Extending the Range of Depth Cameras using Linear Perspective for Mobile Robot Applications

Tasneem Naheyan

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 COMPUTER SCIENCE AND ENGINEERING

YORK UNIVERSITY

TORONTO, ONTARIO

MARCH 2023

Abstract

Reliable depth sensing is essential in robotics for both basic and advanced robot operations. Depth cameras capture depth data that can be used by a robot's vision system, but the quality of the data is limited. Many depth estimation and completion algorithms have been introduced to process camera data and predict depth in a scene, but extending camera range is a little explored problem. This thesis presents a geometry-based method that applies a Manhattan constraint and regresses onto sparse depth input to interpolate and extrapolate lines in the scene in order to extend range. To evaluate the proposed approach, a long-range RGBD dataset with corresponding LiDAR ground truth is presented. Experiments demonstrate that the proposed method successfully interpolates and extrapolates detected 3D lines in Manhattan scenes given sparse depth data within a few centimeters of error, providing depth information in parts of the scene missing input depth data from the sensor. The proposed approach performs comparably to a baseline method in interpolating depth and outperforms it in extrapolation.

Acknowledgements

I would like to express my sincere gratitude to those who supported me while I worked on my Master's research. First, I would like to acknowledge and thank my supervisor, Dr. James Elder, for his ongoing mentorship, patience, and guidance throughout my research project.

I would also like to acknowledge Helio Perroni Filho, the Senior Robotics Engineer of the Elder Lab, for his professional guidance and his patience in answering all of my questions. I want to extend my gratitude to my fellow labmates, for their assistance in conducting experiments and for making my Master's experience enjoyable.

Finally, I would like to thank my friends and family for their never-ending support and encouragement.

Contents

List of Figures

List of Tables

1 Introduction

Recent advances in commercial depth cameras have made them applicable to a variety of computer vision tasks, including 3D simultaneous localization and mapping (SLAM) and machine perception for autonomous robotics. In addition to being a more affordable alternative to high-quality LiDAR for depth sensing, depth cameras also return denser depth estimates. However, they provide reliable depth estimates within a limited range. For example, Intel's RealSense D455 (depth) cameras perform reasonably well in the 0.6 to 6 m range. Humans on the other hand can perceive beyond this range by taking into account both monocular and binocular visual cues. With this as inspiration, my thesis addresses the challenge of extending the range of depth cameras by incorporating the monocular cue of linear perspective. Linear perspective applies when features of a scene are arranged according to one or more systems of parallel lines [1]. This monocular cue is particularly useful in urban environments, which generally consist of families of parallel planar surfaces that generate systems of parallel 3D lines.

While there has been much research on recovering 3D information from monocular images using both classical computer vision and more recent deep learning techniques, the problem of extending depth range is relatively unexplored. Extending the range of depth cameras is useful as it can improve a robot's 3D SLAM capabilities, enabling smooth and reliable autonomous navigation. Accurate depth perception will also result in better scene understanding and human activity comprehension.

A variety of datasets have been created by researchers to test their approaches to depth estimation, but I found these existing datasets to be limited in terms of range, type of scene or availability of ground truth data and unsuitable to the specific problem presented in this thesis: extending the range of depth cameras for mobile robot applications. Consequently, my thesis makes two main contributions: 1) a dataset consisting of long-range RGBD images with corresponding LiDAR ground truth to evaluate depth extension and completion approaches, 2) a novel interpretable approach to extend depth camera range given sparse depth input. The approach recovers depth in the built environment by exploiting the Manhattan constraint, the assumption that a scene contains three sets of mutually orthogonal systems of orthogonal line segments converging to three vanishing points in the image. The 3D orientation of lines detected in the RGB image can then be determined by detecting these vanishing points. While the

distance and scale of each 3D line segment is unknown, these can potentially be estimated in an RGBD image by regressing onto the sparse depth input from the camera. The line segments can then serve to interpolate and/or extrapolate depth between and beyond these sparse depth estimates.

