The tool takes in a sequence of at least three images and produces annotation data in YOLO format and images consisting of masks with coloured bounding boxes around the objects, denoting the class of the object. These images are for illustration purposes only, to help users visually understand the automated algorithm's performance. The automation was integrated into this tool by utilizing and modifying the RSO tracking algorithm from Chapter 3. The objects within the masks are first identified as unique objects using a CCA algorithm, with properties such as the bounding box width, height, and center coordinates extracted [68]. The center coordinates are then used for the next steps. To classify stars, the star removal algorithm discussed in Chapter 3 was modified. For the version of the algorithm in this tool, after detecting the dominant rotation and translation of the points, the points were classified as stars, rather than removed from the images. The RSO tracking step of the algorithm in Chapter 3 was applied to the remaining objects in this tool. Points that travelled linearly and were equidistant between three images (as defined as the RSO detection criteria in Chapter 3) were classified as RSOs. Finally, the objects remaining after both star and RSO classification were deemed to be noise and were classified as such. This tool was developed as a simple run-to-click script, which processed the images and returned annotation data in batch, and therefore, did not need a GUI. Figure 4.5 below gives a visual overview of the inputs and outputs of this tool.

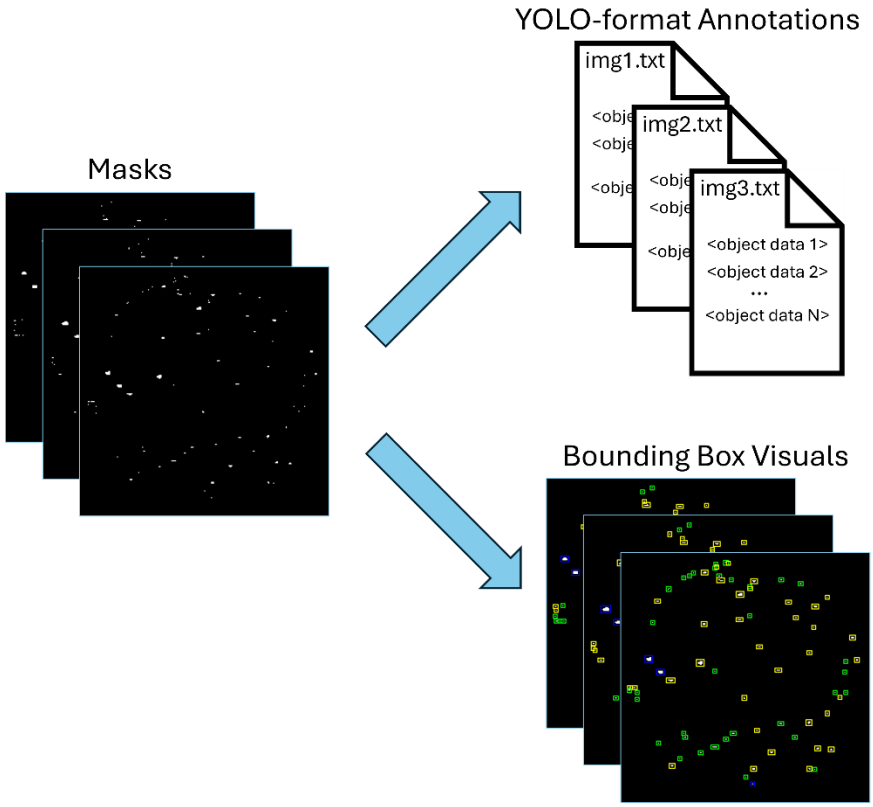

Figure 4.5. Visual overview of the inputs and outputs of the automated object classification tool. Both YOLO-format annotations and bounding box visuals are generated. In the visual, blue, yellow, and green bounding boxes are placed around the objects, denoting RSOs, stars, and noise, respectively.

4.3.4. Object Class Revision

The algorithm in the automation tool in the previous step will not always accurately classify every object in the masks. For this reason, the next tool was developed as an efficient manual method of changing object classes. This tool takes the generated annotations from the previous step and uses them to draw bounding boxes, similar to the visuals, to then display to the user in a GUI. The user is then able to set a class and click on objects to change their class to the set class.

The YOLO-format annotations and bounding box visuals are automatically updated with these class changes once the user saves that particular image. The GUI consists of this bounding box image, buttons to select the class, the file path of the current image, and keyboard shortcuts to traverse and save images. Figure 4.6 shows an example of the GUI.

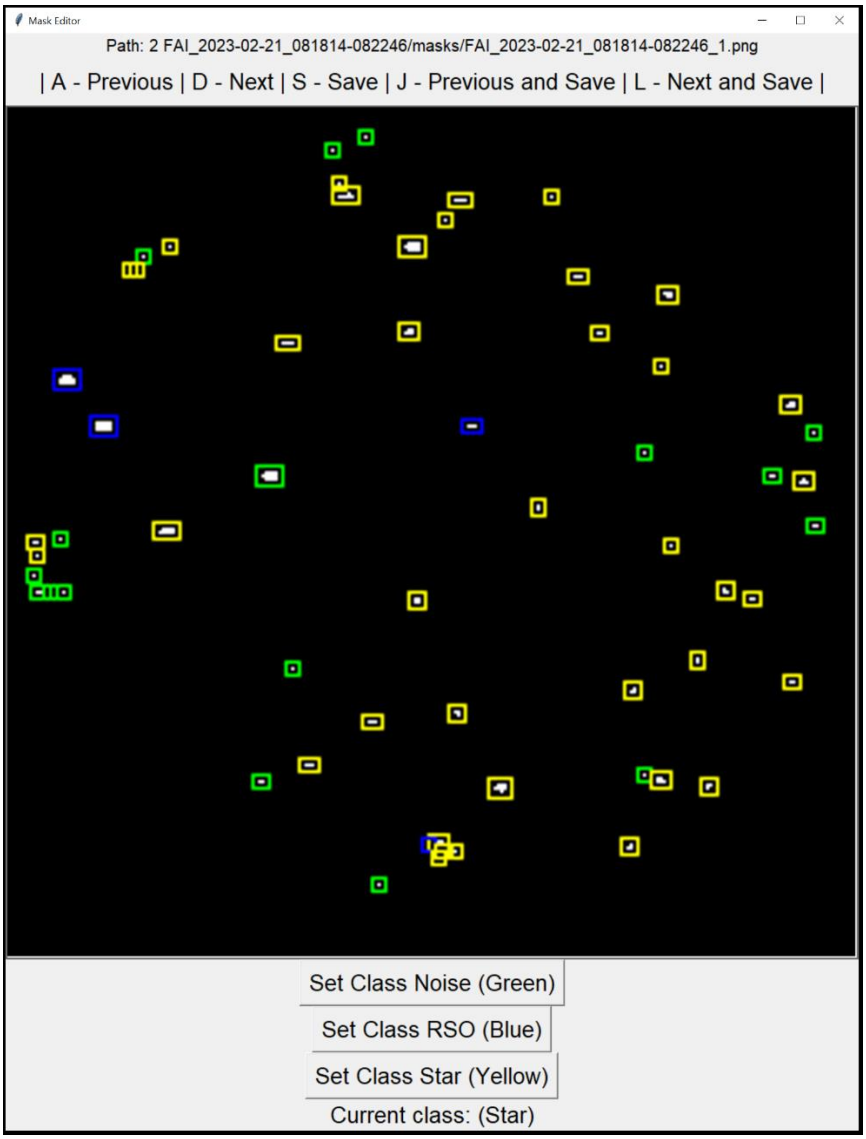

Figure 4.6. An example of the object class revision tool's GUI. The user is able to set a class and click on objects to change them to the selected class. The YOLO-format annotations are automatically updated with these class changes once the user saves.

4.4. Dataset

As mentioned previously, there is limited availability for well-annotated, real, benchmark datasets for unresolved RSO imagery. Using the tools developed in the previous section, a preliminary benchmark dataset was developed, both to demonstrate the efficacy of the labelling tools, and to move towards the development of a benchmark dataset.

4.4.1. Host Imager and Image Characteristics

The data that was labelled and used to construct this dataset was captured by the FAI instrument, a star-tracker-like optical imager onboard the CASSIOPE satellite [64]. While the imager was intended to capture the auroras, RSOs have been observed to pass through its FOV, appearing as unresolved point sources of light, due to the far distances of the observed RSOs and fast exposure time of the imager. Since access to the FAI instrument's data is publicly available [98], the FAI instrument presents a unique opportunity for benchmark dataset development. Detailed information about the FAI instrument can be found in Chapter 3, Section 3.3.

4.4.2. Previous Dataset

An 878-image dataset was created in the previous study to serve as testing data for the developed algorithm. Data came from 12 different sequences of imaging, from the year 2023. The number of images per sequence varied and were selected for characteristics such as RSO quantity variety and illumination variation. The data was labelled using CVAT, and annotations were in the form of YOLO-format attributes, as well as unique object identifiers. While this dataset was helpful in developing and testing the algorithm, the dataset had some shortcomings. Firstly, only RSOs

were annotated, meaning that algorithms that learned RSOs by differentiating them from stars or noise may suffer from a lack of annotations. Additionally, the annotations were only in the form of YOLO-format text files, meaning that semantic segmentation CNNs like U-Net could not use the data directly. Lastly, sequences with many, faint RSOs were not labelled, given the laborious nature of the annotation process.

4.4.3. New Dataset

With the development of the new labelling tools, the annotation process was improved significantly. The data could be annotated much faster due to the automation and better annotations could be produced. The new 500-image dataset includes classes for stars and noise, in addition to the RSOs. Furthermore, masks were generated in addition to text files, providing pixel-level details and truth data for semantic segmentation algorithms. This preliminary benchmark dataset consists of 10 sequences of 50 images each, also capture from the year 2023. Table 4.1 provides properties of the new dataset and old dataset.

Table 4.1. Properties of new dataset and old dataset.

Property	**Old Dataset**	**New Dataset**
Year	2023	2023
Image Resolution	256 x 256 pixels	256 x 256 pixels
Image Exposure Time	100 ms	100 ms
Image Quantity	878	500
Sequences	12	10
Images in Sequence	Variable	50
RSOs Annotated	Yes	Yes
Stars Annotated	No	Yes
Noise Annotated	No	Yes
Tracking Annotations	Yes	No
YOLO-format Text Files	Yes	Yes
Masks	No	Yes

4.4.3.1. Sequences Used

To ensure algorithms developed using this data learn from a variety of images, a balance between sequence variety and imaging continuity was necessary A 50-image length was standardized, since this time was long enough for slow-moving and fast-moving RSOs to both travel across the imager's FOV, and temporally changing illumination conditions would be captured. Figure 4.7 shows an example of the first and 50^{th} image of a sequence, demonstrating that illumination conditions can change in just 50 images. Limiting the sequence length to 50 images allowed

labelling efforts to also go towards using a variety of sequences. This is important because different effects, such as illumination and artifacts, can be captured in different sequences as well. Figure 4.8 shows an example of images from different sequences, highlighting the visual differences between them. However, it is important to note that imaging sequences captured by FAI are not limited to 50 images, and longer imaging sequences are provided in the publicly available data. The 50 best images from each imaging sequence were selected, based on criteria such as the number of RSOs present and the visual variety of the images. Table 4.2 gives detailed information about the labelled sequences, including the imaging date of each sequence, the image range used from the available data, the number of unique RSOs in the sequence, and the total RSO detections. Since the tracking label generation has yet to be developed, identifying the number of unique objects for a class needs to be done visually. Given the numerous stars and noise in the sequences and that RSO tracking is the focus of this research, only the unique instances of objects were found only for the RSO class. Future research will seek to generate this metric automatically, for all classes. Additionally, a further 10 sequences in this dataset are currently being labelled and are left out of Table 4.2. The 10 sequences captured in Table 4.2 will be focused on instead, and together create the preliminary dataset discussed in this study.

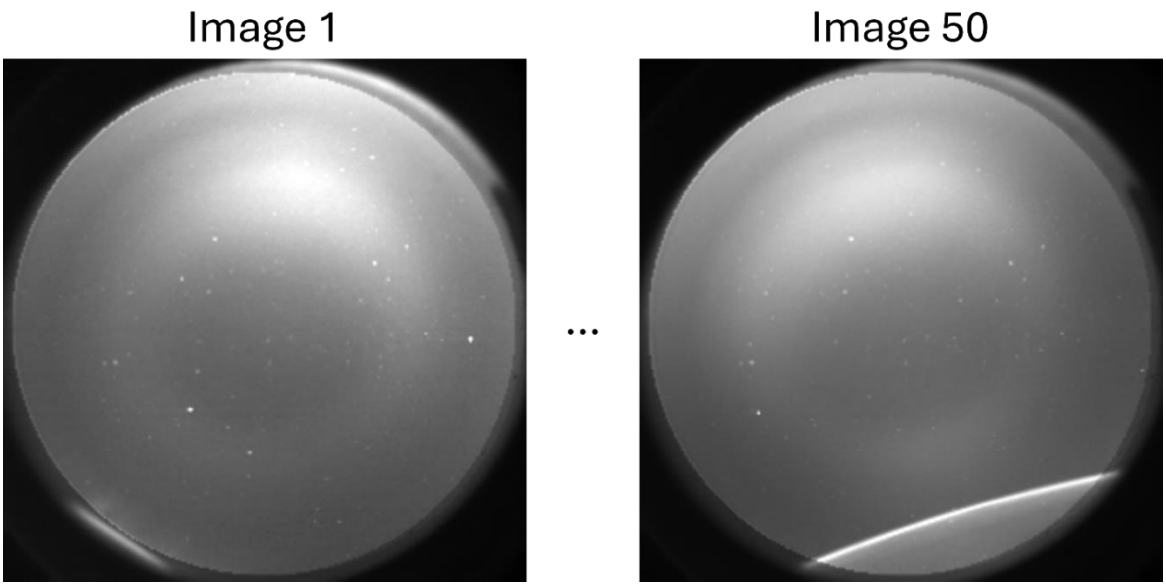

Figure 4.7. An example of the first and 50th image of an image sequence, demonstrating that imaging conditions change temporarily, within a single sequence.

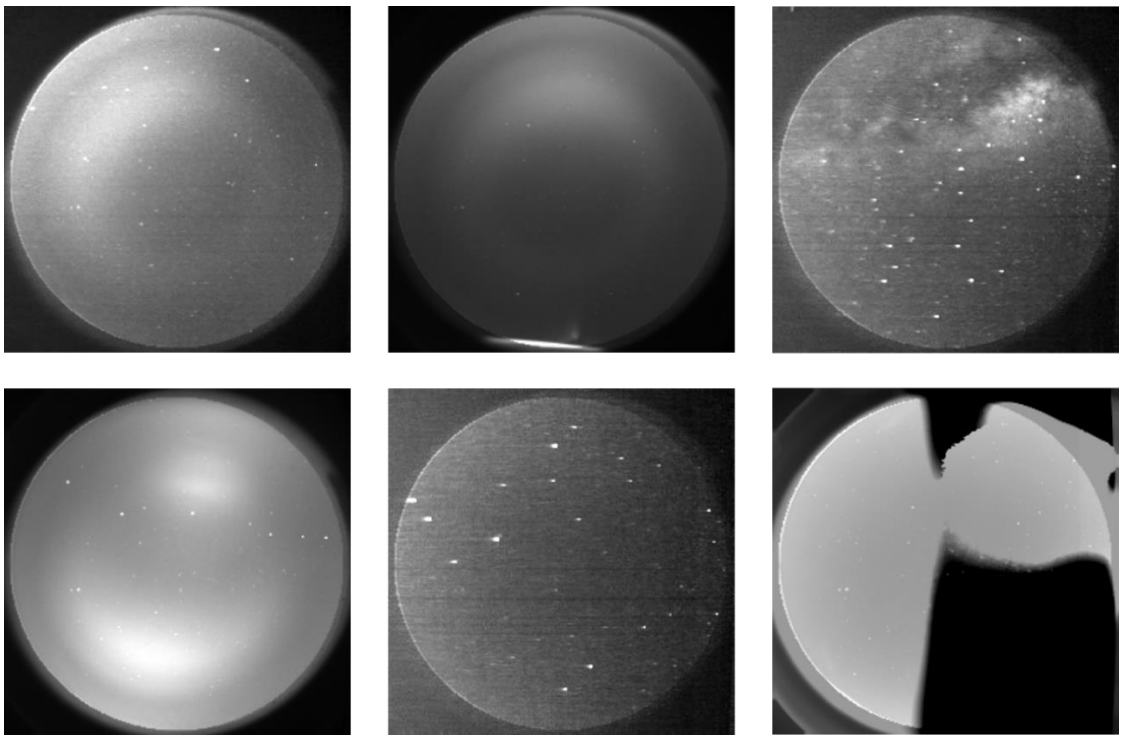

Figure 4.8. Example of six images from different sequences within the dataset, demonstrating that imaging conditions can vary between sequences.

Table 4.2. Description of the imaging date of each sequence, the image range used from the available data, the number of unique RSOs in the sequence, and the number of RSO detections.

Sequence Number	**Date**	**Image Range Used**	**Unique RSOs**	**RSO Detections**
1	2023-02-21	70 to 119	14	263
2	2023-03-16	1 to 50	4	119
3	2023-03-26	1 to 50	5	160
4	2023-04-22	1 to 50	13	364
5	2023-05-08	1 to 50	4	86
6	2023-05-10	1 to 50	2	57
7	2023-06-05	1 to 50	3	142
8	2023-06-19	120 to 169	14	427
9	2023-07-20	1 to 50	5	62
10	2023-07-21	135 to 184	15	366
Total			**79**	**2046**

4.4.3.2. Class Analysis

To give a better understanding of how classes are represented in these images, the number of detections for each class, for each sequence is presented in Figure 4.9, along with the number of pixels in Figure 4.10. These figures are shown separately to develop an understanding of both the number of instances each object has and the amount of space in pixels these detections result in.

On average, each sequence of 50 images contains 205 RSO detections, 2836 star detections, and 401 noise detections. This results in an average of 69 total objects per image. On average, in terms of detections per image, stars represent 82%, noise instances represent 12%, and RSOs represent 6%. In terms of average pixels per sequence, background pixels represent 99.44%, stars represent 0.49%, RSOs represent 0.04%, and noise pixels represent 0.03%.

In terms of detections, stars are overwhelmingly the dominant class, with several times more instances than noise and RSOs. This is not unusual, as high-sensitivity wide FOV imagers capturing the night sky or empty space will mostly pick up stars in their FOVs. However, this does mean that the detection classes are imbalanced, and special considerations need to be made when developing algorithms. This is especially true for RSO tracking algorithms, given that they are the least represented class in terms of detections, yet are the most important class. The review conducted in [99] offers an extensive description of the techniques used to handle the issue of class imbalance. While RSO detections vary by sequence, generally, the more unique RSOs there are, the more RSO detections there will be in a sequence. However, fast moving RSOs and imaging sequences where RSOs have almost completed their transit across the imager's FOV will result in fewer RSO detections.

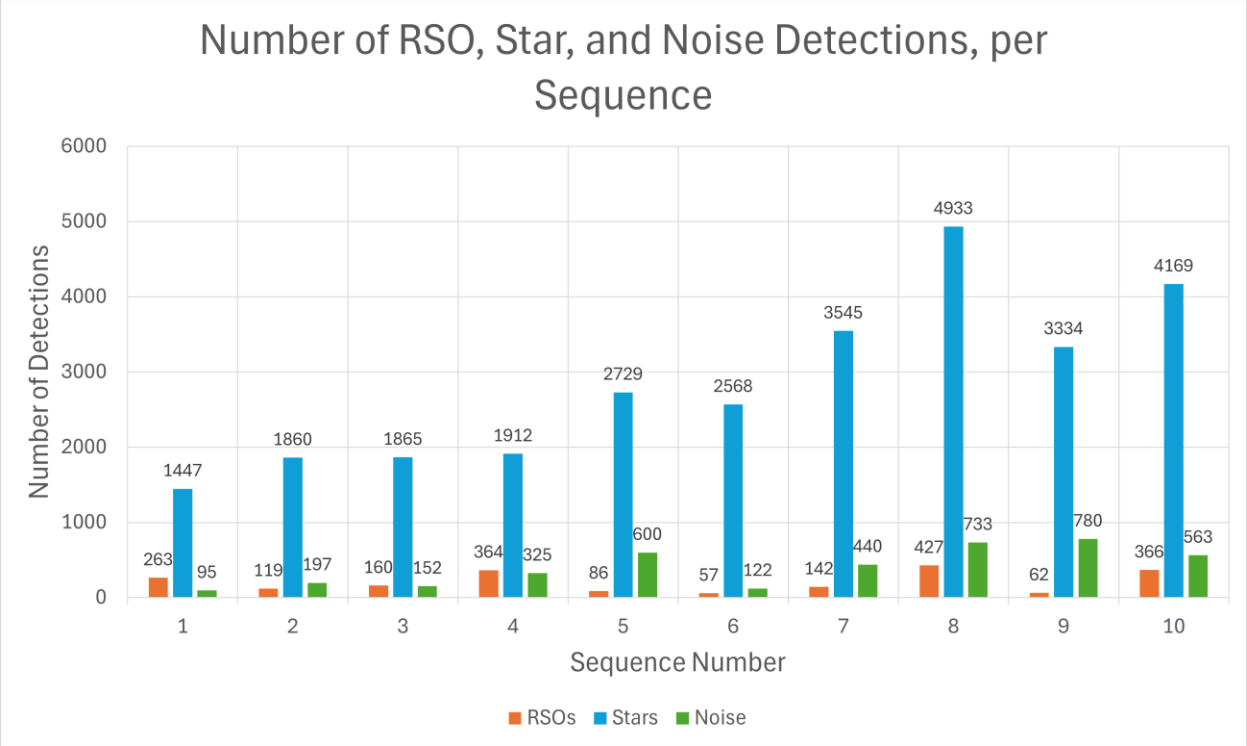

Figure 4.9. Number of RSO, star, and noise detections for each sequence in the preliminary dataset.

In terms of pixels, the background class is overwhelmingly the dominant class, accounting for almost all of the pixels in an average image. Again, this is not unusual, as unresolved SSA imagery of the kind this research focuses on will contain mostly background pixels. Furthermore, the FAI instrument has a circular FOV, circumscribed by the square sensor, which can be observed in several of the presented figures. This means that the corners of each image will by default represent the background class. Again, class imbalance techniques are needed to ensure that algorithms can properly learn to track RSOs.

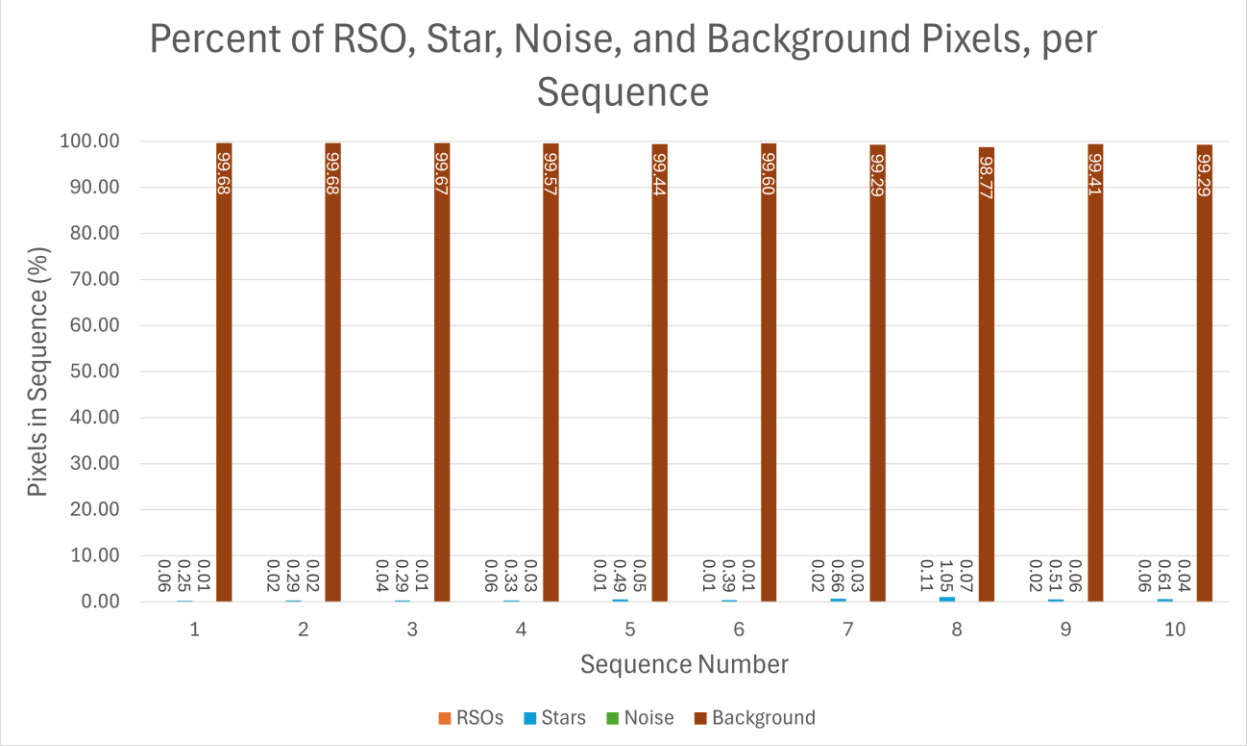

Figure 4.10. Number of RSO, star, noise, and background pixels for each sequence in the preliminary dataset, represented as percentages.

While the detections of the noise class are almost twice the number of the detections in the RSO class, they have almost the same number of pixels, on average. This is due to the fact that noise objects are typically very small, sometimes a single pixel, represented by artifacts on the imager's lens or hot pixels. Another important consideration for the dataset is the subjectivity of the noise class. In this dataset, the Milky Way, Earth's limb, and lighting artifacts were considered as part of the background class. However, arguments can be made that these objects should be part of the noise class. In any case, the background class is an important part of the dataset, given that algorithms need to distinguish between source objects like stars and RSOs as well as background objects like those mentioned above.

The number of objects of each class per sequence varies, as well as the total number of objects per sequence. Sequence 1 is an example of a low-noise, low-object-density sequence, while Sequence 8 is an example of a relatively high-noise, high-object-density sequence. Both types of sequences are important when developing algorithms. Low-object-density sequences are important to test algorithms for their ability to precisely detect objects, and not predict instances of objects where there are none. High-object-density sequences are also important to test algorithms for their ability to classify and keep track of objects, despite occlusions and close approaches between objects. These sequences are also important to evaluate an algorithm's processing time.

In general, it is not recommended to use a spatial feature classification algorithm directly on each image, to classify the objects. This is because most of the objects in an image look nearly identical, and algorithms that use spatial features to perform classification may struggle to distinguish them. Instead, it may be more ideal to perform classification as a later step, using the motion information of each object. For this reason, having many instances of stars is beneficial, since their motion needs to be precisely learned to distinguish them from RSOs, which can have similar motion in slow moving examples.

4.4.3.3. Annotation Format

As mentioned, annotations for this dataset are provided as masks and as labels. The parent folder contains one folder for the data for each sequence. These folders are named by sequence number as well as the date and time of the original image capture. Each of these folders contains five folders within them. The first folder is named "images" and has the original 50 grayscale FAI images corresponding to that sequence, in PNG format. It is important to note that these images

have been cropped, since the images available online contain additional rows of data in the header and footer of the image that is not useful for this task. The second folder is named "masks" and contains the corresponding 50 binary masks, also in PNG format. The third folder is named "texts" and contains the corresponding 50 text files which contain the annotation data for each image in YOLO format. The fourth folder is used internally by the annotation tool. This folder is named "objects" and holds 50 corresponding text files of preliminary data, used to generate bounding boxes for the GUI in the fourth annotation tool. The fifth folder is used for visualization purposes to help users understand the classes of the objects in the images at a glance, without having to go through the text files. This folder is named "boxes" and has the corresponding 50 bounding box images in PNG format, developed by the third annotation tool. Figure 4.11 illustrates the folder hierarchy and contents of this dataset.

4.4.3.4. Planned Dataset Improvements

The 500-image dataset developed using these tools is a preliminary dataset, serving as a foundation towards creating a benchmark dataset. To achieve such a high-quality benchmark dataset, several improvements need to be made. Firstly, tracking labels need to be built into the developed tools, to then generate tracking annotations for the developed data. Once this is done, another 10 sequences of 50 images each, corresponding to a total of 500 images will be labelled to create a 20-sequence, 1000-image dataset. 1000 images would be enough for algorithms to learn the data, given the similar shapes and trajectories of objects across images. These sequences should include various challenges, such as high-noise sequences, sequences without RSOs, and sequences with the Earth's limb in view. Object overlap should be considered as well, given that the generated tools currently cannot generate labels for object occlusions.

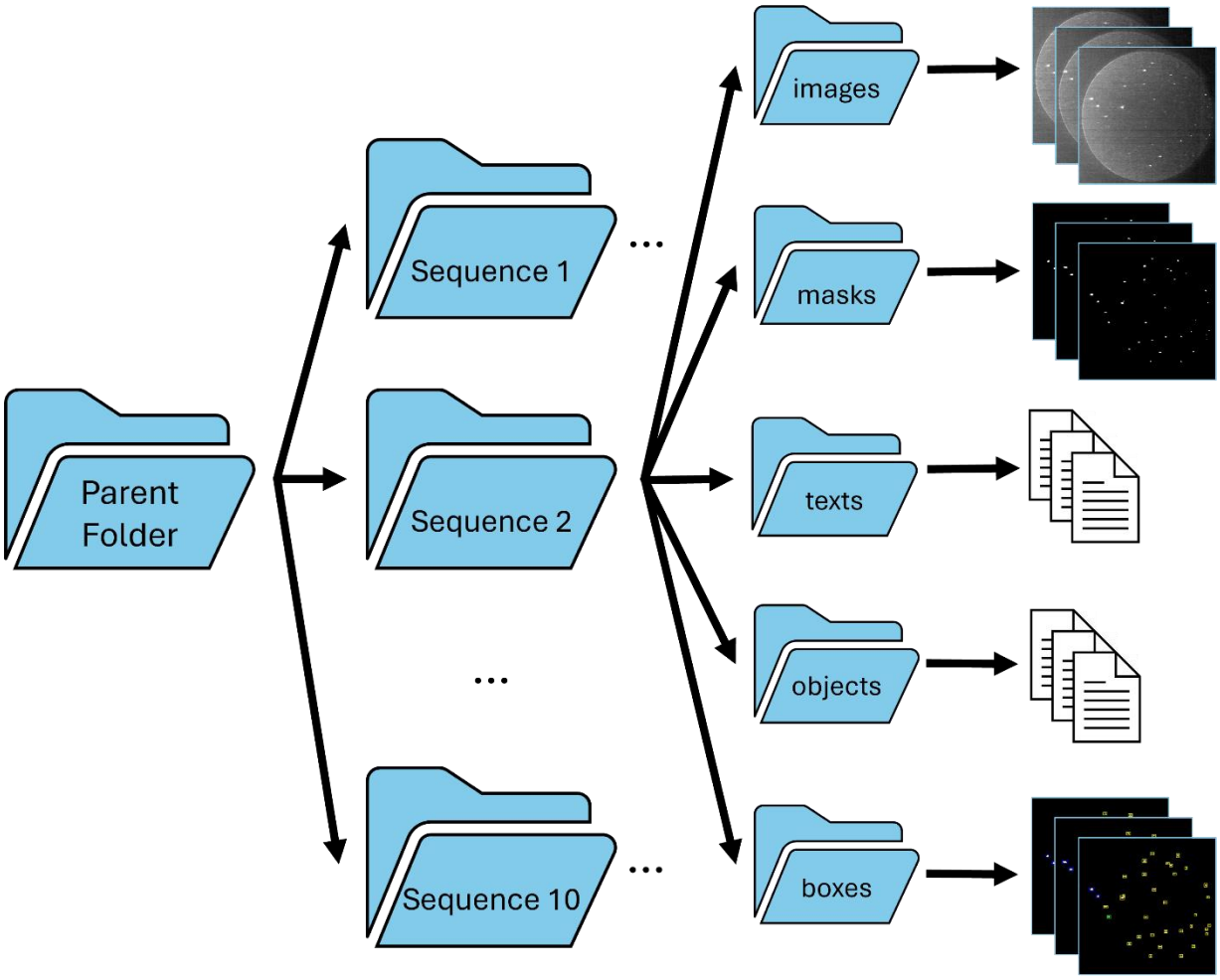

Figure 4.11. Illustration of the folder hierarchy and contents of the preliminary benchmark dataset developed in this study, using the developed tool suite.

After developing this benchmark dataset, it needs to be hosted on a publicly available platform for users to access. A training and testing fold consisting of 16 of the 20 sequences should be made available, while four of the 20 sequences should be held-out and used to validate algorithms. The benchmark dataset should be marked with a version, and further improvements to the dataset should be made and created as new versions.

These new versions could use data from additional imagers and missions to move towards a more generalized dataset. For example, a future version of the dataset could have long-exposure imagery added to it. Images captured from dedicated CubeSat missions and other missions with optical payloads could also be used for this purpose.

4.5. Algorithm Development

4.5.1. Algorithm Evaluation Metrics

Before considering the development of an algorithm to track RSOs, the relevant evaluation metrics need to be considered. It may be desirable to describe an algorithm's performance with a single score, which can help simplify algorithm comparison. However, different algorithms may excel in different metrics, which may be useful for different use cases. For example, an algorithm that does not incorrectly identify RSOs often, but also does not always detect RSOs may be preferred in cases where incorrect RSO detections can be severely detrimental to the mission. Additionally, some algorithms may not have certain metrics applicable to them, such as RSO detection algorithms, which may not be designed for tracking as well. For this reason, several carefully selected metrics are presented to thoroughly evaluate algorithms, while not complicating the comparison. Many of these metrics were also suggested in MOTChallenge, which seeks to evaluate tracking algorithms as well [81].

The first set of metrics are related to object detection. As mentioned, some algorithms may not have been developed to perform tracking or even classification, and simply seek to detect RSOs within them. For this reason, only the precision, recall, and F1 score are used, as defined in Equation 3.4, Equation 3.5, and Equation 4.1.

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4.1)$$

TPs are detections of RSOs made by an algorithm, that are indeed RSOs as confirmed by the truth data. FPs are detections of RSOs made by an algorithm that are not RSOs in the truth data. FNs are RSOs that exist in the truth data but have not been detected by the RSO detection algorithm. For the dataset presented in this study, the resolution of the images is low, objects are small, some detections may be a single pixel in size, and the labelling process itself does not consider a weighted centroid for detection. For these reasons, a TP is arbitrarily defined as a detection that falls within two pixels from the corresponding detection in the truth data. FNs represent the opposite case, where an object in the truth data does not have a corresponding detection made by the algorithm within two pixels. FPs are defined as detections made by the algorithm that do not have a corresponding true detection in the truth data within two pixels. Precision seeks to evaluate the "correctness" of an algorithm's RSO detections. In other words, it seeks to determine, of all the RSO detections made by the algorithm, how many of them were actually correct. Therefore, the metric considers the TPs and FPs alone, with fewer FPs resulting in a better score. This is an important metric to evaluate in use cases where false detections of RSOs is undesirable. Recall seeks to evaluate the "completeness" of an algorithm's RSO detections. In other words, it seeks to determine, of all the RSO detections in the ground truth data, how many of them the algorithm actually detected. Therefore, the metric considers the TPs and FNs alone, with fewer FNs resulting in a better score. This is an important metric to evaluate in cases where false detections of RSOs are acceptable, but the priority is to detect as many RSOs as possible. Naturally, there is a tradeoff between these two metrics. This is because an algorithm can be tuned to more "aggressively" make RSO detections, thereby increasing the

number of TPs and reducing the FNs, but potentially at the cost of more FPs. The opposite can be done as well. Algorithms with high precision and low recall are conservative in their predictions of RSOs and are focused on making correct RSO predictions. On the other hand, Algorithms with low precision and high recall are liberal in their predictions of RSOs and are focused on detecting as many RSOs as possible. The F1 score seeks to combine the precision and recall into a single metric, representing an algorithm's performance across precision and recall. The is an important metric to consider when a balance is needed between an algorithm's precision and recall. It is important to note that for these metrics, TNs have been excluded. TNs represent detections that were not identified as RSOs by the algorithm and were indeed not RSOs in the truth data. In the case of RSO detection, this may be ambiguous, since pixels not deemed as RSO by the algorithm and truth data can each be considered a TN, or the entire background can be considered a TN. In the former case, this would cause many TNs to be reported, since images are mostly background, overwhelming other useful metrics. Additionally, the most important task in this case is to focus on the RSOs, and so the background is of little concern.

The next metric criteria to focus on is object tracking, needed to quantify how well an RSO tracking algorithm is able to track RSOs across sequences of images. Two metrics are given here, called Multiple Object Tracking Accuracy (MOTA), defined by Equation 4.2, and Percent Detected (%Det), defined by Equation 4.3. MOTA was chosen for a variety of reasons. Firstly, MOTA captures a wide variety of errors in a single metric, making it a simple but expressive representation of an algorithm's performance with respect to tracking. It is a widely used metric for evaluating modern MOT algorithms [81]. Secondly, MOTA captures tracking performance from frame to frame, rather than for an entire sequence. Such an evaluation of tracking is

sufficient and desirable due to its simplicity. Some RSOs can disappear for a few frames and reappear, due to the RSO's attitude with respect to the observing imager. This makes annotations and algorithm design difficult and potentially ambiguous, since these RSOs would need to be carefully examined to ensure that a reappearing RSO is indeed the same RSO. Instead, MOTA is used to evaluate tracking performance due to its frame-to-frame method of capturing tracking errors. The % Det metric was developed to capture the number of RSOs detected at least once by an RSO tracking algorithm, providing insights into what sequences and specific RSOs offer challenges for algorithms. For future work, longer, identity-based measures may be considered, such as Identification Precision (IDP), Identification Recall (IDR), and Identification F1 (IDF1) [20]. Multiple Object Tracking Precision (MOTP) is also not considered here, given that the precision of the detections in terms of their respective locations in the images is already captured in the object detection precision metric provided above.

$$MOTA = 1 - \frac{\sum_f FN_f + FP_f + IDSW_f}{\sum_f GT_f} \quad (4.2)$$

$$\%Det = \frac{Number\ of\ Correct\ Unique\ RSOs\ Detected}{Number\ of\ True\ Unique\ RSOs} \times 100 \quad (4.3)$$

In Equation 4.2, f represents the frame index of a particular sequence. An ID Switch (IDSW) refers to the event where an object is incorrectly identified in a subsequent frame. That is, the ID assigned to the object in the second frame is not the ID assigned to the same object in the first frame. The research in [81] makes the important note that while it is desirable to have fewer IDSW, sequences with many objects will naturally have more IDSW. Therefore, it may be desirable to consider the relative IDSW, which considers the IDSW with respect to the recall.

This relative IDSW metric may be considered as part of the final benchmark dataset. Ground Truth (GT) refers to the total number of objects in a frame. In cases where the tracker makes more errors than there are number of objects in a particular frame, the MOTA value for that frame can be negative. In Equation 4.3, it is important to note that an RSO needs to be detected only once to count as a correct unique RSO detection. Again, it may be ideal to present this metric relative to recall, which will be considered in future work.

Processing-related performance metrics, such as per-fame execution time, power consumed, and memory consumed are not considered in this study. However, they are quite valuable to consider for implementation of these algorithms at the edge, in dedicated or secondary missions. Further work may consider these metrics, after baselining hardware to use to evaluate algorithms for these metrics.

4.5.2. Multiple Object Tracking

To develop RSO tracking algorithms, the problem is treated as a MOT problem, where RSOs can be multiple objects in an image, and need to be uniquely tracked across a sequence of images capturing their movement in time. The MOT problem is often separated into two distinct frameworks, tracking-by-detection, and tracking-and-detection.

4.5.2.1. Tracking-by-Detection

The tracking-by-detection framework separates detection and tracking into distinct functions. An independent detection algorithm, or detector first detects objects across multiple images. These detections are then provided to an association algorithm, or tracker, which seeks to find the

corresponding object matches from previous frame(s) to the new frame. This loop of detecting objects and then associating them continues until the end of a sequence is reached.