We evaluate our approach on our novel dataset and find that the proposed algorithm produces relatively reliable depth estimates in interpolated and extrapolated intervals of lines compared to intervals with support from input depth data. We also compare the method to a baseline method and find that it interpolates comparably and extrapolates more successfully than this method.

This thesis is organized as follows. Section 1.1 describes the motivation for the thesis topic. Chapter 2 is an overview of related literature in depth estimation, depth enhancement and depth completion. Chapter 3 presents a survey of existing datasets for testing depth algorithms, and details of the dataset created to evaluate the proposed method. Chapter 4 introduces the proposed method to extend the depth range of a depth camera. Chapter 5 reports experimental results. Chapter 6 discusses findings and Chapter 7 describes potential future work.

1.1 Project Context

SentryNet Project

The motivation for my research topic came while working on the initial stages of the SentryNet project. The SentryNet project (September 2020 – October 2022) is a multifaceted project exploring technical methods to develop trust in human-robot teams. The overall goal is to build a trustworthy robot to function as a sentry in civilian and military settings. The Elder Laboratory's responsibility in SentryNet was sensing for human-robot interaction. This can be divided into sensing for physical scene understanding and human activity understanding. Reliable sensing is an initial step for any autonomous system; the sentry robot must understand its 3D environment to perform tasks such as navigation and interaction with users. While running experiments for physical scene understanding, we found the range of the RealSense[42] cameras to be a limiting factor because we wanted the robot to navigate large indoor spaces but the cameras provide reliable depth data only up to 6 m. This prompted the subtopic of research in extending depth sensing, which is the focus of my thesis.

Early stages of the SentryNet project involved setting up the robot platform for basic operations: sensing the physical environment in real-time, robot localization with respect to a stored map and navigation with obstacle avoidance. The team began development with a modular prototype consisting of a Dingo [2] omnidirectional robot base from Clearpath Robotics with external hardware mounted on it on a custom support frame (Figure 1.1). This prototype can be easily modified to add or remove components as required. Our work will ultimately be migrated to the commercial CrossWing Cleanbot platform (Figure 1.1).

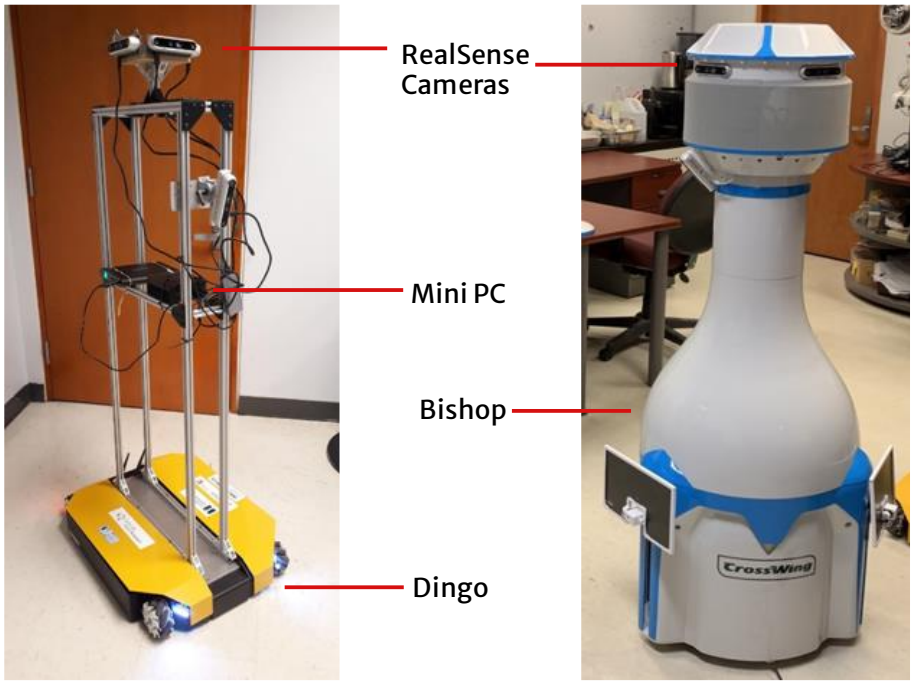


Figure 1.1: Dingo robot prototype (left) and Cleanbot robot (right).