Multiple algorithms exist for the purpose of detecting these objects in images and can be used as the detector in the tracking-by-detection framework. As mentioned, algorithms such as source detection, derivative-based edge detection, template matching algorithms, and even spatial-feature-based ML algorithms can be used to extract all objects from the images used in this study [55,56,57,58,49]. The key to using these methods is to distinguish the objects in an image from the background, rather than trying to distinguish the objects from each other (which occurs in a later step). The custom thresholding algorithm in the previous research in Chapter 3, combining windowed multi-Otsu thresholding and CCA can also be used for this purpose. Ideally, these algorithms ultimately produce a list of objects for a particular image, with information tied to each object, such as its centroid coordinates, pixel width and height, and total pixel intensity. This information can then be handed off to the tracker.

Multiple algorithms also exist for the purpose of associating objects across images, developing and maintaining unique instances of them. The nearest neighbour algorithm is a simple association algorithm that can be used for this purpose. Objects in the current image that are closest to the objects in the previous image (in terms of Euclidean distance, or some other distance metric) are associated correspondingly. However, trackers using this algorithm need to consider special cases, such as multiple objects in the previous image being assigned to a single object in the current image. The Hungarian algorithm is another popular choice for associating objects, which seeks to solve the Linear Assignment Problem (LAP), where a one-to-one matching is found between two sets [100]. In this case, each set represents the objects in two

consecutive images. To use the Hungarian algorithm, a cost matrix needs to be created, where the rows of the matrix correspond to the objects in the previous image, and the columns correspond to the objects in the current image. Each individual entry of the matrix represents the cost of assigning the corresponding object from the previous image to the corresponding object in the current image. Various costs can be used for this purpose, such as the distance between objects and the differences in their intensities. The Hungarian algorithm minimizes this matrix, finding the optimal assignment between objects in the first image and objects in the second image. Implementations may need to be further augmented to handle cases where objects disappear from the frame, and when new objects appear in the frame.

More recently, data-driven methods have been used to perform data association. The survey in [101] provides a comprehensive review of these methods, and the multiple ways in which this can be done. The research in [102] developed a Deep Neural Network (DNN) that learned to associate objects by using cost matrices, such as the example presented above, as truth data. Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) networks have also been shown to be promising methods of associating objects in the field of particle tracking, which face many similar challenges to RSO tracking [103]. Another way data-driven methods have been used for data association is by learning features to use in generating costs for cost matrices, as defined above, to then pass to algorithms such as the Hungarian algorithm. Rather than hand-crafting cost functions, RNNs can be used to extract feature vectors from candidate associations between an object in the previous frame and objects in the next frame.

In the images in this research, the objects in an image are not just limited to RSOs, but stars and noise as well. In the tracking-by-detection MOT paradigm, a distinction needs to be made

between these objects, in addition to tracking them. To do this, track features should be extracted for each object track, and classification algorithms should be used to classify the features. Such track features can be the object's speed, the object's acceleration, and the coefficients of a polynomial that fit the shape of the track. The idea is that while the visual appearance of RSOs, stars, and noise may be difficult to distinguish in a single image, their motion over time can be used to tell them apart.

4.5.2.2. Tracking-and-Detection

The tracking-and-detection MOT paradigm tightly combines the detection and tracking process. The tracking process is usually used to help find new detections of previously detected objects in new data. The tracker may propose certain ROIs to consider based on existing tracks for detections, and the rest of the data can be ignored. Many methods that operate in this paradigm use the Kalman filter or variants of it [104]. The Kalman filter is used to estimate the new state of an object in two steps. First, it makes a prediction of an object's new state by using old state information and a motion model of the object. The Kalman filter then updates its prediction based on new sensor data, applying a weighting to the prediction as well as the measurement data, using uncertainties from each. Individual objects in images can have their own Kalman filter applied to them, but additional modifications need to be made to handle missed detections, new detections, false detections, and assigning the correct objects across images.

To aid the problem of assigning the correct objects to each other across images, probability-based methods can be used, such as the Joint Probabilistic Data Association (JPDA) method [105]. The idea is to find the best matching candidate by generating hypotheses for all new detections about how they can be associated with the predicted positions of existing objects. A

validation gate is used to ensure that only detections that are close enough to these predictions are considered. The probabilities are then calculated to represent how well each new detection matches the valid corresponding existing objects, also considering the possibility that the new detections are attributed to errors. Tracks generated using JPDA can consist of multiple weighted hypotheses, not just a single hypothesis. The low-probability hypotheses can be pruned over time to reduce computational time and complexity.

In the tracking-and-detection paradigm, a distinction may need to be made between objects that are being tracked as well, to define them as RSOs (which are of interest), stars, and noise. However, in this paradigm, this distinction can be leveraged early on so that only detections that are suspected to be RSOs are tracked across frames. Furthermore, the motion characteristics of RSOs can be leveraged into motion models for algorithms using tracking-and-detection.

4.6. Sample RSO Tracking Algorithm

In this research, the tracking-by-detection paradigm was used to develop an RSO tracking algorithm. This paradigm was used because it allowed for the distinct separation between the detection algorithm and tracking algorithm. This meant that either algorithm could be improved or substituted, without having an impact on the other algorithm, allowing for a comparison of various permutations of detector and tracker in future work, as well as modularity when implementing the RSO tracking algorithm in various scenarios. A custom implementation of the U-Net CNN [94] was used as the detector in this study, for several reasons. Firstly, a convolution-based method of detection was thought to prove superior to a thresholding-based method of detection, since while the background of the scenes might change, the objects

themselves will appear as point sources of light through images. Secondly, a Neural Network (NN) was chosen for this task so that the various shapes of the objects could be learned from the data, rather than hand-crafting templates for each object, which may prove to be both laborious and erroneous. For the tracker, both the Hungarian algorithm with hand-crafted cost functions, as well as a custom assignment algorithm were used.

4.6.1. CNN for Object Detection

A custom U-Net implementation was developed for this research, differing slightly from the original version. The input layer was set to accept 256-pixel by 256-pixel grayscale images. The contracting path consisted of five convolutional layers. Each convolutional layer consisted of a 2D convolution, a dropout layer, another 2D convolution, and a max pooling layer, save for the fifth and final convolutional layer, which did not have a max pooling layer. The contracting path served to extract deeper and deeper features from the data. The expanding path consisted of four transposed convolutional layers. Each transposed convolutional layer consisted of a 2D transposed convolution, a concatenation operation to combine the output with the corresponding layer in the contracting path, a 2D convolution, a dropout layer, and another 2D convolution. The output layer consisted of a single 2D convolution with a single 1-pixel by 1-pixel filter, using the sigmoid activation function, to generate the output probability map. Figure 4.12 gives a visual overview of the CNN.

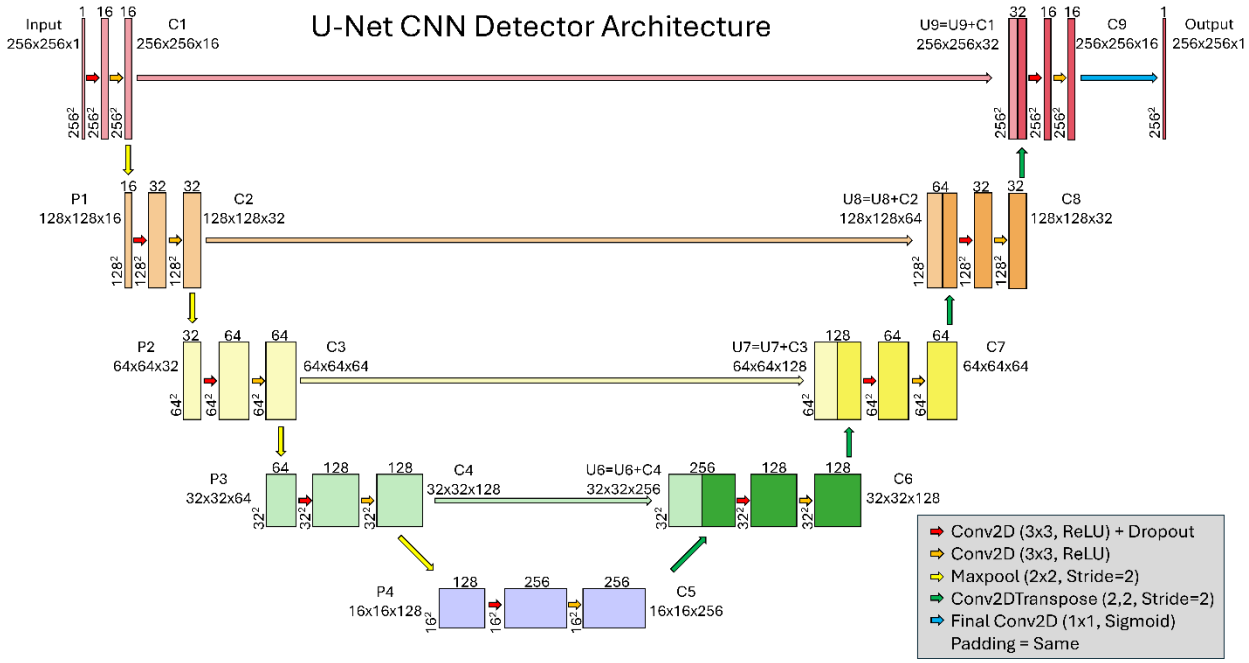

Figure 4.12. U-Net CNN architecture for detector developed for the RSO tracking algorithm.

In all 2D convolution operations, the Rectified Linear Unit (ReLU) activation function was used, save for the final 2D convolution used in the output layer. Padding was added after the 2D convolution operations to ensure the image size stayed the same. The first convolutional layer in the contracting path used 16 filters for both 2D convolutions. Following convolutional layers sequentially doubled the number of filters used for each corresponding 2D convolution, with 256 filters used for both 2D convolutions in the final layer of the contracting path. A similar scheme was used in the expanding path, starting with 128 filters used for the 2D transposed convolution and both 2D convolutions in the first 2D transposed convolutional layer. Following 2D transposed convolutional layers halved the number of filters, until 16 were used in the final 2D transposed convolutional layer. Each 2D convolution used a 3-pixel by 3-pixel filter. Max pooling layers used a pool size of two pixels. Dropout layers sequentially increased the percentage of dropped neurons in the contracting path, dropping 10% of neurons in the first and

second 2D convolutional layer, 20% of neurons in the third and fourth layer, and 30% of neurons in the fifth layer. In the expanding path, the dropout sequentially decreased, with 20% of neurons dropped in the first and second layer, and 10% dropped in the fourth and fifth layer. Adam was used as the optimizer, while the binary cross entropy function was used as the loss function, given that there were two classes (objects and background). The network consisted of 1.9 million learnable parameters.

The filters used in each layer and the number of layers were chosen somewhat arbitrarily. However, it is likely that there are far too many filters and layers than necessary. This intuition comes from the fact that RSOs, stars, and noise are simple, blob-like objects. They can likely be represented by far fewer filters than was used. The same can be said about using multiple layers, given that deeper layers capture deeper features, which may be unnecessary for the case of RSOs, stars, and noise. The number of filters and layers will be reduced iteratively, in future work.

4.6.2. Algorithms for Object Tracking

Two different algorithms were developed for the tracker component of the RSO tracking algorithm in this research. The first algorithm used the Hungarian algorithm, while the second algorithm used a priority-based assignment technique.

4.6.2.1. Hungarian Algorithm-Based Assignment Tracker

The first tracker was developed by using the Hungarian algorithm along with hand-crafted cost functions, with further modifications to handle new objects that have been detected, old objects

that have left the scene, and missed detections. The algorithm started by loading in the points detected from the first image and initiating tracks for all the objects.

In the case of the second image, the points were loaded in again, and a distance cost matrix was formed. The rows in this cost matrix represented existing tracks, while the columns represented new detections. The entries of the matrix were computed by calculating the Euclidean pixel distance between the current locations of the corresponding tracks and locations of the new detections. A maximum distance of 10 pixels was arbitrarily set, such that a Euclidean pixel distance greater than 10 pixels had an arbitrarily large cost.

In the case of the third image and all subsequent images, the distance cost matrix was replaced by an estimation cost matrix. The entries of the estimation cost matrix were calculated as follows. First, new positions were estimated for established tracks, using a linear motion model. The scenarios where the estimation is beyond the image borders and where there are too many estimations for a particular track were handled accordingly. Next, the Euclidean pixel distances between tracks and new detections were similarly calculated, using the track's estimated position rather than the track's last position. In this case, the cost for corresponding Euclidean pixel distances greater than 0.1 pixels was set arbitrarily high, while the cost for distances less than or equal to 0.1 pixels were set as 0, to represent perfect assignment. A function was used to prune high-cost rows, the idea being that the smallest cost of a row for any corresponding detection will still be large for cases where the track has terminated, with no new detections. A similar pruning scheme was done for high-cost columns, the idea being that detections with high costs for any corresponding track will likely be new detections.

In either case, the Hungarian algorithm was then used to find the best assignment of existing detections to tracks by finding an assignment of rows (tracks) to columns (new detections) in the corresponding cost matrix. Functions were added to update tracks, create new tracks, estimate missed tracks (that are thought to still exist, but were undetected by the detector), delete old tracks, and generate helpful visuals and text files.

4.6.2.2. Priority-Based Assignment Tracker

The second tracker was developed to simplify the assignment process and tackle some of the nuances experienced with the data. The algorithm was constructed similarly to a nearest neighbour algorithm, where well-established tracks would be associated to new detections first, followed by newer tracks. The idea is that well-established tracks are likely to have a detection within the next frame (save for when the detection has left the frame), while newer tracks may be falsely created tracks, which do not have new detections in the next frame. The algorithm again started by loading points for the first image and establishing tracks for them. The rest of the algorithm occurred in a loop for the rest of the image, regardless of the image number.

The algorithm then assigned detections to the well-established tracks, which are tracks that had at least two prior detections. This was done by first estimating the new position for each track, then looping through the estimated positions of those well-established tracks, and then finding the closest detection to each track, based on Euclidean distance. Again, a maximum distance was enforced. If no close detection was found, the estimated position of that track was kept as a detection but marked as an estimated detection rather than a true detection. Only tracks that consisted of true detections in the previous frame were estimated to prevent estimating false

detections multiple times. Detections that had been successfully assigned were deleted from the list of new detections.

The algorithm then assigned the remaining detections to newer tracks, following a similar assignment method as above. However, since the newer tracks could not be estimated, only the detections from their last frame were used to find the nearest corresponding new detection.

In the case that there were still new detections left over after assignment, the algorithm created new tracks out of them. Additionally, tracks that could not be estimated or left the image boundary were archived. The algorithm ended by generating visuals and text files for review.

4.6.3. Object Classification

As mentioned previously, classifying the tracked objects into RSOs, stars, and noise is necessary, since RSOs are of interest, and can only be distinguished based on their motion characteristics, which is different than that of stars and noise. While this was not completed in this research, future work will seek to perform this classification by extracting track characteristics, such as the average velocity of the object, and the coefficients of the quadratic that models the shape of the object's track.

4.7. Preliminary Algorithm Results

To offer preliminary insight into the algorithm's performance, several results are presented, of which most are qualitative. The purpose of the developed algorithm is to offer a means of testing the generated preliminary dataset and to offer a paradigm of RSO tracking algorithm development, rather than to push the current state-of-the-art.

4.7.1. Object Detection

To train and test the CNN detector, the entire 500-image preliminary dataset was used. A test-train split of 80%-20% was used, where Sequences 1 to 8 were used for training for a total of 400 training images, while Sequences 9 and 10 were used for testing for a total of 100 testing images. Of the 400 images, a validation split of 90%-10% was used for training and validation, using 360 images and 40 images each. A batch size of 16 was used, and the training period was set for 25 epochs. An early-stop function was used to stop training in the case that the CNN's accuracy stagnated, to prevent overfitting. Figure 4.13 illustrates the model's training loss for each epoch, while Figure 4.14 illustrates the training accuracy. Given the stagnation in training accuracy, the model only trained for 11 epochs. Figure 4.15 provides examples of the model's output probability maps thresholded at a probability of 0.5, as well as the corresponding images and masks. The model achieved a final test loss of 0.018, and a test accuracy of 99.34%.

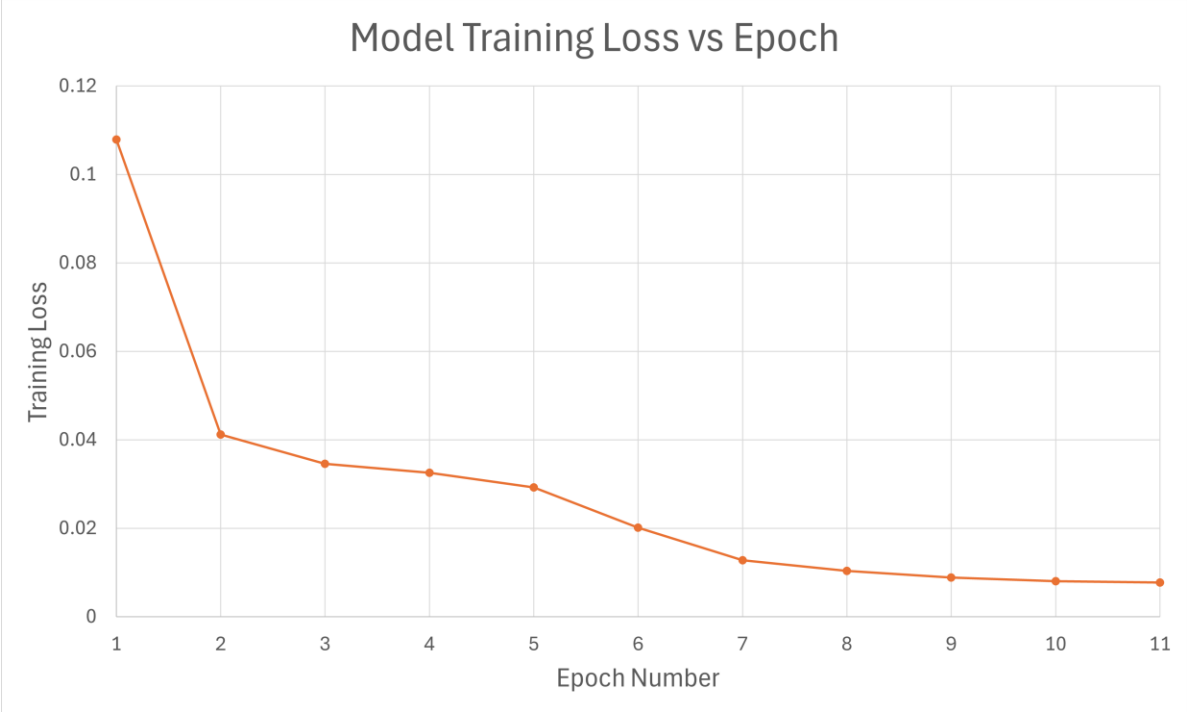

Figure 4.13. Custom U-Net CNN's training loss, per epoch.

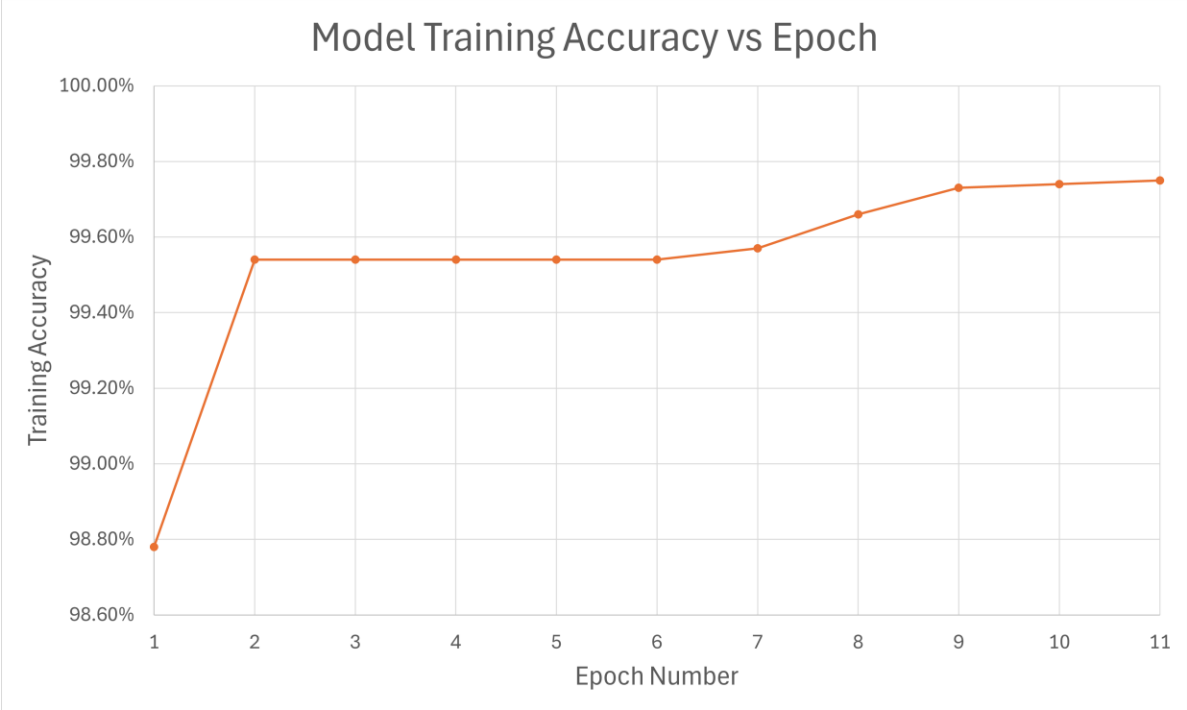

Figure 4.14. Custom U-Net CNN's training accuracy, per epoch.

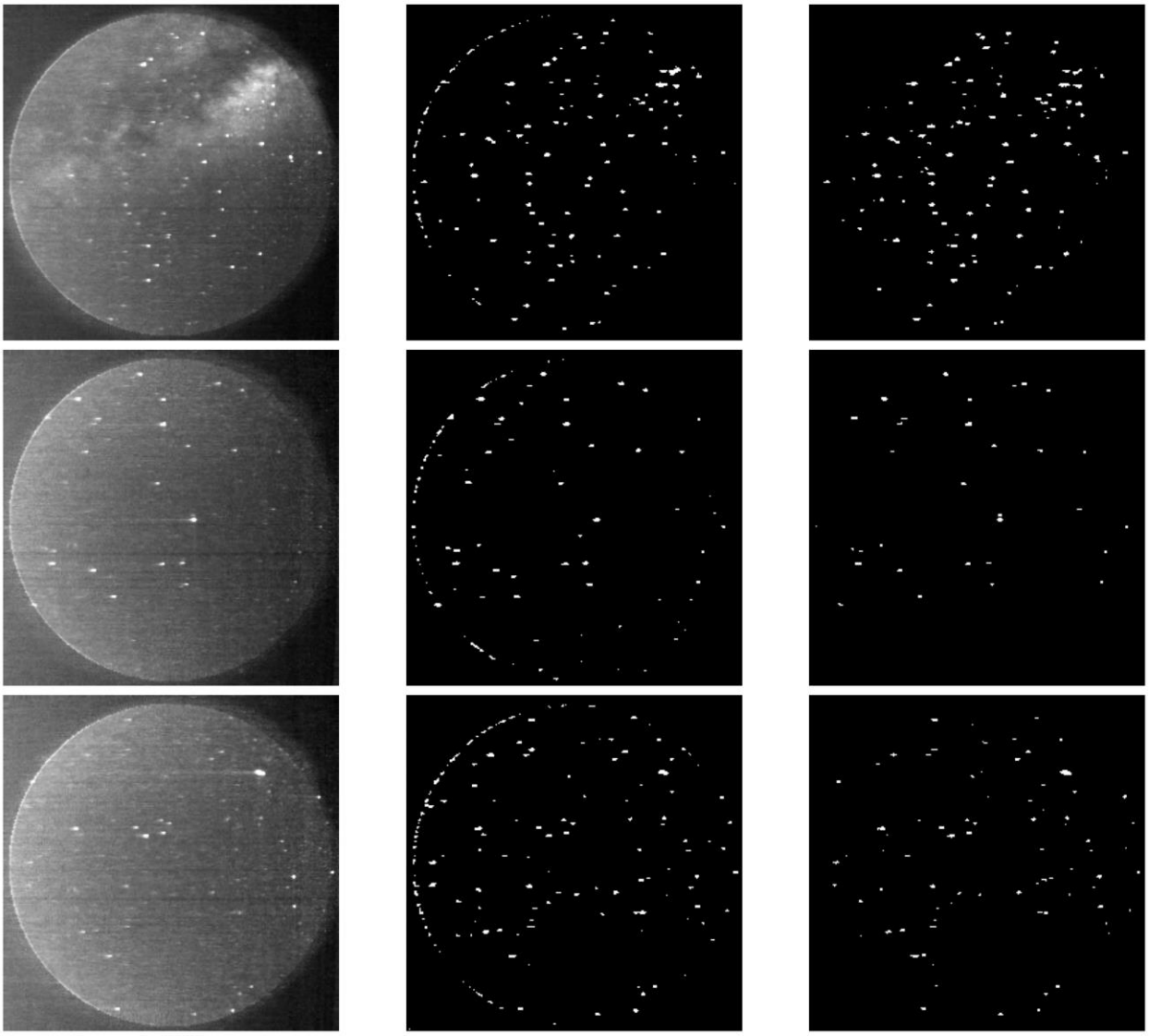

Figure 4.15. Output probability maps thresholded at 0.5 (center), along with corresponding images (left) and corresponding masks (right).

A probability threshold of 0.5 was found to be the ideal value for balancing between capturing faint objects in the images and generating false detections. However, this may be indicative of the fact that the model was struggling to properly separate detections from the background with high confidence. The model ended up stopping training after 11 epochs, which indicates that the training had stagnated by the 11^{th} epoch. Given that the training loss was quite low, and accuracy

was quite high at this point, it can be said that the CNN had sufficiently learned enough from the data, such that 11 epochs were enough to separate foreground objects from background. The high accuracy indicates that the model is able to predict each pixel as foreground or background correctly very often, while the low loss suggests that, of the incorrectly predicted pixels, the prediction itself was not far off from the actual value. Another important consideration is the class imbalance. As mentioned previously, on average, images are 99.4% background. The accuracy and loss of the model may be significantly overrepresented and underrepresented respectively because of this, since the model is incentivized to predict the background class more often, even though the objects in the images are of most importance. Thresholding the predictions at lower and lower values can help overcome this and may be why 0.5 is a good threshold to use. Furthermore, this may also suggest that further training may be occurring at later epochs, as the model increases its accuracy, such as during epochs seven to nine. For this reason, it may be worth removing the early-stopping function. Alternatively, it may be worth leveraging a different loss function, which weighs the background class less heavily than the foreground class, or vice-versa. Lastly, it may be the case that the model is far larger than necessary, both in terms of filters and layer depth, given that the training accuracy reaches almost 99% within the first epoch alone. Many of the filters and layers may be redundant, and so iteratively and experimentally removing them will be done in future work.

In terms of the qualitative results, the model appears to be learning detections quite well. Even in the case of the image with the Milky Way, the model is able to separate objects within the illumination of the Milky Way. In some cases, the model appears to be even more sensitive than the truth data, as it is able to pick up very small objects that were not even annotated. However,

the model does classify the illumination around the edge of the imager's FOV as a foreground object. This raises an important issue on the subjectivity of the noise class, since the ring around the imager's FOV could be considered as background by some, or as noise by others. Whatever the case, it is important not to severely penalize false detections, given the mentioned ability of the model to capture fainter objects than in the truth data and the mentioned ambiguity.

4.7.2. Object Tracking

Given that the tracking labels for the preliminary dataset are yet to be developed, only qualitative results are shown for the trackers designed in this research. The trackers were tested separately from the detector to isolate the evaluation of each tracker independently of the detector. Therefore, the trackers were directly given the detections from the truth data. Figure 4.16 shows an example of the tracker's visual output from a sequence of four images from Sequence 1 of the preliminary dataset, for each tracker.

Both tracking algorithms work well on the truth data, and are able to uniquely track stars, RSOs, and noise across the images, based on the visual outputs produced across the sequences. However, the Hungarian algorithm-based tracker struggled with ID switches in certain cases where a new track should have been initiated instead. In Figure 4.16, this can be observed in that algorithm's visual output for the third image (frame 14), where a new object, likely a noise detection, appeared right below object two. Instead of initiating a new track, the algorithm incorrectly identified the object as an extension of track four, which had a cascading ID switching effect across the other objects in the scene. This was likely because the new object was close enough to surrounding objects that it passed the distance minimum threshold. The

corresponding output for the priority-based tracker did not suffer from this problem for this case and was able to correctly identify the object as a new object, giving it the ID 44. Extensive testing and qualitative results based on the metrics mentioned above will be conducted in the future.

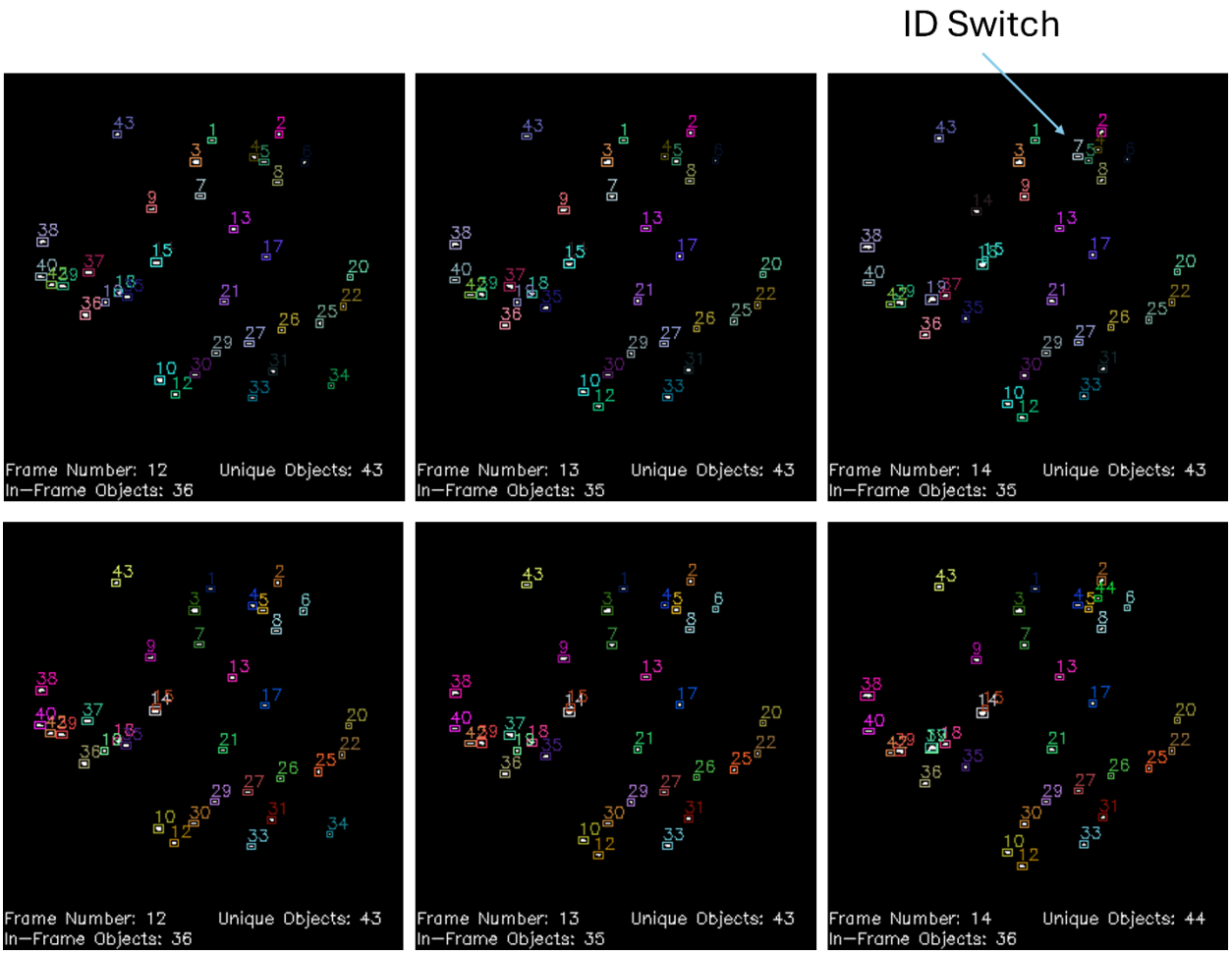

Figure 4.16. Visual example of the performance of the Hungarian algorithm-based tracker (top) and the performance of the priority-based tracker (bottom) across four images from Sequence 1 in the preliminary dataset. An ID switch error is illustrated for the third image (frame 14) of the Hungarian algorithm-based tracker.

4.8. Future Work

In terms of the data generation tools, several improvements and additions need to be made. Firstly, the ability to track objects and create unique identities for them needs to be implemented. To do this, the same automation scheme can be applied, where a tracking algorithm such as the one developed in this research can be used, along with a method of manually editing the object identities. For larger images, it may also be beneficial to add zoom and pan functionality, given that some higher resolution wide FOV imagery may have very small RSOs that may be difficult to label. Once the tools have been augmented with tracking label generation, the 500-image dataset can be revised with tracking labels. It is also recommended to add more sequences that are difficult to process, such as sequences with the Earth's limb in the FOV and sequences with severe particle strike effects. In terms of the developed algorithms, the CNN needs to be reduced in size, needs to have a different loss function used that captures the class imbalance, and needs to be thoroughly tested. The tracking algorithm also needs to be thoroughly tested, especially in terms of MOTA with generated tracking labels. Finally, the algorithms, dataset, and data generation tools need to be iteratively improved over time. As developed algorithms improve, they need to be incorporated into the automation tools to reduce annotation time and improve annotation quality, consequently improving the generated annotation data, in a positive feedback loop.

4.9. Conclusion

In this research, an automated data annotation framework and corresponding data annotation tools were developed for the purpose of generating a benchmark dataset of unresolved RSO SSA

imagery. The developed tools were used to generate a preliminary annotated 500-image dataset. The dataset was explored in terms of the distribution of the number of objects and pixels per class. A tracking algorithm consisting of a semantic segmentation-based CNN paired with either a Hungarian algorithm-based assignment tracker or priority-based assignment tracker was developed. Preliminary qualitative algorithm outputs showed promising results, proving the feasibility of such an automated labelling framework, benchmark dataset, and tracking algorithm.

5. A Dual-Purpose Camera for Attitude Determination and Resident Space Object Detection on a Stratospheric Balloon

This chapter is based on a paper [106] describing a payload, namely Star Tracker Attitude and RSO Detection for Unified Space Technologies (STARDUST). This payload was developed to demonstrate the concept of a dual-purpose payload, which could simultaneously perform AD and RSO detection. Such a payload concept would be an efficient way to perform space-based RSO detection, since many spacecraft are already equipped with star trackers. The payload was demonstrated onboard a stratospheric balloon, as part of CSA's Strato-Science 2023 campaign. A custom RSO detection algorithm, derived from research carried out from previous chapters, was developed for this purpose. Several modifications were made to enable real-time performance, as well as to adapt to the imaging conditions of a stratospheric balloon platform. Since the gondola of the stratospheric balloon would be stabilized relative to the background stars, the ICP algorithm used in the research in Chapter 3 was removed. Instead, a modified linear motion model was implemented, allowing for slight movements from background stars to be accounted for. Another modification was the implementation of an adaptive threshold, which would increase or decrease the threshold in real-time depending on the number of sources detected in the previous image. The AD and RSO detection algorithms were successfully demonstrated during the flight in August 2023. Although the STARDUST camera experienced degradation, 11 unique RSOs were identified in real-time by the onboard algorithm, corresponding to 669 detections. Both the AD algorithm and the RSO algorithm ran with a per-iteration processing time of 502 ms, including the 100 ms exposure time and health data saving. This research proved

that a dual-purpose star tracker to perform AD and RSO detection is a feasible concept and is a step towards a space-based demonstration.

In this chapter, sections that I did not write in [106] were rewritten, with custom figures used. My part in this research mainly involved RSO detection, and so the emphasis in this chapter is placed on the RSO detection algorithm and results, as well as payload development, parts selection, and general mission results. AD research is described thoroughly in [106] and is referenced as needed.

I developed the real-time RSO detection algorithm, the main flight software (including boot-up operations, camera initialization, image capture functions, and health data saving), and the parts selection trade study. Alongside Gabriel Chianelli, I was involved in the assembly of the payload, the planning of the mission, thermal and vacuum testing, field campaigns, payload health monitoring during the mission, and post-mission operations. I was a member of the crew that carried out these tasks to successfully oversee the STARDUST payload's launch, along with the other payloads, on a stratospheric balloon from Timmins, Ontario, in August 2023. I prepared all figures and tables in this chapter, wrote the sections pertaining to RSO detection section, and rewrote the rest of the sections. Gabriel Chianelli and Marissa Myhre developed the AD algorithm and helped integrate it into the main flight software. Regina S. K. Lee created the metrics for the payload, developed documentation for the design of the mission, gave feedback after testing and field campaigns, and provided supervision throughout the entire research project.

I would like to give special thanks to Randa Qashoa, Vithurshan Suthakar, and Hyunbin Yim for their advice, expertise, and help during the testing of the STARDUST payload.

5.1. Introduction

With rapidly increasing interest in space, the total number of RSOs around the Earth also continues to increase [5]. This increase in RSOs corresponds to an increase in the probability of collision, which places great importance on having a well-defined understanding of these objects. This includes knowledge of RSO orbital parameters and characteristics, an understanding often referred to as SSA. Typically, activities for SSA are often conducted using technologies on the ground and in space, using dedicated instruments and payloads. Examples of these SSA technologies can be found in the United States SSN, which is the most comprehensive, open source SSA system [107]. The Ground-Based Electro-Optical Deep Space Surveillance (GEODSS) System is an example of a combination of ground-based systems in the SSN. GEODSS keeps track of deep space objects at an altitude of 10000 km to 45000 km using multiple one-meter telescopes [108]. An example of a space-based SSA system is the Space Based Space Surveillance (SBSS) System. This system currently consists of a satellite in Sun-Synchronous Orbit (SSO), designed to detect small objects all the way out to GEO [109].

While the SSN is a well-developed, comprehensive system, the increasing number of RSOs, especially in LEO can strain the system. Furthermore, the publicly available catalogue produced by the SSN currently does not contain small objects, and some objects are even lost over time [110,12]. Greater SSA capabilities are necessary to solve these issues and keep up with the exponential growth of RSOs to ultimately allow for the safe, sustainable use of space.

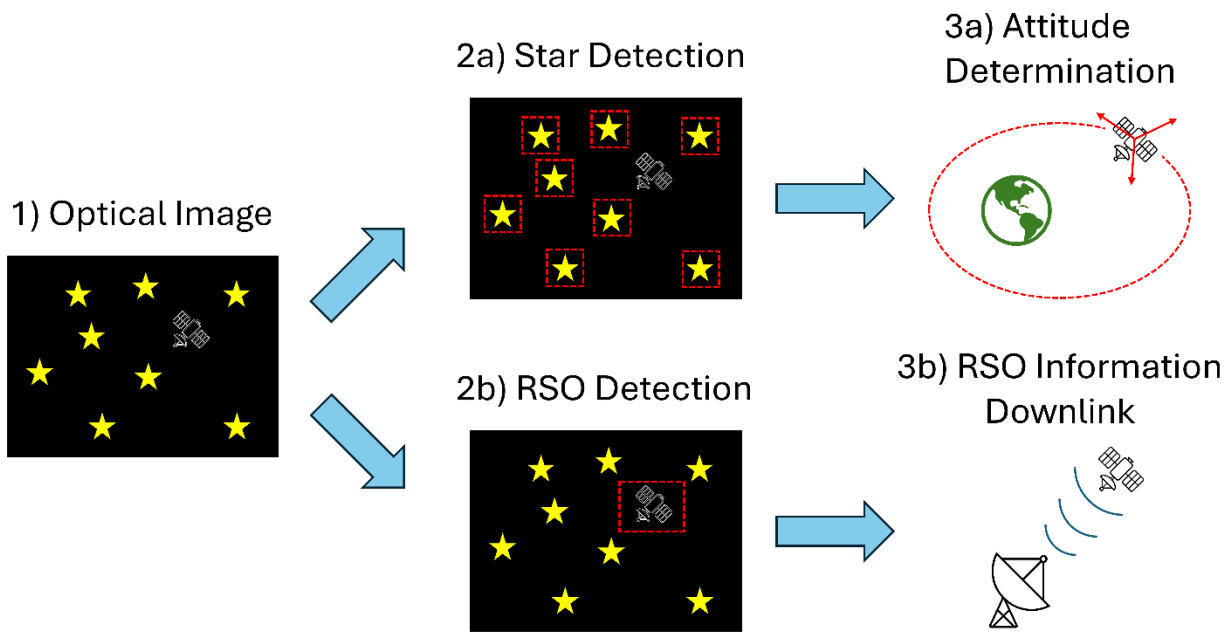

Figure 5.1. Illustration of a dual-purpose payload, where the primary purpose of the payload is to determine the host satellite's attitude, while the secondary purpose is to detect RSOs to improve SSA.