The Dingo prototype employs five RealSense D455 stereo depth cameras; four at the top at a height of approximately 160 cm, and the fifth angled downwards at 45° for near-space sensing. Each RealSense camera has an 87° horizontal field of view; together the four cameras provide a panoramic field of view with four narrow 3° gaps. The cameras capture RGB images and depth maps with 1280x720 resolution. SLAM, navigation and object detection algorithms are run on the onboard computing platform or a remote server depending on the computational requirements of the algorithms.

We programmed the robot to perform simple tasks e.g. to autonomously tour a room, detect a person and move to them. ROS and Python were used for software implementation. For basic navigation there are two stages: map formation and autonomous task execution. In the first stage, the robot is teleoperated around the environment to create a global map using the depth data from the RealSense cameras. Localization and mapping were done using a real-time appearance based mapping algorithm [3]. Depth estimates from the cameras were projected to the 2D floor plane; these projected depth estimates are visible in Figure 1.2 as red contour fragments.

In the second stage this 2D depth representation is used to update the robot's local map, which covers the robot's immediate surroundings, and detect and avoid new obstacles that appear in its path. The global and local map provide the robot's location with respect to the map to the ROS package managing the robot's movement, enabling it to move to a new location when given a target position, e.g. the position of a detected person.

For the person detection task, people are detected in the RealSense color images using the YoloV3 [4] object detector and localized in 3D in the near-field using the corresponding depth maps. The 3D coordinates of the detected people are converted to ground plane coordinates in the robot-centric frame using camera intrinsic and extrinsic parameters. These coordinates can then be input as target goals for the robot. While performing initial tests, we found that the range of the cameras limited the scenario to the near field; there were no depth returns in the depth map for people or obstacles beyond the camera range of around 6 m and so they could not be reliably detected. This is illustrated in Figure 1.2: there is no depth data for the people standing further than 6 m away from the front camera, beyond where the red 2D depth representation appears on the map. If the focal length is known, the location of the people beyond the range of the cameras can be estimated by assuming the bottom of the person detection bounding box is on the ground plane. However, this method is not ideal as it assumes that the ground plane is flat. Depth data from cameras is needed for more accurate localization.