One promising idea is the implementation of a dual-purpose payload. The idea is to conduct space-based optical imaging for SSA as a secondary or parallel operation to the primary use of the onboard optical sensor. As an example, many spacecraft are equipped with star trackers, optical sensors used to determine the spacecraft's attitude by using the background stars [111]. Theoretically, the images that are captured and sent to the AD algorithm could also be sent to an RSO detection algorithm. This RSO detection algorithm could process the images for RSOs either simultaneously or in series with the AD algorithm. Such an algorithm could filter captured images for those only containing RSOs in them or keep processed detection information alone for efficient storage and downlink. This dual-purpose concept, illustrated in Figure 5.1, would be an efficient way of conducting RSO detection for SSA since dedicated payloads do not need to

be launched for this purpose. Moreover, equipping existing satellites with RSO detection capabilities would massively expand current SSA systems. Additionally, RSO detection as a secondary purpose can even provide the host spacecraft with awareness of nearby RSOs, which can alert operators faster and even allow for autonomous maneuvering for collision avoidance [112].

In this chapter, a dual-purpose AD and RSO detection payload for a technology demonstration mission is described. The payload was flown as part of the second iteration of Resident Space Object Near-space Astrometric Research (RSONAR II), a stratospheric balloon mission to demonstrate technologies to improve SSA [113]. The flight occurred in August 2023, during the CSA's Strato-Science 2023 campaign, part of the CSA's STRATOS program [114,115].

5.1.1. Dual-Purpose Star Tracker for RSO Detection

Many of the existing optical systems used to detect RSOs are narrow FOV telescopes. One such system is NASA's Eugene Stansbery Meter-Class Autonomous Telescope (ES-MCAT) [116]. This system optically images all orbital altitudes, with a FOV of 0.68^{o} by 0.68^{o} [116]. On the other hand, star trackers are wide FOV devices, with FOVs often in the 10s of degrees [31]. This wider FOV ensures that multiple stars can be observed in a single image to perform AD. Using these wide FOV star trackers for RSO detection is not a new concept. Feasibility studies have been conducted to determine the suitability of star trackers for RSO detection and have shown that these devices can detect multiple RSOs per orbit of the host spacecraft, depending on the orientation of the star tracker [31,117]. Image processing and orbit determination of RSOs have also been done with star tracker-like images, proving that preliminary orbit estimates of RSOs

can be found using star trackers [49,118]. The aim of this research is to prove this dual-purpose star tracker for RSO detection by developing such a payload and conducting a technology demonstration mission.

Simultaneously performing AD and RSO detection using the same payload comes with many challenges. On their own, star tracker algorithms and RSO detection algorithms can be quite computationally expensive. Improvements have been made over time to AD algorithms but have traditionally been too computationally expensive for smaller payloads, such as CubeSats [119,120,121]. RSO detection algorithms have similar requirements and are usually used on ground-based systems where computational power is plentiful and real-time performance is unnecessary. [18,122]. The physical characteristics of the star tracker camera system and algorithm design both need to be carefully considered when developing such a payload. Moreover, the algorithms themselves have competing requirements. As an example, AD algorithms generally use shorter exposure time images from the star tracker, while longer exposure time images are sometimes preferred for RSO detection algorithms to make use of streak-based detection methods.

5.1.2. Research Overview

In this research, a dual-purpose payload for AD and RSO detection is demonstrated onboard a stratospheric balloon. The hardware description, software description, image collection campaigns, and results from this mission are described. The parts selection process describes the parameters considered for each of the parts used in the payload, including the sensor, lens, and computer, with a focus on COTS components. The software description provides a thorough

overview of the RSO detection algorithm as well as general mission-related software functions. A thorough description of the AD algorithm is omitted and can be found in [106]. The image collection campaigns are discussed as well, including the experiments conducted to determine the most desirable settings to use for the camera and algorithms. Lastly, the results from the stratospheric mission and conclusions are presented, with emphasis placed on lessons learned for future missions. Again, AD results are omitted from this chapter and can instead be found in [106].

5.2. STARDUST Payload Overview

5.2.1. Hardware Description

The hardware selected for the STARDUST payload considered metrics such that they would resemble a star tracker, while being of small form factor and cost effective to fit the constraints of the mission. A wide FOV lens was needed for this purpose, as well as a sensor capable of dim-light imaging at short exposure times.

In terms of sensors, the Raspberry Pi High Quality camera, Alvium 1500 C-500m camera, and IDS UI-3370CP-M-GL Rev. 2 camera were considered based on their price points, availability, and compact form factor [123,124,125]. To further understand their suitability for dim-light imaging, their quantum efficiencies, pixel sizes, resolutions, noise characteristics, and exposure time limits were compared. Ultimately, the IDS camera was selected for its excellent low-light performance, even at short exposure times in the 100s of milliseconds. The IDS UI-3370CP-M-GL could capture images at a high resolution of 2048 by 2048 pixels, which could be

subsampled at factors of 2, 4, 6, and 8. The sensor used was the 1” CMV4000-3E5M sensor, consisting of large 5.5 µm pixels.

In terms of lenses, C and CS-mount lenses designed for the Raspberry Pi camera series were considered to compensate for the cost of the IDS camera. The 6 mm wide angle lens, 16 mm telephoto lens, and 25 mm lens were considered [126,127,128]. Ultimately, the 16 mm telephoto lens was chosen, given its large aperture to allow for better light throughput and the large FOV of the configuration, at 40 degrees by 40 degrees. While this FOV was much larger than traditional star trackers, the group wanted to test the AD and RSO detection algorithms at such extremes.

In terms of the computer, the Raspberry Pi Zero 2 and Raspberry Pi 4 Model B were considered, chosen for their well-developed OS and support [129,130]. While the Raspberry Pi Zero 2 was the more desirable choice due to its small form factor and low power draw, the algorithms demanded much higher computational resources than the Zero 2 could offer. For this reason, the Model B was selected, in the 8 GB ram configuration. Along with the quad core Cortex-A72 processor running at 1.5 GHz, the board could run both algorithms. An overview of the selected parts is shown in Figure 5.2.

Figure 5.2. Components selected for the STARDUST payload, consisting of the IDS UI-3370CP-M-GL Rev. 2 camera, 16 mm telephoto lens, and Raspberry Pi 4 Model B 8 GB [125,127,130].

In terms of the complete system, power delivery was accomplished using a power line from the full RSONAR II payload's PDU. The PDU converted the variable 24-36 VDC provided by CSA's gondola batteries to the 5 V needed by the Raspberry Pi. Further details regarding the power harness, DC-to-DC converters, wiring, and followed specifications can be found in [106]. Two thermocouples were also used to record environmental temperature values, paired to MAX31855 amplifier breakout boards [131,132]. The Serial Peripheral Interface (SPI) was used to communicate with the sensors to retrieve temperature values.

5.2.2. Software Description

The software of the STARDUST payload was designed to initiate autonomously on power on. After ensuring that a power out did not occur, the camera and code were initialized, performing functions such as setting the camera's parameters and initiating counters.

The code then captured an image, extracted the centroids of the objects in those images, and saved the health and temperature sensor data, for two iterations. On the third iteration, the Lost-in-Space (LIS) AD function was executed first, followed by the capturing of the third image and centroid extraction. With three images loaded into memory now, RSO detection was possible and was performed with the RSO detection algorithm. Afterwards, the time elapsed since the LIS AD function, as well as its return type (successful or unsuccessful) were used to determine which attitude function to use. If the time elapsed since the LIS AD function was greater than five minutes, or if the function did not execute successfully, the function was executed again. Otherwise, the tracking AD function was executed. Regardless of the AD function used, the health and temperature sensor data were stored again before the next iteration. The next

subsections serve to further explain each of the functions. Figure 5.3 shows the software block diagram for the STARDUST payload, visually outlining the functions.

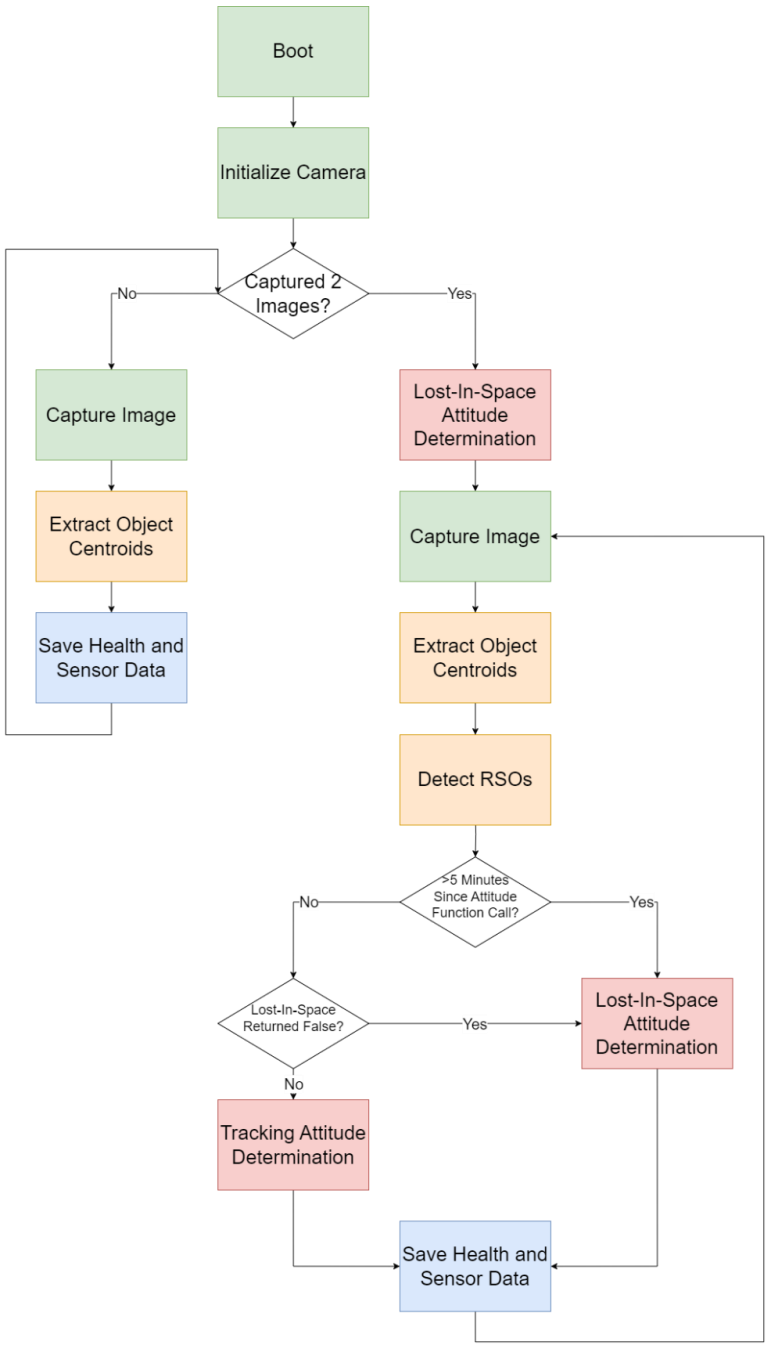

Figure 5.3. Software block diagram for STARDUST, visually outlining the functions and logic.

5.2.2.1. Initialize Camera Function

This function was used to set the camera's parameters, including the resolution (2048 by 2048 pixels), exposure time (100 ms), gamma (1.0), gain (100), and pixel clock frequency (100 MHz). This function also recorded the boot up time and ensured that a power out did not occur. If a power out did occur, this function forced a software reboot to ensure that the entire system resets properly, avoiding issues such as driver errors.

5.2.2.2. Capture Image Function

This function captured an image with the camera, converted the data into an array, logged the image name using the onboard date, time, and counter, saved the image with these details, and incremented the counter.

5.2.2.3. Extract Centroids Function

This function was the first step in the RSO detection algorithm. It extracted the centroids of all the objects (stars, RSOs, noise) in an image, and converted them to x and y coordinate pairs to be analyzed by the next step of the RSO detection algorithm. Further details are provided in Section 5.3.

5.2.2.4. Detect RSOs Function

This function was the second step in the RSO detection algorithm. It took in three sets (from three sequential images) of x and y coordinate pairs and identified unique x and y coordinate pairs that satisfied the conditions to be considered an RSO. This function also served to save x and y coordinate pairs to the SD card. Further details are also provided in Section 5.3.

5.2.2.5. Lost-In-Space Attitude Determination Function

This function served to determine the attitude results in LIS mode. More details can be found in [106].

5.2.2.6. Tracking Attitude Determination Function

This function served to determine the attitude from prior AD results. More details can be found in [106].

5.2.2.7. Save Health and Sensor Data Function

This function was used to save the health data reported by the Raspberry Pi OBC (the frequency of each CPU core and the CPU temperature) as well as the temperature reported by the connected temperature sensor.

5.3. RSO Detection

RSO detection is a critical part of the SSA pipeline, and involves finding RSOs within a sequence of images, distinguishing them from stars and noise. The RSO detection algorithm for this mission was developed to detect RSOs in real-time in a rolling window of three starfield images received from the camera. Several simplifications and assumptions were made when designing the algorithm. Firstly, it was assumed that the background stars captured in the images would not appear to move more than one pixel between each image (the movement of which would primarily be from the gondola's sway). Next, it was assumed that RSOs would appear to be travelling mostly linearly across the FOV of the imager. Finally, it was assumed that RSOs

would travel the same amount of distance across the FOV of the imager, between each image that captured the RSO (equidistance).

The algorithm works in two main steps as described in Section 5.2. The first step consists of extracting the centroids of the objects in each image, while the second step consists of detecting RSOs across three sets of these centroids.

5.3.1. Extracting Centroids

In the first step, the algorithm begins by binarizing the image by applying a simple threshold to the image. CCA using 8-pixel-neighbourhood connectivity is then performed to uniquely segment each of the objects in an image [68]. This algorithm returns details for each segmented object, most notably the x and y subpixel locations of each object's centroid and the pixel size of each object. This list of centroids is then filtered to remove objects deemed too small or large to be considered stars or RSOs, such as illumination effects or hot pixels. The size of this point list (which corresponds to the number of detected objects) is then analyzed to determine an appropriate threshold to use for the next iteration. If there are too few objects, the threshold is reduced, thereby allowing dimmer objects to be picked up by the algorithm. If there are too many objects, the threshold is increased, having the opposite effect. This point list (and two more point lists, corresponding to a sequence of three images) is passed on to the next step of the algorithm. Figure 5.4 shows the block diagram outlining the steps of the extracting centroids step.

5.3.2. Detecting RSOs

In the next step, after ensuring that the three point lists from the previous step contain data, the three point lists are looped through. For each set of unique (one point from each point set),

unmatched set of three points, the Euclidean distance is calculated between the point belonging to the first image and second image, d_1, and again calculated between the second image and third image, d_2. These distances are then checked to ensure they are far enough to be considered RSOs, as defined in the assumptions mentioned previously. Equation 3.1 is then used to determine how similar these distances are $d_{similarity}$, followed by Equation 3.2 to determine the angle between the vectors, θ (where d_1 is the vector from point one to point two, and d_2 is the vector from point two to point three). These two calculations are done to fulfill the linear and equidistant RSO assumption.

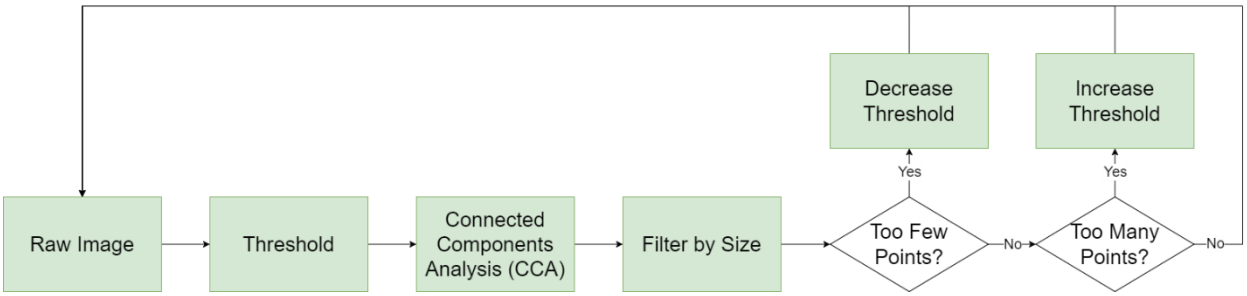

Figure 5.4. Block diagram outlining the first step, extracting centroids, in the RSO detection algorithm.

From experimentation with using this formula on existing optical RSO imagery, it was found that the angle was related to the distance similarity by Equation 5.1. This equation also provides the maximum angle, θ_{max}, below which the angle is deemed small enough to consider the triplet of three points as a roughly linearly moving RSO.

$$\theta_{max} = 39 d_{similarity} - 8 \quad (5.1)$$

The triplet of three points is then marked as matched, and the next set of points are analyzed until all points have been considered. The function then saves these RSO points to the OBC's SD card. Figure 5.5 below shows the block diagram outlining the steps of the detecting RSOs step.

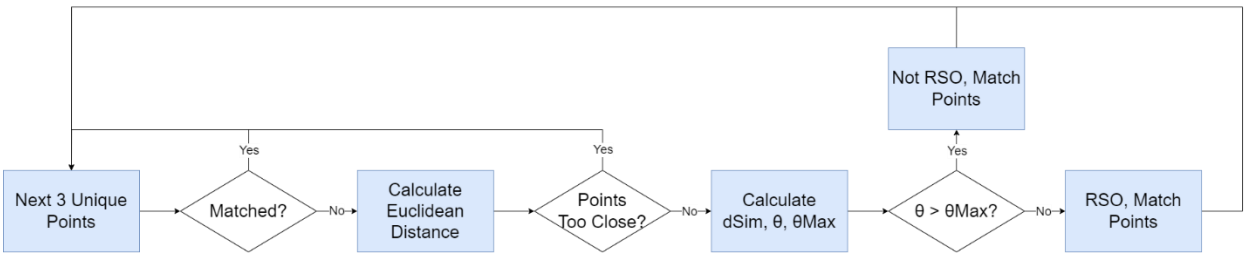

Figure 5.5. Block diagram outlining the first step, detecting RSOs, in the RSO detection algorithm.

5.4. Attitude Determination

AD is done to determine a spacecraft's orientation with respect to some fixed coordinate system. Usually, the attitude is determined with respect to the background stars, which do not tend to move with respect to a distant observer. This is done by finding the stars and their positions in an image and comparing them to an existing onboard star catalogue. This information is crucial and necessary to perform operations such as orienting a spacecraft's solar panels towards the sun or pointing an onboard antenna towards a ground station on Earth [133]. The attitude determined by star trackers is considered the highest accuracy relative to other AD sensors [134]. Star trackers generally have two modes of function: LIS mode, which determine the attitude without prior information, as described above, and tracking mode, which tracks the movement of the stars across images, in time, to determine the attitude. Both functions were implemented for the dual-purpose payload, and further details about the implementation can be found in [106].

5.5. Field Campaigns

To optimize camera parameters, test some of the sensors mentioned previously, and fine tune the AD and RSO detection algorithms, multiple field campaigns were conducted. Parameters such as the exposure time, binning factor, and gain were adjusted in permutations, and their effects on image quality were visually inspected.

The first field campaign occurred in King City, Ontario, after dusk. This location was chosen for the relatively low light pollution in comparison to nearby urban areas. Both the Raspberry Pi HQ camera and IDS camera, each paired to the 16 mm lens, were tested, varying the parameters above. Figure 5.6 is an example image captured from the IDS camera, using an exposure time of 500 ms. While not ideal for AD, this exposure time was used to test the visibility of stars and RSOs. In the yellow circles are examples of stars, and in the red box is an example of an RSO, which appears to streak through the image due to the longer exposure time. While the IDS camera was able to see many stars and RSOs in its FOV, the Raspberry Pi HQ camera could only see a fraction of the stars, at an exposure time of 30 s. Such an exposure time would be far too large to use during a stratospheric balloon mission, given that the gondola motion, even while stabilized, would cause streaking of stars and RSOs within the images. For these reasons, the Raspberry Pi HQ camera was abandoned, and the IDS camera was selected due to its proven performance.

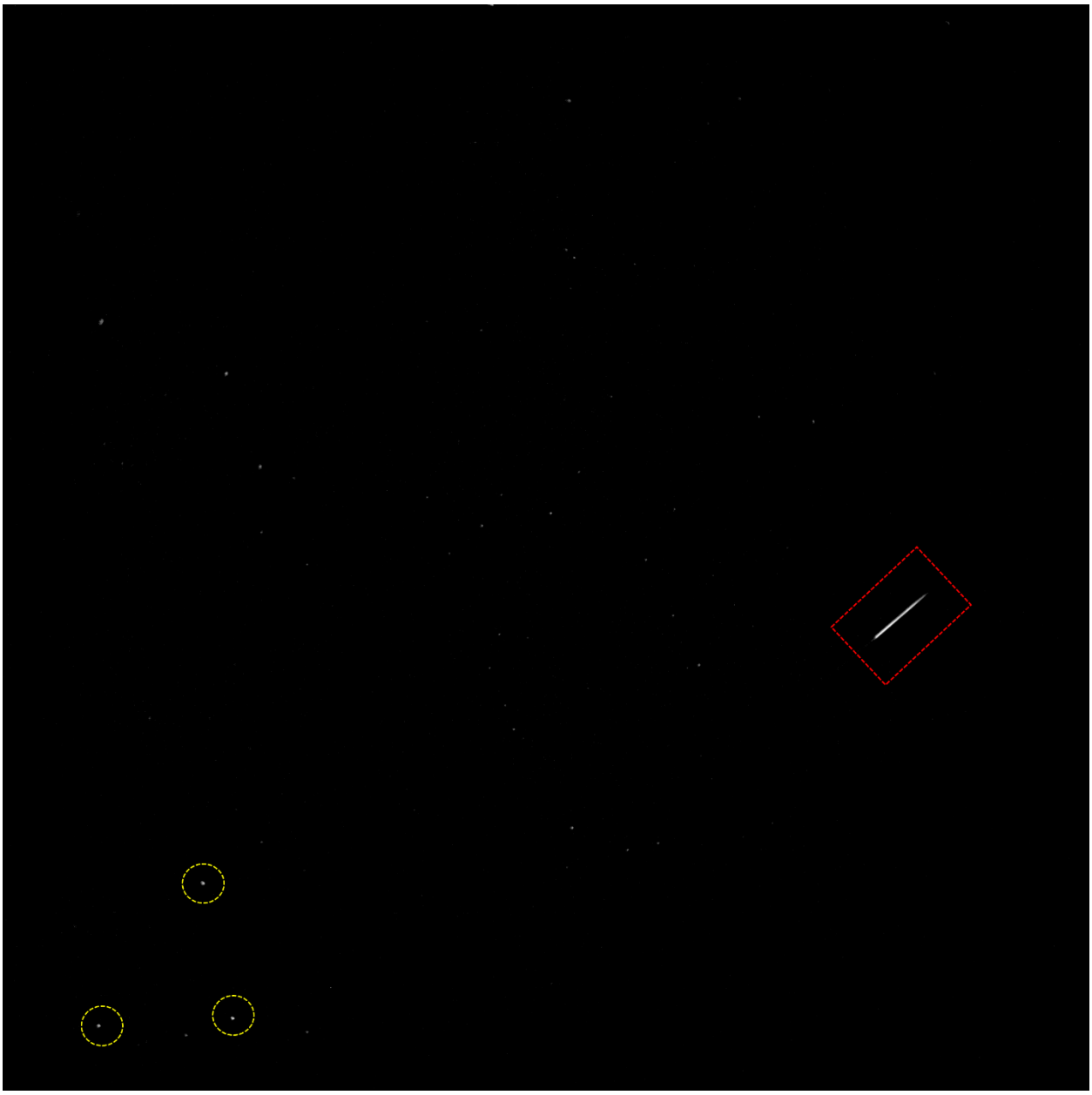

Figure 5.6. An example image captured from the IDS camera during the King City field campaign, using an exposure time of 500 ms.

The second field campaign occurred in Timmins, Ontario, the city where the Strato-Science 2023 campaign was also held. During this campaign, the IDS camera's selected settings were once again tested, with even more stars visible due to the decreased light pollution. During these

campaigns, the adaptive thresholding step in the RSO detection algorithm was tuned to account for situations where there may be too many or too few point sources in the FOV.

5.6. Results

The stratospheric balloon flight lasted 4 hours and 32 minutes, from initial gondola lift off to the payload shutdown, issued by a command from the ground. During the flight, the gondola experienced stability issues during the balloon's ascent into the stratosphere. The images captured during this time were difficult to process, given that the background stars and RSOs in the image moved drastically from frame to frame because of this motion. Performing visual verification of RSOs and processing AD results during this time was extremely difficult, and so these images were not considered in these results. Furthermore, the flight duration was significantly shorter than expected, further decreasing the potential analysis window for RSO detection and AD. Given these two factors, the analysis was limited to a period of 92 minutes, corresponding to 11087 images captured during this time. However, there was still some gondola movement for this duration, which may have negatively impacted the results. Overall, the mission was successful, and the payload survived the ascent, descent, and environmental conditions of the stratosphere. Figure 5.7 shows the payload before and after the mission.

Including the 100 ms exposure time for each image, AD algorithm processing time, RSO detection processing time, and payload health data saving, the average iteration processing time was 502 ms. While this shows potential in real-time applications, this processing time could be reduced significantly with further algorithm and general software optimizations. These optimizations are discussed in Section 5.7.

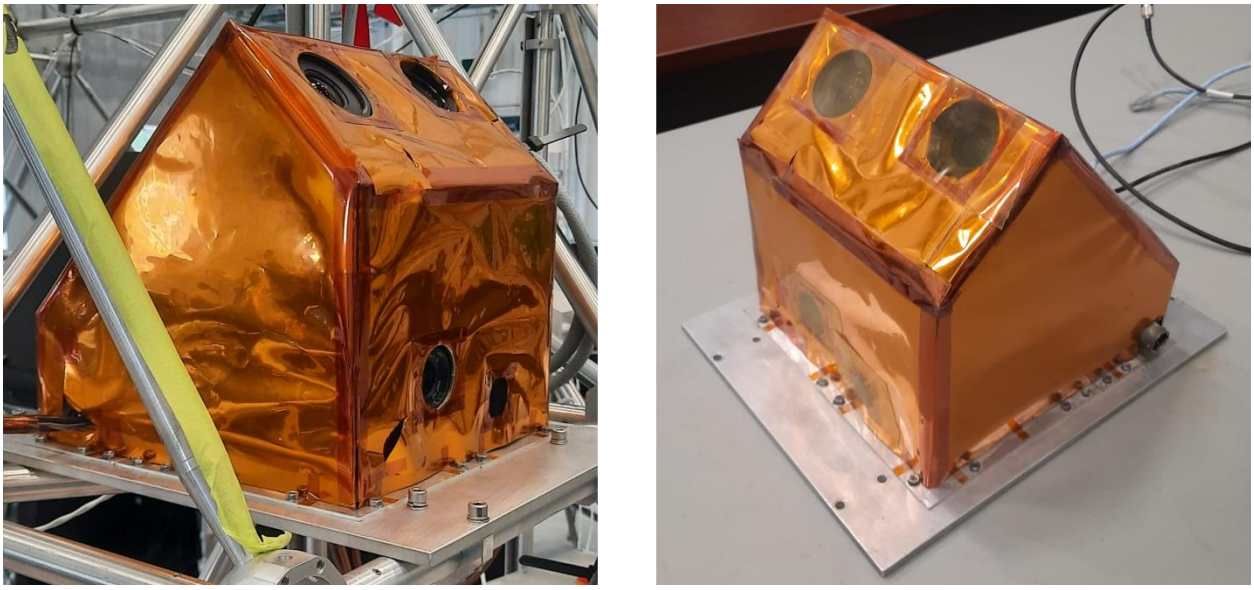

Figure 5.7. Payload integrated on the gondola, ready for flight (left) and payload retrieved from gondola, after the flight (right).

5.6.1. RSO Detection Results

Given that the mission was a technology demonstration, the main objective for the RSO detection segment of the payload was to detect any RSOs at all, during the flight. Success would be achieved if an RSO detection reported by the algorithm's output was verified visually against the corresponding raw images. Given this objective, these results do not consider metrics such as precision and recall, and instead seek to quantify and verify the number of RSOs reported by the algorithm and the number of total detections corresponding to each RSO. To give an understanding of the consistency of the detections, the longest consistent detection of each RSO is also given. These results are provided in Table 5.1.

Table 5.1. Number of RSOs detected by the algorithm that were verified, along with the total number of detections and the longest consistent detection for each RSO.

RSO Number	**Total Detections (Frames)**	**Longest Consistent Detection (Frames)**
1	53	18
2	34	34
3	40	40
4	36	36
5	14	14
6	66	55
7	31	16
8	106	106
9	13	6
10	159	159
11	117	117
Total Unique RSOs: 11	**Total Detections: 669**	**Longest Consistent Detection: 159**

By plotting the pixel centroids of an RSO, corresponding to each of the three sequential images it was observed in, a visual can be constructed to represent how the RSO detection algorithm works, and what it outputs. Figure 5.8 is an example of this.

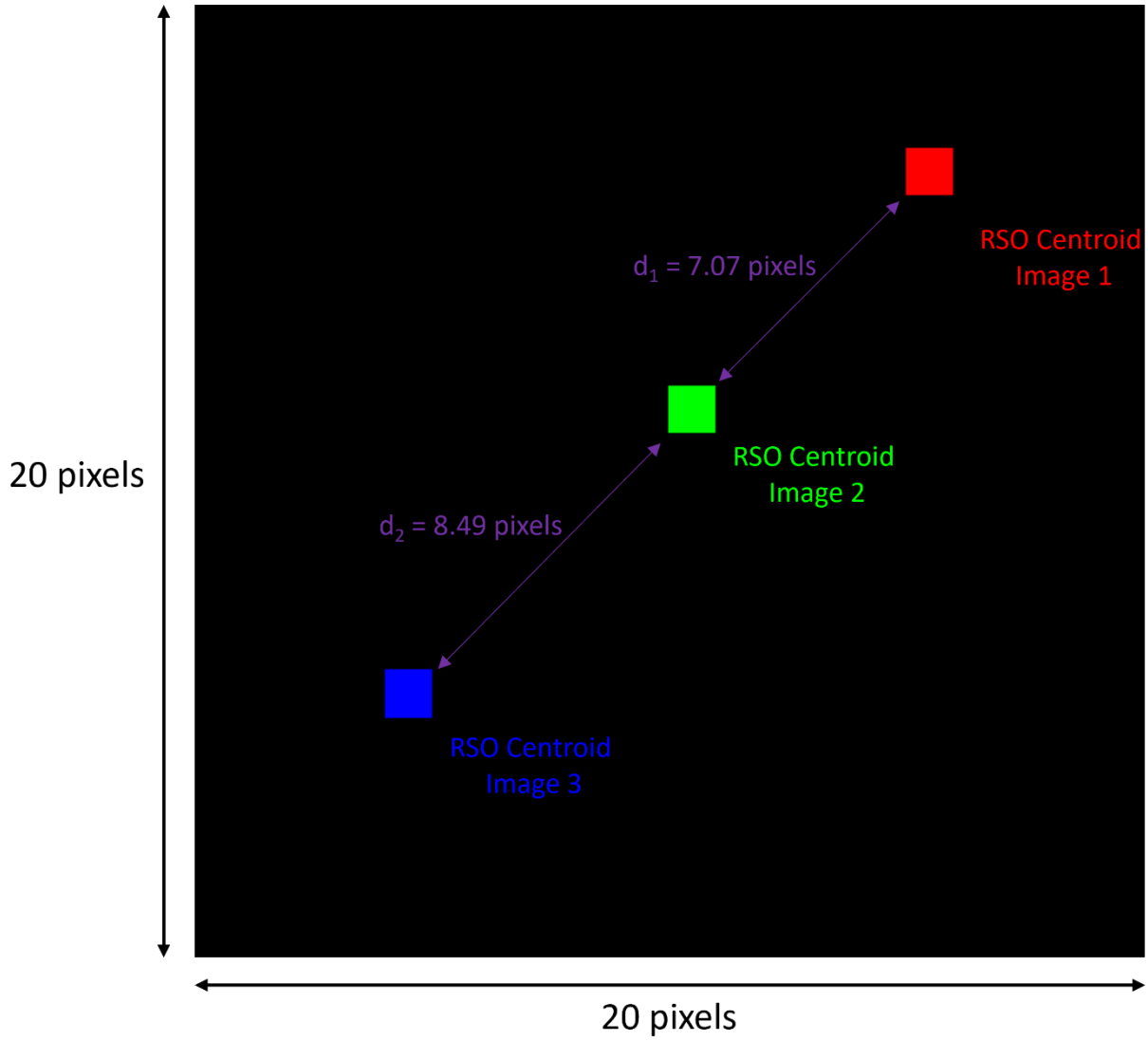

Figure 5.8. RSO centroids corresponding to three sequential images, plotted as single, coloured pixels on a black background. The Euclidean distances calculated by the algorithm are also plotted.

Upon analyzing the raw images captured by the STARDUST imager, it was observed that the lit pixels corresponding to stars and RSOs in the images were fewer and less intense (in terms of pixel value) than what was observed during ground campaigns. The cause of this is currently being investigated, with the current hypothesis being that the harsh vacuum of the stratospheric

environment resulted in degradation of the imager. Due to this suspected degradation, most of the RSOs in the images corresponded to a single pixel each, which would not be detected by the algorithm. This was a result of the algorithm imposing a minimum pixel threshold of 10, used to filter out noise, and was experimentally determined during ground campaigns. This meant that objects in the images corresponding to fewer than 10 pixels would be ignored, which includes these single-pixel RSOs. Nevertheless, the algorithm was able to detect RSOs in the images, given that there were some larger and brighter RSOs during this imaging window.

The dynamic threshold function implemented in the algorithm proved to be very useful, given that the algorithm automatically decreased the threshold to account for the fewer points it was detecting during the mission. This allowed fainter RSOs to be detected, though they had to pass the minimum pixel threshold as well. The algorithm could certainly benefit from better object analysis in the images, employing a dynamic analysis of the points similar to the dynamic threshold. For example, when fewer than desirable points are detected in an image, the algorithm could reduce the minimum pixel threshold to allow more points to be considered.

The linear motion model appears to be a good estimate of RSO motion in these images, given that the RSOs that passed the threshold and pixel area requirement were consistently detected in the images, as the RSOs passed through the FOV of the imager. The built-in tolerance to the linear motion model as defined in Equation 5.1 proved to be invaluable in detecting RSOs with the gondola movement, since this movement also caused RSOs to move non-linearly in the images. The equidistant requirement for RSO detection, as enforced by Equation 3.1, was also a good estimate of the motion of the RSOs.

Though intended to be a technology demonstration, the images captured by the STARDUST payload have proved to be an invaluable dataset for SSA, given the presence of numerous RSOs. Furthermore, a unique opportunity is offered by the challenges with the dataset, including the drastic background star movement (caused by the gondola movement) and the faint, small RSOs. Improvements and further research into algorithms for detecting RSOs in these images is covered in the Conclusion and Future Work section.

5.6.2. Attitude Determination Results

Both the LIS and tracking mode AD algorithms yielded promising results, both in real-time and in post-mission analysis. Results for these algorithms are discussed in-depth in [106].

5.7. Conclusions and Future Work

The real-time RSO detection algorithm was successfully able to detect RSOs during the mission. Though the RSOs within the image appeared much smaller and fainter than what was observed during ground campaigns, the algorithm was able to detect 11 unique RSOs corresponding to 669 total detections in the 92-minute analysis window. The dynamic image thresholding technique and linear motion model proved to be excellent algorithms in consistently capturing the RSOs that passed the minimum pixel threshold. However, it was determined that the minimum pixel threshold itself needs improvement, given that many RSOs observed in the raw images were much smaller than the 10-pixel threshold.

Several improvements could be made to the RSO detection algorithm and are planned to be incorporated in future research. As mentioned, the swaying of the gondola, which caused the background stars to appear to move throughout the images, suggests that a method to correct the

apparent motion of the stars could be added to create a more robust RSO detection algorithm. Such a method is currently being developed and is being tested with optical imagery from the FAI onboard the CASSIOPE satellite. Another improvement to be made is the addition of a tracking method to give each detected RSO a unique identity, carried through its detection in subsequent images. This can give be used to identify RSOs of particular interest, in real-time. Such a tracking implementation is also being investigated for future work. Lastly, the RSO detection algorithm can be improved by incorporating a more advanced motion model than what was experimentally determined, as described by Equation 5.1. While the linear motion model appears to work effectively for many of the RSOs found in the images captured from the stratosphere, RSOs in general do not always appear to be moving linearly through an imager's FOV, as can be observed in the FAI instrument's images.

While the RSO detection and AD algorithms were isolated to reduce risk, future iterations of the mission will seek to combine the algorithms to improve their respective performance. For example, the RSO detection algorithm could be used to remove RSOs in the images that are passed to the AD algorithm. This would be helpful in reducing false detections in the AD algorithm from the RSOs present in the images.

6. Conclusions

6.1. Summary

In recent years, the number of RSOs orbiting the Earth has increased, calling for an improvement in SSA, given the large orbital velocities of these objects and potential for collisions. The long-term goal of this research involves the creation of a unifying, open source, community-driven RSO identification framework, which can allow any party anywhere in the world to contribute to the improvement of SSA. However, even at the first step of this framework, RSO detection and tracking, various challenges exist. Real, open-source optical imagery needed for RSO tracking algorithm development is limited. Moreover, the optical imagery that is available is difficult to label, given the presence of artifacts, the large number of objects, and the small size of each object, rendering general labelling tools ineffective for efficient labelling. Lastly, to further support the RSO identification framework, algorithms developed using such imagery should ultimately be able to run onboard a satellite, in real-time, allowing for efficient RSO data downlink. These algorithms could be used for payloads dedicated to SSA or added as an additional function to existing payloads. This thesis aims to tackle these challenges associated with the first step of this framework.

In Chapter 2, a stratospheric balloon payload was developed to address the need for more optical imagery for RSO tracking algorithm development. The algorithm was developed to autonomously determine the current flight conditions of the payload and change camera parameters to capture high-quality images during ideal conditions, and low-quality images during poor conditions. This was done by extracting the altitude and time values from an

onboard GPS in real-time and using a sliding window to determine the host gondola's flight conditions. A variety of modes were programmed for each condition. Low-quality imaging modes included ascent and descent, corresponding to when the gondola is ascending and descending respectively, causing unfavorable conditions for imaging. High-quality imaging modes included coast and dawn-and-dusk, corresponding to when the gondola had stabilized at a high altitude and the time preceding sunrise and proceeding sunset, respectively, which provided the best imaging opportunities. A safe mode was also programmed in the event of software or hardware errors, such as a malfunctioning GPS or unintended power out condition. The imaging algorithm and payload were successfully demonstrated onboard a stratospheric balloon flight which launched from Timmins, Ontario in August 2022. While the onboard GPS experienced failure, the imaging algorithm successfully operated in safe mode, capturing over 93000 images of the night sky from the stratosphere, with 100s of visually confirmed RSOs within them. These images are also being used in further research outside the scope of this thesis. I developed the flight software that was used to autonomously control the main camera, consisting of the boot-up operations, GPS sensor data processing, mode selection, camera parameter setting, image acquisition, payload status data saving, and post-mission results analysis.