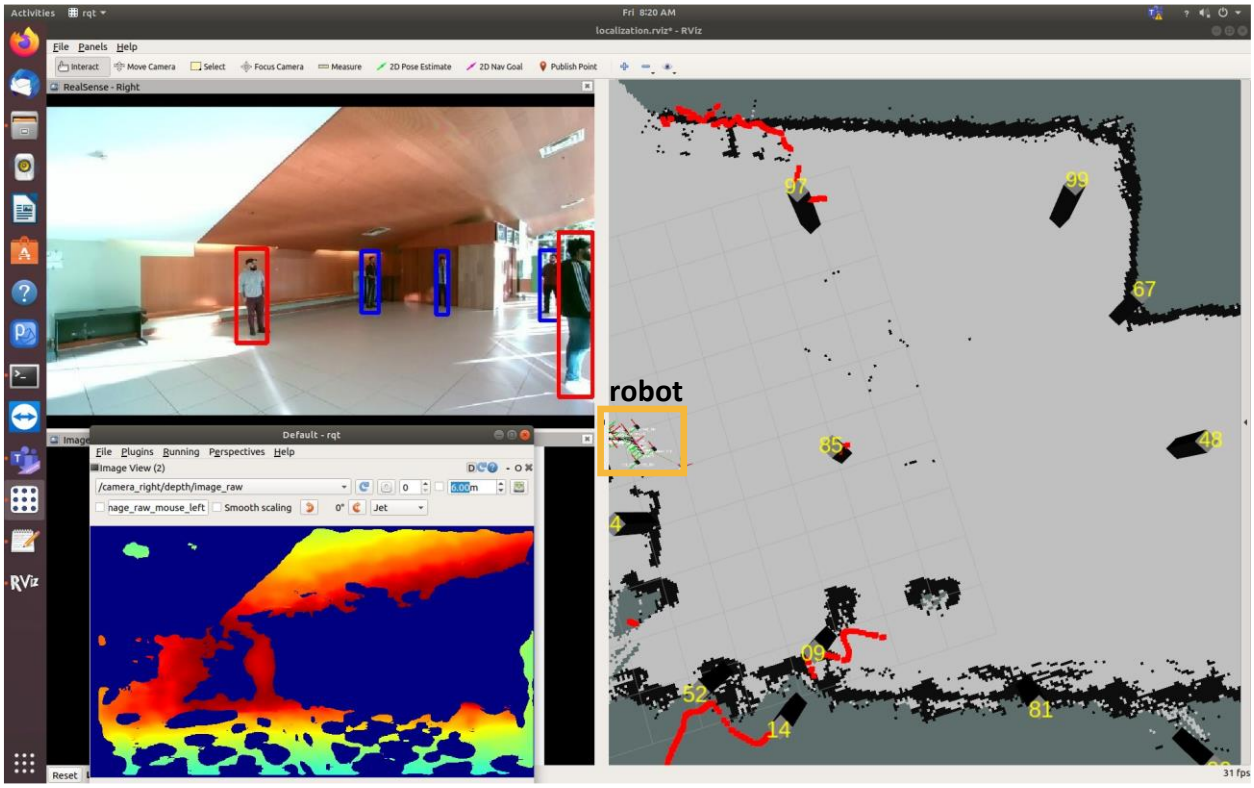


Figure 1.2: Top-left: color image from our Dingo platform's front RealSense camera: person detections with depth returns have red bounding boxes while those without have blue bounding boxes. Bottom-left: corresponding depth map, there are no depth returns in the navy blue regions. Right: Top-down view of a map of the environment. Block cuboids mark the locations of the people present.

To reliably detect obstacles, people, or any kind of target in large spaces and be aware of the physical environment, the robot's vision system should support extended depth sensing. Raw data from depth cameras is generally not adequate at longer ranges so development of an algorithm to make use of the raw data and generate improved depth estimates is useful for improving robot functionality. Work done for this thesis builds towards improving physical scene understanding in long-range spaces and is important to achieve the SentryNet project's long-term goal of designing a robot with robust sensorimotor functionality that supports navigation and interaction in social environments.

2 Related Work

Little research has looked specifically at extending the range of depth cameras, so here I have summarized the literature on the related and relevant research areas of monocular depth estimation and depth completion.

2.1 Visual Cues

People perceive depth in the near field by stereopsis, and we are also able to estimate depth quite well at longer ranges by incorporating monocular cues and prior knowledge, with exceptions. The Ames room illusion is an example of where human monocular depth perception fails; observers view the specially constructed room through a peephole with one eye and it appears as a normal rectangular room despite actually having a trapezoidal shape. Sense of depth is lost by viewing the room with only one eye, but a person standing at one corner of the room appears larger to the observer than a person standing at the other corner which is further away while the room retains its rectangular shape [52]. Prior work has studied the use of visual cues to estimate depth from monocular images.

Monocular cues to depth include relative size, texture gradient, linear perspective, interposition, atmospheric cues, shading and image height [6, 5]. The cue of relative size reflects the fact that as an object approaches the observer, its projection in the image expands (considering 2D motion towards the observer). Texture gradients also provide a relative size cue as well as a cue to 3D surface orientation. Relative size and texture gradients are related to linear perspective, which exploits the fact that parallel lines converge when projected to the image. Interposition occurs when an object is partially occluded by another; the occluded object is perceived to be further way. Atmospheric cues arise from the scattering of blue light in the atmosphere which makes distant objects appear blurred and bluish. Shading arises as surface orientation changes relative to the light source. The image height cue reflects the fact that as objects lying on the ground plane get further away, they project to a higher location in the image. Further details on monocular cues are provided in [6, 5]. Human vision takes into account all of these cues to interpret the depth in a 2D image. Depth estimation algorithms attempt to incorporate some of these cues or simulate prior knowledge.