In Chapter 3, a novel rules-based RSO tracking algorithm was developed to address the need for an RSO tracking algorithm itself, able to process space-based, low-resolution imagery, even if the host satellite experienced reduced attitude control. The algorithm used a series of smaller algorithms to process images in a three-image sliding window for RSO detection and tracking. The RSO tracking algorithm first used a custom thresholding algorithm, which consisted of a dynamic threshold determined using the local background of an image applied to small sections

of the image at a time. The point sources revealed by this technique were then passed to a simultaneous pose and correspondence estimation algorithm to account for the effect of the host spacecraft's non-constant attitude. This algorithm was used to determine the rotation and translation of the majority of the points between the three images, the idea being that the stars will account for that majority. These stars were then removed from all the images, and the remaining points were rotated and translated to counter the rotation and translation of the stars, effectively aligning the images together. These remaining points consisting of RSOs, noise, and erroneously unremoved stars were then passed to a final algorithm. This algorithm applied a linear motion model to detect points travelling in straight lines across images that were also equidistant from image to image, intended to capture the RSOs and separate them from the other objects. The algorithm was successfully demonstrated on a hand-labelled, 878-image dataset, achieving 79% precision and 71% recall, and was able to detect 87% of objects passing the FOV of the imager at least once. I developed the entire RSO tracking algorithm outlined here, as well as the 878-image dataset.

In Chapter 4, multiple fundamental components of RSO tracking algorithm development were described and developed. A framework for autonomously and efficiently labelling SSA imagery was proposed, alongside a corresponding four-tool annotation suite to address the problem of the lack of efficient labelling tools. The tools were used to construct a preliminary dataset to demonstrate the efficacy of the tools and promote the creation of a benchmark dataset for clear and equal comparison of future developed algorithms. Lastly, a paradigm for RSO tracking algorithm development was proposed, alongside a preliminary corresponding RSO tracking algorithm to demonstrate the efficacy of the preliminary dataset and demonstrate the potential for

future algorithm development. The four-tool annotation suite was developed in Python, and incorporated components of the RSO tracking algorithm developed in Chapter 3 to automate the annotations. The first tool used the custom thresholding algorithm from Chapter 3 to automatically produce preliminary masks, followed by a second tool to manually edit these masks in a simple, intuitive GUI. The third tool modified the star removal algorithm and linear motion model to instead automatically classify points as stars, RSOs, and noise. The fourth tool again allowed users to simply and intuitively edit the classes of the objects. The preliminary dataset was generated using these tools, where sequences were hand-selected from images captured by the host imager during 2023. Sequences were selected to capture a variety of imaging conditions, such as the appearance of Earth's limb, the appearance of the Milky Way, high-noise instances, and multiple-RSO instances. The tracking algorithm was developed by using the proposed tracking-by-detection paradigm, implementing the U-Net CNN as the detector and combining it with two different trackers. The first tracker was based on the Hungarian algorithm to assign detections from frame to frame. The second tracker was based on a custom priority scheme, assigning new detections to well-established tracks first. The tools were successfully demonstrated by creating the mentioned 500-image dataset, resulting in richer annotations available in more formats than the dataset developed in Chapter 3. The dataset and RSO tracking algorithm were also successful, given that the dataset was used to train and test the RSO tracking algorithm, with qualitative results showing promise for future development. I developed the four-tool annotation suite, the 500-image dataset, and the RSO tracking algorithm.

In Chapter 5, a real-time RSO detection algorithm was demonstrated onboard a stratospheric balloon platform to further support the overarching RSO identification framework. The

algorithm was developed to run alongside an existing AD algorithm for a dual-purpose payload concept. The algorithm was developed by using the lessons learned from stratospheric RSO imaging in Chapter 2, as well as the lessons learned from RSO tracking algorithm development in Chapter 3. Given the lack of artifacts observed in the imagery in Chapter 2, the custom thresholding algorithm from Chapter 3 was simplified to perform simple thresholding. Additionally, the stabilized attitude of the gondola during the flight prompted the modification of the star removal algorithm in Chapter 3, such that the new algorithm did not need to register a rotation and translation between the images, and instead removed static objects, which were the stars. The same linear motion model from Chapter 3 was implemented to detect RSOs. Finally, further modifications were added for real-time operation, such as a custom dynamic threshold. This threshold was programmed such that it changed depending on the number of images present, increasing the threshold in the case of many objects observed, and vice versa. While the payload's imager experienced hardware degradation during the flight, the RSO detection algorithm successfully detected 11 unique RSOs, corresponding to 669 frames of RSO detection. The complete algorithm, including image capture time, AD, RSO detection, and health data saving operated with a per-frame processing time of 502 ms, successfully demonstrating real-time operation. I developed the real-time RSO detection algorithm, the main flight software (including boot-up operations, camera initialization, image capture functions, and health data saving), and the parts selection trade study.

The successful demonstration of all these components involved in RSO tracking pave the way for using optical imagery for the development of a unifying, open source, community driven RSO identification framework for the continued sustainable use of space.

6.2. Future Work

Several promising options exist to continue the work developed in each of these chapters, and overall.

With respect to Chapter 2, several options exist to further build on the optical image acquisition payload concept. The next step with this payload is to develop a CubeSat mission around it, to capture space-based images of RSOs. The autonomous algorithm developed in Chapter 2 would need to be modified to efficiently capture images from space, in the following ways. Firstly, the algorithm can compress the images by using a compressed image format or storing segments of images instead of full frames. Alternatively, the RSO detection data could be stored alone, using an algorithm such as the one developed in Chapter 5 to do so. Images without RSO detections could be discarded entirely. Furthermore, the FPGA platform in Chapter 2 could be leveraged further, dedicating parts of the fabric to efficiently perform operations such as the mentioned compression scheme or RSO detection.

With respect to Chapter 3 and 4, the RSO tracking algorithms developed here could be further improved. Once well-developed tracking-by-detection algorithms have been developed, the detection process and tracking process could be more tightly combined into a tracking-and-detection paradigm. While algorithms using this paradigm may lose the modularity and other advantages posed by the tracking-by-detection paradigm, using existing tracks to search for future detections could speed up the detection process for real-time implementation and be more robust to missed detections. Furthermore, detection techniques could benefit from special consideration applied towards detecting faint objects and RSOs in particular. For example, if

using a CNN detector, the loss function can be modified such that the RSO class is weighted more significantly, or the background class is weighted less significantly. Additionally, the datasets developed in these chapters can be further reviewed and enhanced by others, paying special attention to labelling low-illumination objects.

With respect to Chapter 4 specifically, the automation tools can be further iteratively improved by incorporating a CNN detector for the automated mask generation. This CNN detector can be improved by learning from the edited data, which can then be used to generate better data in a positive feedback loop. The classification tools and algorithms can use a similar iterative improvement scheme, while tracking functions can be added to the existing tool suite to generate tracking annotations as well. The preliminary dataset needs to be augmented with these tracking labels, and further sequences need to be labelled to develop a complete benchmark dataset. Dataset hosting, algorithm comparison, and dataset version control need to be considered to extend this research.

With respect to Chapter 5, the RSO detection algorithm can next be improved for implementation on a dedicated SSA mission. The RSO detection algorithm should be augmented to an RSO tracking algorithm, making use of previous detections to individually track RSOs for further space-based or ground analysis for RSOs of interest. Furthermore, the RSO tracking algorithm can be used to move the host imager itself such that the RSO of interest is consistently held in the center of the imager's FOV for dedicated imaging of a particular RSO.

Overall, this research can be extended by developing the rest of the RSO identification framework. Significant research needs to be conducted to assess the feasibility of using angles-only IOD on the RSOs extracted from the images in this research. Furthermore, the feasibility of

RSO identification needs to be conducted as well, using algorithms to match preliminary orbits determined in the previous step, along with other metrics.

References

[1] West, J.; Wark, W.; Shull, A. *The Importance of Satellites to Life on Earth.* https://www.cigionline.org/multimedia/the-importance-of-satellites-to-life-on-earth/ (accessed 2024-06-11).

[2] Pultarova, T.; Howell, E. *Starlink Satellites: Facts, Tracking and Impact on Astronomy.* https://www.space.com/spacex-starlink-satellites.html (accessed 2024-06-11).

[3] Zhang, J.; Cai, Y.; Xue, C.; Xue, Z.; Cai, H. Leo Mega Constellations: Review of Development, Impact, Surveillance, and Governance. *Space: Science & Technology* **2022**. DOI: 10.34133/2022/9865174.

[4] Howard, K.; Ah, A. V. *Large Constellations of Satellites: Mitigating Environmental and Other Effects.* https://www.gao.gov/products/gao-22-105166 (accessed 2024-06-11).

[5] *LEGEND: 3D/OD Evolutionary Model.* https://orbitaldebris.jsc.nasa.gov/modeling/legend.html (accessed 2024-06-11).

[6] Weeden, B. *2009 Iridium-Cosmos Collision Fact Sheet.* Secure World Foundation, 2010. https://swfound.org/media/6575/swf_iridium_cosmos_collision_fact_sheet_updated_2012.pdf (accessed 2024-06-11).

[7] Kelso, T. S.; Gorski, A. *SPACE SURVEILLANCE: LESSONS LEARNED FROM THE IRIDIUM-COSMOS COLLISION.* AGI. https://www.agi.com/getmedia/d4bbbff7-2e79-48e8-a3ac-2407afff9951/Space-Surveillance-Lessons-Learned-From-The-Iridium-Cosmos-Collision.pdf?ext=.pdf) (accessed 2024-06-11).

[8] Weeden, B. *2007 Chinese Anti-Satellite Test Fact Sheet.* Secure World Foundation, 2010. https://swfound.org/media/9550/chinese_asat_fact_sheet_updated_2012.pdf (accessed 2024-06-11).

[9] *Space Debris: Assessing the Risk.* https://www.esa.int/Enabling_Support/Operations/Space_debris_assessing_the_risk (accessed 2024-06-11).

[10] Kessler, D. J.; Johnson, N. L.; Liou, J.-C.; Matney, M. The Kessler Syndrome: Implications to Future Space operations. In *Advances in the Astronautical Sciences; 2010.*

[11] McWhinney, M. *SPACE SITUATIONAL AWARENESS.* Space Security Index, 2020. https://spacesecurityindex.org/wp-content/uploads/2020/11/IssueGuide_SSAfix.pdf (accessed 2024-06-11).

[12] Kelso, T. S. *Lost List.* https://celestrak.org/satcat/lost.php (accessed 2024-06-11).

[13] Garofalo, M. *"Lost" Satellite Found after Orbiting Undetected for 25 Years.* https://www.space.com/lost-satellite-found-us-space-force-data (accessed 2024-06-11).

[14] Phillips, C. *Time for Common Sense with the Satellite Catalog.* https://www.thespacereview.com/article/3215/1 (accessed 2024-06-11).

[15] *Help Documentation.* https://www.space-track.org/documentation#/user_agree (accessed 2024-06-11).

[16] Shell, J. R. Optimizing orbital debris monitoring with optical telescopes. In *Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference; 2010.*

[17] Cabello, A.; Fletcher, J. SatSim: a synthetic data generation engine for electro-optical imagery of resident space objects. In *Proc. SPIE 12121, Sensors and Systems for Space Applications XV*, Orlando, FL, 2022. DOI: 10.1117/12.2618733

[18] Suthakar, V.; Sanvido, A.A.; Qashoa, R.; Lee, R.S.K. Comparative Analysis of Resident Space Object (RSO) Detection Methods. *Sensors* **2023**, *23*, 9668. DOI: 10.3390/s23249668

[19] Kunalakantha, P.; Baires, A.V.; Dave, S.; Clark, R.; Chianelli, G.; Lee, R.S.K. Stratospheric Night Sky Imaging Payload for Space Situational Awareness (SSA). *Sensors* **2023***, 23,* 6595. DOI: 10.3390/s23146595

[20] Schirru, L.; Pisanu, T.; Podda, A. The Ad Hoc Back-End of the BIRALET Radar to Measure Slant-Range and Doppler Shift of Resident Space Objects. *Electronics* **2021***, 10,* 577. DOI: 10.3390/electronics10050577

[21] Ionescu, L.; Rusu-Casandra, A.; Bira, C.; Tatomirescu, A.; Tramandan, I.; Scagnoli, R.; Istriteanu, D.; Popa, A.E. Development of the Romanian Radar Sensor for Space Surveillance and Tracking Activities. *Sensors* **2022***, 22,* 3546. DOI: 10.3390/s22093546

[22] Ender, J.; Leushacke, L.; Brenner, L.; Wilden, H. Radar Techniques for Space Situational Awareness. In *Proceedings of the IEEE International Radar Symposium (IRS),* Leipzig, Germany, September 7–9, 2011.

[23] Losacco, M.; Di Lizia, P.; Massari, M.; Naldi, G.; Pupillo, G.; Bianchi, G.; Siminski, J. Initial orbit determination with the multibeam radar sensor BIRALES. *Acta Astronautica* **2020***, 167,* 374–390. DOI: 10.1016/j.actaastro.2019.10.043

[24] Kennewell, J.A.; Vo, B.-N. An overview of space situational awareness. In *Proceedings of the 16th International Conference on Information Fusion*, Istanbul, Turkey, July 9–12, 2013; pp. 1029–1036.

[25] Abercromby, K.J.; Seitzer, P.; Cowardin, H.M.; Barker, E.S.; Matney, M.J. *Michigan Orbital DEbris Survey Telescope Observations of the Geosynchronous Orbital Debris Environment Observing Years: 2007–2009*. National Aeronautics and Space Administration, 2011. https://ntrs.nasa.gov/api/citations/20110022976/downloads/20110022976.pdf (accessed on 13 January 2023).

[26] Scott, R.; Thorsteinson, S. Key Findings from the NEOSSat Space-Based SSA Microsatellite Mission. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference,* Maui, HI, USA, September 11–14, 2018.

[27] Krantz, H.; Pearce, E.C.; Block, A.; Observatory, S. Characterization of LEO Satellites with All-Sky Photometric Signatures. In *Proceedings of the Advanced Optical Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Maui, HI, USA, September 27–30, 2022.

[28] Petit, A. Extraction of light curve from passive observations during survey campaign in LEO, MEO and GEO regions. In *Advanced Optical Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Maui, HI, USA, September 27–30, 2022.

[29] Oltrogge, D.L. The “we” approach to space traffic management. In *Proceedings of the 15th International Conference on Space Operations,* Marseille, France, May 28–June 1, 2018; pp. 1–21.

[30] Clark, R.; Fu, Y.; Dave, S.; Lee, R. Simulation of RSO Images for Space Situation Awareness (SSA) Using Parallel Processing. *Sensors* **2021***, 21,* 7868. DOI: 10.3390/s21237868

[31] Clemens, S. On-Orbit Resident Space Object (RSO) Detection Using Commercial Grade Star Trackers. M.Sc. Thesis, York University, Toronto, ON, Canada, 2019. https://yorkspace.library.yorku.ca/xmlui/bitstream/handle/10315/36799/Clemens_Samuel_D_2019_MSc.pdf?sequence=2&isAllowed=y (accessed 2023-03-02).

[32] Dave, S.; Clark, R.; Gabriel, C.; Lee, R. Machine Learning Implementation for in-Orbit RSO Orbit Estimation Using Star Tracker Cameras. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Online, September 15–18, 2020.

[33] Sease, B.; Flewelling, B.; Xu, Y. Catalog-Free Angular Rate Estimation and on-Line Detection of Resident Space Objects. In *Proceedings of the 24th AAS/AIAA Space Flight Mechanics Meeting,* Santa Fe, NM, USA, January 26–30, 2014. DOI: 10.13140/RG.2.1.1591.4728

[34] Spiller, D.; Magionami, E.; Schiattarella, V.; Curti, F.; Facchinetti, C.; Ansalone, L.; Tuozzi, A. On-Orbit Recognition of Resident Space Objects by Using Star Trackers. *Acta Astronautica* **2020***, 177,* 478–496. DOI: 10.1016/j.actaastro.2020.08.009

[35] Hasenohr, T. Initial Detection and Tracking of Objects in Low Earth Orbit. M.Sc. Thesis, University of Stuttgart Institute of Applied Optics, Stuttgart, Germany, 2016. https://core.ac.uk/download/pdf/77234443.pdf (accessed 2023-03-02).

[36] Fitzgerald, G.; Funke, Z.; Cabello, A.; Asari, V.; Fletcher, J. Toward Deep-Space Object Detection in Persistent Wide Field of View Camera Arrays. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Maui, HI, USA, September 14–17, 2021.

[37] Lyle, A. *Up through the Atmosphere; AlbertaSat Gets Next Satellite Test off the Ground with Stratospheric Balloon.* University of Manitoba, 2018. https://www.ualberta.ca/science/news/2018/august/albertasat-satellite-test-stratospheric-balloon.html (accessed 2023-03-02).

[38] Flaten, J.; Bartlett, J.; Krieg, E.; Lens, E.; Bowers, R. Flying “Mock CubeSats” on Stratospheric Balloon Missions. In *Proceedings of the 2019 Academic High-Altitude Conference,* Ames, IA, USA, June 26–28, 2019; pp. 1–13. DOI: 10.31274/ahac.240

[39] Wozniak, P.; Prasad, L.; Wohlberg, B. Moving point source detection and localization in wide-field images. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Maui, HI, USA, September 11–14, 2018.

[40] April, J. Nanosat Employment: A Theoretical CONOPS for Space Object Identification. M.Sc. Thesis, Naval Postgraduate School, Monterey, CA, USA, 2014. https://core.ac.uk/download/pdf/36734841.pdf (accessed 2023-03-02).

[41] *PYNQ-Z1 Reference Manual*. https://digilent.com/reference/programmable-logic/pynq-z1/reference-manual?redirect=1 (accessed 2023-03-02).

[42] *ZED-F9P u-Blox F9 High Precision GNSS Module Data Sheet.* u-blox, 2018. https://cdn.sparkfun.com/assets/8/3/2/b/8/ZED-F9P_Data_Sheet.pdf (accessed 2023-03-02).

[43] *MicroSD Card 3TE4.* https://www.innodisk.com/en/products/flash-storage/sd-card-and-microsd-card/microsd-card-3te4 (accessed 2023-03-02).

[44] *pco.panda 4.2 Ultra Compact SCMOS Camera*; Excelitas PCO GmbH. https://www.pco.de/fileadmin/user_upload/pco-product_sheets/DS_PCOPANDA42_V104.pdf (accessed 2023-03-02).

[45] *ZEISS Dimension 2/25*. Carl Zeiss AG, 2018. https://www.zeiss.com/content/dam/consumer-products/downloads/industrial-lenses/datasheets/en/dimension-lenses/datasheet-zeiss-dimension-225.pdf (accessed 2023-03-02).

[46] Loff, S.; Dunbar, B. *CubeSats Overview*. https://www.nasa.gov/mission_pages/cubesats/overview (accessed 2023-03-02).

[47] Early, D. A.; Krimchansky, A. Using Commercial Off-the-Shelf Fuses in Vacuum. *Journal of Spacecraft and Rockets* **2019***, 56* (4), 1282–1285. DOI: 10.2514/1.a34427

[48] *Strato-Science 2022 Campaign*. https://www.asc-csa.gc.ca/eng/sciences/balloons/campaign-2022.asp (accessed 2023-01-13).

[49] Dave, S.; Clark, R.; Lee, R.S.K. RSOnet: An Image-Processing Framework for a Dual-Purpose Star Tracker as an Opportunistic Space Surveillance Sensor. *Sensors* **2022***, 22,* 5688. DOI: 10.3390/s22155688.

[50] Kelso, T.S. Analysis of the Iridium 33-Cosmos 2251 Collision. In *19th AIAA/AAS Astrodynamics Specialist Conference,* Pittsburgh, PA, August 11, 2009.

[51] Johnson, N. L.; Stansbery, E.; Liou, J.-C.; Horstman, M.; Stokely, C.; Whitlock, D. The Characteristics and Consequences of the Break-up of the Fengyun-1C Spacecraft. *Acta Astronautica* **2008**, *63* (1–4), 128–135. DOI: 10.1016/j.actaastro.2007.12.044

[52] Migaud, M. R. Protecting Earth's Orbital Environment: Policy Tools for Combating Space Debris. *Space Policy* **2020**, *52*, 101361. DOI: 10.1016/j.spacepol.2020.101361

[53] Weeden, B. C.; Cefola, P.; Sankaran, J. Global Space Situational Awareness Sensors. Secure World Foundation, 2010. https://swfound.org/media/15274/global%20ssa%20sensors-amos-2010.pdf (accessed 2024-06-21).