A number of prior approaches make use of the Manhattan World assumption to estimate depth from 2D images. The Manhattan constraint can be defined in terms of an i, j, k cartesian

coordinate system on which most built environments are based [55]. The constraint assumes the scene is organized according to three sets of mutually orthogonal parallel lines. If we can estimate the directions $i, j,$ and k of these lines, it becomes easier to detect the prominent lines e.g. boundaries of walls and floor, in the scene since they would mostly lie along these directions. The three sets of parallel lines in the 3D scene converge to three vanishing points in the image (Figure 2.1). We can detect the vanishing points by finding the intersection of each set of parallel lines and compute the unit vectors corresponding to each point. Then by assuming that the three orthogonal planes of the Manhattan World are aligned with the x, y, and z axes of the world coordinate system, we can estimate the orientation of the world in relation to the camera using these unit vectors corresponding to the vanishing points. With this assumption it is possible to estimate the 3D orientation of the lines but not their true scale or depth.

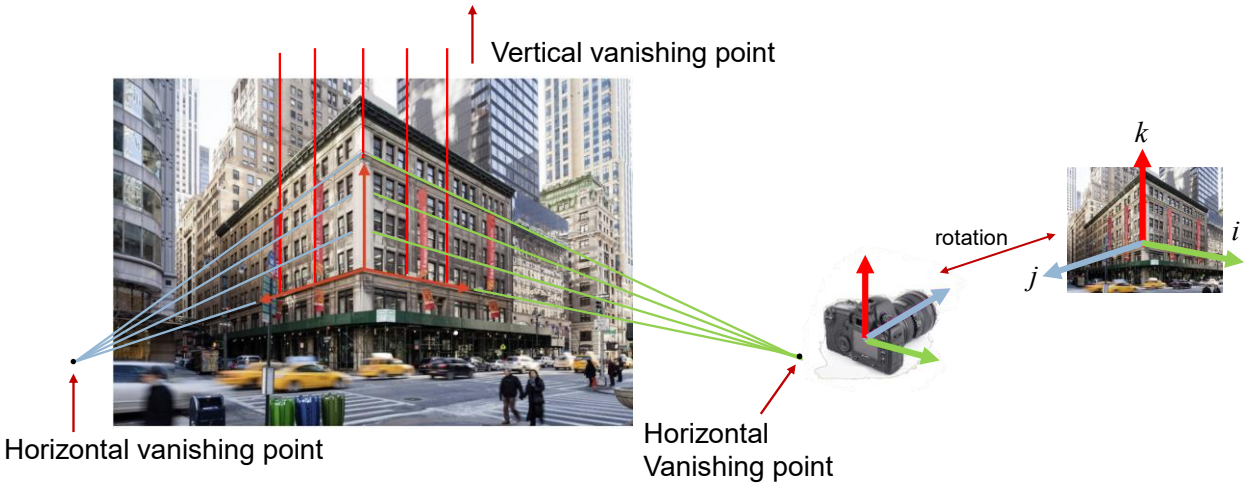


Figure 2.1: Manhattan World assumption: the scene is organized in three orthogonal sets of parallel lines; each set converges on a vanishing point when projected to the image plane. The vanishing points can be used to determine the relative orientation of the scene.

2.2 Depth Estimation

Many monocular depth estimation algorithms have been proposed to estimate depth from a single image. Estimating depth from a single image is challenging since the problem is ambiguous: multiple 3D scenes can produce the same 2D image [5]. Early work in depth

estimation made use of hand-crafted features, visual cues and assumptions while more recent work has relied on deep learning.

Traditional Monocular Depth Estimation

Early work explored a variety of techniques for depth estimation. Several methods employ handcrafted filters, assumptions and loss functions in MRF-based models [5]. Hoiem et al. introduced a framework to estimate 3D scene structure from a single image; their model applies the explicit assumptions that a scene is composed of a horizontal floor and vertical surfaces standing on it, and returns confidence maps for the main geometric labels e.g. “ground”, “sky”, “vertical” [7]. Saxena et al. introduced a discriminatively-trained Markov Random Field (MRF) model that uses image features at different scales to capture monocular cues [8]. Their method divides an image into patches and applies custom filters to extract image features to estimates a depth value for each patch. Local image features are not enough to estimate absolute depth, so they use image features extracted at multiple image resolutions. They used an MRF to model the relation between the depth of a patch and depths of neighboring patches, and proposed a Gaussian MRF model as well as a better performing Laplacian MRF model.

Building on their work in [8], in a later paper Saxena et al. introduced a 3D reconstruction method that uses an MRF model and supervised learning to learn monocular cues, relations between monocular cues and 3D structure in the scene and relations between different sections of the image [9]. Their algorithm segments an input image into small planar surfaces by splitting the image into superpixels and calculating the 3D position and orientation of the surface the superpixel most likely lies on. Their proposed MRF model captures four properties: the image features of a superpixel are related to the depth of the superpixel, neighboring superpixels are likely connected to each other, neighboring superpixels likely belong to the same plane if they have similar features and no edges separating them, and long straight lines are probably straight lines in the final 3D model. All four properties are considered together to predict relative depth. The model is trained to estimate depth with a LiDAR training set. The authors claim that since their approach only makes the assumption that the scene is composed of small planes, it is able to generalize better to scenes with less vertical structure compared to other traditional methods [7, 10] that apply stronger assumptions e.g. the scene contains a ground plane, vertical walls, the sky, etc. They report that their method produced qualitatively correct 3D models for

64.9% of 588 test images. The approach in [9] introduced useful techniques in depth estimation: 1) dividing the image into superpixels and estimating depth: this reduces computational complexity, and 2) the coplanarity assumption: superpixels with similar RGB properties likely lie on the same surface. This assumption smooths a model's estimation.

Other approaches have incorporated semantic segmentation to estimate depth from a single monocular image; [11] developed an algorithm that predicts the semantic class of pixels and the location of the horizon in the image, and then uses this semantic and geometric context to estimate depth.

More recently Qian et al. introduced the Line-Segment-to-3D (LS3D) algorithm which applies linear perspective, Manhattan constraints and the Gestalt principle of proximity to generate a 3D CAD model of a building [1]. LS3D is a method for single-view 3D reconstruction that assumes three mutually orthogonal systems of parallel lines and applies the Manhattan constraint to label line segments with their 3D orientation. Groups of line segments can be connected to produce Manhattan tree representations by applying the Gestalt principle of proximity, which states that lines closer together are perceived to be more related than those that are further apart [53]. The LS3D algorithm creates a dense graph with each line segment as a vertex; edges connect orthogonal segments separated by less than a threshold distance [1]. It applies a sparsity constraint to ensure that each endpoint connects to at most one other endpoint in the two orthogonal Manhattan directions through a sequence of optimal bipartite matchings. The algorithm then calculates minimum spanning trees to sparsify the graph further and produce Manhattan trees. The Manhattan trees are lifted to 3D using Manhattan direction constraints and the algorithm identifies each tree's 3D three-junctions and three-paths to define minimal spanning cuboids which form the solid model of the trees. L1 minimization is applied to calculate the relative depth of the models. With this process the LS3D algorithm reconstructs 3D Manhattan buildings up to a single scaling parameter. It performs better than prior machine learning and deep learning methods for such buildings. Incorporating Manhattan tree representations generated by LS3D to the method proposed in this thesis to extend depth given sparse depth data is a direction of future work.