[54] Nir, G.; Zackay, B.; Ofek, E. O. Optimal and Efficient Streak Detection in Astronomical Images. *The Astronomical Journal* **2018**, *156* (5), 229. DOI: 10.3847/1538-3881/aaddff

[55] Waszczak, A.; Prince, T. A.; Laher, R.; Masci, F.; Bue, B.; Rebbapragada, U.; Barlow, T.; Surace, J.; Helou, G.; Kulkarni, S. Small Near-Earth Asteroids in the Palomar Transient Factory Survey: A Real-Time Streak-Detection System. *Publications of the Astronomical Society of the Pacific* **2017**, *129* (973), 034402. DOI: 10.1088/1538-3873/129/973/034402

[56] Canny, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1986**, *PAMI-8* (6), 679–698. DOI: 10.1109/tpami.1986.4767851

[57] Cvrček, V.; Šára, R. Detection and Certification of Faint Streaks in Astronomical Images. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, Prague, Czech Republic, 2019; pp. 498–509. DOI: 10.5220/0007399804980509

[58] Jeffries, C.; Acuña, R. Detection of Streaks in Astronomical Images Using Machine Learning. *Journal of Artificial Intelligence and Technology* **2023**. DOI: 10.37965/jait.2023.0413

[59] Yao, R.; Zhang, Y. Compressive Sensing for Small Moving Space Object Detection in Astronomical Images. *Journal of Systems Engineering and Electronics* **2012**, *23* (3), 378–384. DOI: 10.1109/jsee.2012.00047

[60] Privett, G.; Appleby, G.; Sherwood, R. Image stacking techniques for GEO satellites and a three-site collection. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Maui, HI, USA, September 9–12, 2014.

[61] Clemens, S.; Lee, R.S.K.; Harrison, P.; Soh, W. Feasibility of Using Commercial Star Trackers for On-Orbit Resident Space Object Detection. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Maui, HI, USA, September 11–14, 2018.

[62] Lang, D.; Hogg, D. W.; Mierle, K.; Blanton, M.; Roweis, S. ASTROMETRY.NET: BLIND ASTROMETRIC CALIBRATION OF ARBITRARY ASTRONOMICAL IMAGES. *The Astronomical Journal* **2010**, *139* (5), 1782–1800. DOI: 10.1088/0004-6256/139/5/1782

[63] Lovell, T.A.; Sinclair, A.J.; Newman, B. Angles Only Initial Orbit Determination: Comparison of Relative Dynamics and Inertial Dynamics Approaches with Error Analysis. In *2018 Space Flight Mechanics Meeting, Kissimmee,* FL, USA, January 8–12, 2018.

[64] Cogger, L.; Howarth, A.; Yau, A.; White, A.; Enno, G.; Trondsen, T.; Asquin, D.; Gordon, B.; Marchand, P.; Ng, D.; Burley, G.; Lessard, M.; Sadler, B. Fast Auroral Imager (FAI) for the e-POP Mission. *Space Science Reviews* **2014**, *189* (1–4), 15–25. DOI: 10.1007/s11214-014-0107-x

[65] *CASSIOPE/e-POP Fact Sheet*. https://epop.phys.ucalgary.ca/quickfacts/ (accessed 2024-06-21).

[66] Bertin, E.; Arnouts, S. SExtractor: Software for Source Extraction. *Astronomy and Astrophysics Supplement Series* **1996**, *117* (2), 393–404. DOI: 10.1051/aas:1996164

[67] Otsu, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics* **1979**, *9* (1), 62–66. DOI: 10.1109/tsmc.1979.4310076

[68] Bolelli, F.; Allegretti, S.; Baraldi, L.; Grana, C. Spaghetti Labeling: Directed Acyclic Graphs for Block-Based Connected Components Labeling. *IEEE Transactions on Image Processing* **2020**, *29*, 1999–2012. DOI: 10.1109/tip.2019.2946979

[69] Chen, Y.; Medioni, G. Object Modelling by Registration of Multiple Range Images. *Image and Vision Computing* **1992**, *10* (3), 145–155. DOI:10.1016/0262-8856(92)90066-c

[70] Besl, P. J.; McKay, N. D. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **1992**, *14* (2), 239–256. DOI:10.1109/34.121791

[71] Lu, F.; Milios, E. Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, June 21–23, 1994. DOI: 10.1109/cvpr.1994.323928

[72] Wenger, M.; Ochsenbein, F.; Egret, D.; Dubois, P.; Bonnarel, F.; Borde, S.; Genova, F.; Jasniewicz, G.; Laloë, S.; Lesteven, S.; Monier, R. The SIMBAD Astronomical Database. *Astronomy and Astrophysics Supplement Series* **2000**, *143* (1), 9–22. DOI: 10.1051/aas:2000332

[73] Nandakumar, S.; Eggl, S.; Tregloan-Reed, J.; Adam, C.; Anderson-Baldwin, J.; Bannister, M. T.; Battle, A.; Benkhaldoun, Z.; Campbell, T.; Colque, J. P.; Damke, G.; Plauchu Frayn, I.; Ghachoui, M.; Guillen, P. F.; Kaeouach, A. E.; Krantz, H. R.; Langbroek, M.; Rattenbury, N.; Reddy, V.; Ridden-Harper, R.; Young, B.; Unda-Sanzana, E.; Watson, A. M.; Walker, C. E.; Barentine, J. C.; Benvenuti, P.; Di Vruno, F.; Peel, M. W.; Rawls, M. L.; Bassa, C.; Flores-Quintana, C.; García, P.; Kim, S.; Longa-Peña, P.; Ortiz, E.; Otarola, Á.; Romero-Colmenares, M.; Sanhueza, P.; Siringo, G.; Soto, M. The High Optical Brightness of the BlueWalker 3 Satellite. *Nature* **2023**, *623* (7989), 938–941. DOI: 10.1038/s41586-023-06672-7

[74] Jordan, J.; Posada, D.; Zuehlke, D.; Radulovic, A.; Malik, A.; Henderson, T. *Satellite Detection in Unresolved Space Imagery for Space Domain Awareness Using Neural Networks*. arXiv, 2022. https://arxiv.org/pdf/2207.11412 (accessed 2024-07-04).

[75] Anderson, J.; Anderson, A.; Zuehlke, D.; Garcia, D.C.; Lovell, T.A. *Resident Space Object Identification in Arbitrary Unresolved Space Images*. ResearchGate, 2023. https://www.researchgate.net/profile/Joseph-Anderson-39/publication/368836988_AAS_23-179_RESIDENT_SPACE_OBJECT_IDENTIFICATION_IN_ARBITRARY_UNRESOLVED_SPACE_IMAGES/links/63fcafc50cf1030a5657c5f5/AAS-23-179-RESIDENT-SPACE-OBJECT-IDENTIFICATION-IN-ARBITRARY-UNRESOLVED-SPACE-IMAGES.pdf?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19. (accessed 2024-07-04).

[76] Brennan, P. *Star Tracker*. https://exoplanets.nasa.gov/star-tracker/ (accessed 2024-07-04).

[77] Qashoa, R.; Driedger, M.; Clark, R.; Harrison, P.; Berezin, M.; Lee, R. S. K.; Howarth, A. SPACEDUST-Optical: Wide-FOV Space Situational Awareness from Orbit. In *Proceedings of the Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, Maui, HI, USA, September 19–22, 2023.

[78] Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision Meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research* **2013**, *32* (11), 1231–1237. DOI: 10.1177/0278364913491297

[79] Kristan, M.; Matas, J.; Leonardis, A.; Vojíř, T.; Pflugfelder, R.; Fernández, G.; Nebehay, G.; Porikli, F.; Čehovin, L. A Novel Performance Evaluation Methodology for Single-Target Trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2016**, *38* (11), 2137–2155. DOI: 10.1109/tpami.2016.2516982

[80] Müller, M.; Bibi, A.; Giancola, S.; Alsubaihi, S.; Ghanem, B. TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild. In *Computer Vision – ECCV 2018*, 2018. DOI: 10.1007/978-3-030-01246-5_19

[81] Dendorfer, P.; Ošep, A.; Milan, A.; Schindler, K.; Cremers, D.; Reid, I.; Roth, S.; Leal-Taixé, L. MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking. *International Journal of Computer Vision* **2020**, *129* (4), 845–881. DOI: 10.1007/s11263-020-01393-0

[82] Dave, A.; Khurana, T.; Tokmakov, P.; Schmid, C.; Ramanan, D. Tao: A Large-Scale Benchmark for Tracking Any Object. In *Computer Vision – ECCV 2020*, 2020. DOI: 10.1007/978-3-030-58558-7_26

[83] Mueller, M.; Smith, N.; Ghanem, B. A Benchmark and Simulator for UAV Tracking. In *Computer Vision – ECCV 2016*, 2016. DOI: 10.1007/978-3-319-46448-0_27

[84] Chenouard, N.; Smal, I.; de Chaumont, F.; Maška, M.; Sbalzarini, I. F.; Gong, Y.; Cardinale, J.; Carthel, C.; Coraluppi, S.; Winter, M.; Cohen, A. R.; Godinez, W. J.; Rohr, K.; Kalaidzidis, Y.; Liang, L.; Duncan, J.; Shen, H.; Xu, Y.; Magnusson, K. E.; Jaldén, J.; Blau, H. M.; Paul-Gilloteaux, P.; Roudot, P.; Kervrann, C.; Waharte, F.; Tinevez, J.-Y.; Shorte, S. L.; Willemse, J.; Celler, K.; van Wezel, G. P.; Dan, H.-W.; Tsai, Y.-S.; de Solórzano, C. O.; Olivo-Marin, J.-C.; Meijering, E. Objective Comparison of Particle Tracking Methods. *Nature Methods* **2014**, *11* (3), 281–289. DOI: 10.1038/nmeth.2808

[85] *VOT Challenge*. https://www.votchallenge.net/index.html (accessed 2024-07-04).

[86] Dung, H. A.; Chen, B.; Chin, T.-J. A Spacecraft Dataset for Detection, Segmentation and Parts Recognition. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021. DOI: 10.1109/cvprw53098.2021.00229

[87] Tang, Q.; Li, X.; Xie, M.; Zhen, J. Intelligent Space Object Detection Driven by Data from Space Objects. *Applied Sciences* **2023**, *14* (1), 333. DOI: 10.3390/app14010333

[88] Sharma, S.; Beierle, C.; D'Amico, S. Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks. In *2018 IEEE Aerospace Conference*, 2018. DOI: 10.1109/aero.2018.8396425

[89] Park, T. H.; Martens, M.; Lecuyer, G.; Izzo, D.; D'Amico, S. SPEED+: Next-Generation Dataset for Spacecraft Pose Estimation across Domain Gap. In *2022 IEEE Aerospace Conference (AERO)*, 2022. DOI: 10.1109/aero53065.2022.9843439

[90] *Turn Images, Videos & Documents into Trustworthy AI*. https://www.v7labs.com/ (accessed 2024-07-04).

[91] *Best in-class labeling tools to create high-quality training data*. https://encord.com/annotate/ (accessed 2024-07-04).

[92] *Open Data Annotation Platform*. https://www.cvat.ai/ (accessed 2024-07-04).

[93] *The data factory for next gen AI*. https://labelbox.com/ (accessed 2024-07-04).

[94] Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Munich, Germany, October 5–9, 2015. DOI: 10.1007/978-3-319-24574-4_28

[95] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 27–30, 2016. DOI: 10.1109/cvpr.2016.91

[96] Pokhrel, S. *Image Data Labelling and Annotation — Everything You Need to Know*. https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1 (accessed 2024-07-04).

[97] Dendorfer, P.; Rezatofighi, H.; Milan, A.; Shi, J.; Cremers, D.; Reid, I.; Roth, S.; Schindler, K.; Leal-Taixe, L. *CVPR19 Tracking and Detection Challenge: How crowded can it get?* arXiv, 2019. https://arxiv.org/pdf/1906.04567 (accessed 2024-07-04).

[98] *Welcome to the E-POP Data Set*. https://epop-data.phys.ucalgary.ca/ (accessed 2024-07-04).

[99] Rezvani, S.; Wang, X. A Broad Review on Class Imbalance Learning Techniques. *Applied Soft Computing* **2023**, *143*, 110415. DOI: 10.1016/j.asoc.2023.110415

[100] Kuhn, H. W. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly* **1955**, *2* (1–2), 83–97. DOI: 10.1002/nav.3800020109

[101] Emami, P.; Pardalos, P. M.; Elefteriadou, L.; Ranka, S. Machine Learning Methods for Data Association in Multi-Object Tracking. *ACM Computing Surveys* **2020**, *53* (4), 1–34. DOI: 10.1145/3394659

[102] Sun, S.; Akhtar, N.; Song, H.; Mian, A. S.; Shah, M. Deep Affinity Network for Multiple Object Tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2019**, 1–1. DOI: 10.1109/tpami.2019.2929520

[103] Yao, Y.; Smal, I.; Grigoriev, I.; Akhmanova, A.; Meijering, E. Deep-Learning Method for Data Association in Particle Tracking. *Bioinformatics* **2020**, *36* (19), 4935–4941. DOI: 10.1093/bioinformatics/btaa597

[104] Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering* **1960**, *82* (1), 35–45. DOI: 10.1115/1.3662552

[105] Fortmann, T.; Bar-Shalom, Y.; Scheffe, M. Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association. *IEEE Journal of Oceanic Engineering* **1983**, *8* (3), 173–184. DOI: 10.1109/joe.1983.1145560

[106] Chianelli, G.; Kunalakantha, P.; Myhre, M.; Lee, R. S. K. A Dual-Purpose Camera for Attitude Determination and Resident Space Object Detection on a Stratospheric Balloon. *Sensors* **2023**, *24* (1), 71. DOI: 10.3390/s24010071

[107] David, J. E.; Byvik, C. *What's up There, Where Is It, and What's It Doing? The U.S. Space Surveillance Network.* https://nsarchive.gwu.edu/briefing-book/intelligence/2023-03-13/whats-there-where-it-and-whats-it-doing-us-space-surveillance (accessed 2024-06-18).

[108] *Ground-Based Electro-Optical Deep Space Surveillance.* https://www.spaceforce.mil/About-Us/Fact-Sheets/Fact-Sheet-Display/Article/2197760/ground-based-electro-optical-deep-space-surveillance/ (accessed 2024-06-18).

[109] *Space Based Space Surveillance*. https://www.spaceforce.mil/About-Us/Fact-Sheets/Article/2197743/space-based-space-surveillance/ (accessed 2024-06-18).

[110] Kelso, T. S. *Space Surveillance*. https://celestrak.org/columns/v04n01/ (accessed 2024-06-18).

[111] *Satellite Technology*. https://www.starlink.com/technology (accessed 2024-06-18).

[112] Low, D.; Koulas, C.; Giovanni, D. D. Training Neural Networks to Detect Resident Space Objects using Space Based Optical Payloads and Low-SWaP Onboard Processing. In *Proceedings of the Advanced Optical Maui Optical and Space Surveillance (AMOS) Technologies Conference,* Maui, HI, USA, September 27–30, 2022.

[113] Qashoa, R.; Suthakar, V.; Chianelli, G.; Kunalakantha, P.; Lee, R. S. K. Technology Demonstration of Space Situational Awareness (SSA) Mission on Stratospheric Balloon Platform. *Remote Sensing* **2024**, *16* (5), 749. DOI: 10.3390/rs16050749

[114] *Strato-Science 2023 Campaign*. https://www.asc-csa.gc.ca/eng/sciences/balloons/campaign-2023.asp (accessed 2024-06-18).

[115] *About STRATOS, the CSA's Stratospheric Balloon Program*. https://www.asc-csa.gc.ca/eng/sciences/balloons/stratos.asp (accessed 2024-06-18).

[116] Manis, A.; Arnold, J. A.; Murray, J.; Buckalew, B.; Cruz, C.; Matney, M. An Overview of Ground-based Radar and Optical Measurements Utilized by the NASA Orbital Debris Program Office. In *2nd International Orbital Debris Conference*, Sugar Land, TX, USA, December 4–7, 2023.

[117] Curti, F.; Spiller, D.; Schiattarella, V.; Orsi, R. Recognition of Orbiting-Objects through Optical Measurements of Light-reflecting-targets by using Star-sensors. In *1st IAA Conference on Space Situational Awareness (ICSSA)*, Orlando, FL, USA, January 17, 2017.

[118] Bernander, K. B. A Method for Detecting Resident Space Objects and Orbit Determination Based on Star Trackers and Image Analysis. MSc. Thesis, Uppsala University, Uppsala, Sweden, 2014. https://uu.diva-portal.org/smash/get/diva2:765745/FULLTEXT01.pdf (accessed 2024-06-18).

[119] Delabie, T.; Schutter, J. D.; Vandenbussche, B. Robustness and Efficiency Improvements for Star Tracker Attitude Estimation. *Journal of Guidance, Control, and Dynamics* **2015**, *38* (11), 2108–2121. DOI: 10.2514/1.g000894

[120] Sarvi, M. N.; Abbasi-Moghadam, D.; Abolghasemi, M.; Hoseini, H. Design and Implementation of a Star-Tracker for LEO Satellite. *Optik* **2020**, *208*, 164343. DOI: 10.1016/j.ijleo.2020.164343

[121] Erlank, A. O. Development of CubeStar A CubeSat-Compatible Star Tracker. MSc. Thesis, Stellenbosch University, Stellenbosch, South Africa, 2013. https://core.ac.uk/download/pdf/37420644.pdf (accessed 2024-06-18).

[122] AlDahoul, N.; Karim, H. A.; De Castro, A.; Tan, M. J. Localization and Classification of Space Objects Using EfficientDet Detector for Space Situational Awareness. *Scientific Reports* **2022**, *12* (1). DOI: 10.1038/s41598-022-25859-y

[123] *Raspberry Pi High Quality Camera*. https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/ (accessed 2024-06-18).

[124] *Alvium 1500 C-500*. https://www.alliedvision.com/en/products/alvium-configurator/alvium-1500-c/500/ (accessed 2024-06-18).

[125] *UI-3370CP Rev. 2*. https://www.ids-imaging.us/store_us/ui-3370cp-rev-2.html (accessed 2024-06-18).

[126] *6mm Wide Angle Lens for Raspberry Pi HQ Camera CS.* https://www.pishop.ca/product/6mm-wide-angle-lens-for-raspberry-pi-hq-camera-cs/ (accessed 2024-06-18).

[127] *16mm Telephoto Lens for Raspberry Pi HQ Camera CS.* https://www.pishop.ca/product/16mm-telephoto-lens-for-raspberry-pi-hq-camera-cs/ (accessed 2024-06-18).

[128] *25mm 5MP Lens CS Mount Fixed Iris for HQ Camera*. https://www.pishop.ca/product/25mm-5mp-lens-cs-mount-fixed-iris-for-hq-camera/ (accessed 2024-06-18).

[129] *Raspberry Pi Zero 2 W*. https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/ (accessed 2024-06-18).

[130] *Raspberry Pi 4 B 4G Computer Board*. https://ca.robotshop.com/products/raspberry-pi-4-b-4g-computer-board (accessed 2024-06-18).

[131] *Thermocouple Type-K Glass Braid Insulated Stainless Steel Tip*. https://www.adafruit.com/product/3245 (accessed 2024-06-18).

[132] *Thermocouple Amplifier MAX31855 breakout board (MAX6675 upgrade).* https://www.adafruit.com/product/269 (accessed 2024-06-18).

[133] Liebe, C. C. Star Trackers for Attitude Determination. *IEEE Aerospace and Electronic Systems Magazine* **1995**, *10* (6), 10–16. DOI: 10.1109/62.387971

[134] Sun, T.; Xing, F.; Wang, X.; You, Z.; Chu, D. An Accuracy Measurement Method for Star Trackers Based on Direct Astronomic Observation. *Scientific Reports* **2016**, *6* (1). DOI: 10.1038/srep22593