Deep Learning Based Monocular Depth Estimation

Recently many deep learning methods have been proposed for monocular depth estimation, and most make use of convolutional neural networks (CNNs). Supervised machine learning methods learn to estimate depth from a training set of images with corresponding ground truth depth collected using depth sensors such as RGBD cameras or laser scanners. The networks require a large quantity of high-quality depth data for training. Methods employing deep CNNs often are composed of an encoder for dense feature extraction paired with a decoder for predicting the depth map. For example, Laina et al. introduced a fully convolutional residual network that is trained end-to-end with novel up-sampling blocks that requires few parameters and less training data, but outputs low resolution depth maps of maximum resolution 160x128 pixels that are up-sampled by bilinear interpolation back to original input size [12]. Ibraheem et al introduced a transfer learning-based network that produces higher quality depth maps [13]. It uses the DenseNet-169 network pretrained on ImageNet, a network originally built for image classification, as a deep feature encoder. The decoder, consisting of upsampling layers and associated skip-connections, constructs the predicted depth map at half the input image resolution. The network is trained with RGBD data augmented by horizontal flipping and color distortion. RGBD datasets collected with commercial sensors contain noisy depth data and this limits the performance of learning methods that apply local geometric constraints such as surface normals in their pipeline. To address this Yin et al. proposed a global 3D geometric constraint, a *virtual normal,* in their approach. The *virtual normal* is the normal vector of a virtual plane created by randomly sampling three non-colinear points in 3D space [14]. Their loss term enforces this longer-range geometric constraint, resulting in more accurate depth predictions.

Self-supervised and semi-supervised methods factor in additional input during training e.g. stereo images or visual odometry, so they require less labelled training data and are less affected by errors and noise in the ground truth data. These methods require additional information such as camera parameters and are unable to correct their own bias [15]. Kuznietsov et al. proposed a semi-supervised method that uses both supervised and unsupervised training cues [16]. They use a deep residual network in encoder-decoder architecture with long skip connections to predict depth with a loss function that includes both supervised and unsupervised cue constraints. The network learns in a self-supervised way through direct alignment between stereo images. Their approach produces detailed and accurate depth maps on the KITTI dataset,

but requires camera calibration information in addition to ground truth data [16]. Bauer et al. introduced NVS-MonoDepth, a semi-supervised approach with a novel training scheme that improves training by using novel view synthesis to address the lack of geometrical consistency in predictions of monocular depth networks from different view points [17]. The training pipeline contains a monocular depth prediction network, a view transformation process, and an image synthesis network. The network is trained with images from two neighboring viewpoints which are predicted during training and serve as additional supervisory signals for training. This approach also requires camera calibration data.

As an alternative to training models with accurate ground truth depth, self-supervised approaches are trained using synchronized stereo pairs or monocular video. Godard et al. introduced an unsupervised CNN architecture MonoDepth which learns depth from stereo pairs by exploiting epipolar geometry constraints and enforcing left-right depth consistency [18]. MonoDepth does not require ground truth depth in training, but it does not handle occlusion well during training and produces inconsistent depth on transparent and specular surfaces. The authors improved their method in MonoDepth2, a self-supervised approach trained with monocular video, stereo pairs or both combined [19]. Their updated approach includes a minimum reprojection loss to better handle occlusions, a full-resolution multi-scale sampling method to reduce visual artifacts and an auto-masking loss to ignore pixels where there is no observed camera motion in training. Guizilini et al. introduced the encoder-decoder network PackNet built with symmetrical, detail-preserving ‘packing’ and ‘unpacking’ blocks for compressing and decompressing high resolution visual data to generate detailed predictions with accompanying skip connections [20]. The network is trained on unlabeled monocular videos, and can optionally include weak velocity supervision during training to generate scale-aware depth models. The addition of the novel blocks increases the network complexity, and the resulting depth maps have correct metric scale only if instantaneous velocity measurements are given as input.

Data-driven depth estimation techniques have their limitations: the depth prediction classifier is biased towards the distribution of depths in the dataset. Reliability of such depth estimation methods depends on whether or not a similar scenario has been seen in the training set. Performance of these models is impacted by the quality of the dataset: sensors typically used

for data collection such as depth cameras and laser scanners can return noisy depth estimates which produce errors.

While monocular depth estimation methods can be cost-effective, they are often insufficient for real world applications due to low accuracy and inability to return the true scale of scenes. Depth sensors return depth estimates that are true to scale, but their direct output can be inadequate. LiDAR for example returns accurate depth estimates at long ranges, but the range estimates are sparse. Depth cameras produce denser depth maps but do not extend to the far field and the maps commonly have missing portions or holes since these cameras fail to return estimates for shiny, transparent, or thin surfaces. Holes can appear on object surfaces, or in larger regions along the boundary between foreground and background objects. These missing regions affect the performance of algorithms using depth maps for computer vision tasks such as 3D reconstruction, scene understanding, and navigation.

2.3 Depth Completion

To address the limitations of depth estimation methods and depth sensors, depth completion algorithms have been proposed. Depth completion algorithms aim to generate dense depth maps from incomplete ones. Techniques to complete depth vary depending on the sparsity of the initial depth map. Depth maps produced by RGBD cameras are relatively dense whereas depth maps generated from LiDAR readings are sparse [21]. Approaches utilizing conventional image processing techniques have been proposed to complete denser depth maps and are commonly referred to as depth enhancement methods.

Depth Enhancement

Depth enhancement techniques aim to fill in holes in depth maps. These methods, often called inpainting methods, can be grouped into diffusion-based and exemplar-based methods [22].

Telea's diffusion-based method fills holes using the Fast Marching Method; holes are filled a pixel at a time by averaging the depth values of pixels surrounding a target hole pixel [23]. The averages are weighted by the distance between hole pixel and pixel being considered. Chen et al.'s method interpolates using a bilateral filter [24] instead of averaging [25]. Their method improves the depth map from a Kinect sensor by detecting pixels with incorrect depth values with a region growing method using depth and color information, filling holes with the

region growing method combined with a bilateral filter, and finally reducing noise in the depth map with an adaptive joint bilateral filter. Cho et al. proposed a technique that fills holes from the hole border towards an object boundary within the hole [22]. The object boundaries are detected from the corresponding color image. Their method sets a strong edge map, fills gaps between edge endpoints and accounts for cases where there are edge pixels between two border pixels. Depth error regions are removed and depth is copied first from a hole border toward an edge pixel horizontally, then the rest of the hole is filled vertically. This process is reiterated until all holes are filled. They apply their method on images from a RealSense camera. The method in [26] applies a series of morphological operations e.g. dilation and closing, and median and gaussian blurring, to fill in empty hole pixels. Diffusion-based techniques are typically simple and fast to implement, working well for small holes. For larger holes these methods can result in blurring and a loss of geometric properties of the region.

[27] and [28] are examples of exemplar-based inpainting techniques. In Criminisi et al.'s patch-based filling approach a hole is filled with the depth values of the most similar patch in the image; the most similar patch is found by scrolling over the image to find the best-matching region [27]. Filling order is important for this method and results are better when holes are filled with patches containing strong edges and the patch is close to the hole. This exemplar-based method works better than diffusion-based methods for filling large holes but is slower. A similar method proposed in [28] reduces computation time by limiting the search region for finding a similar patch to be around the hole border rather than the whole image since relevant image data is usually in neighboring pixels. Such exemplar-based methods are limited in cases where there are no patches similar to the missing portion of the image.

The method in [29] proposed more recently successfully completes sparse depth maps from LiDAR data by exploiting local surface geometry using conventional image processing only. Pixels with unknown depth are assumed to lie on the same surface as a nearby LiDAR point and the algorithm estimates the depth at these pixels as that LiDAR depth value with an added residual computed by considering the orientation of the local surface and the distance on the image plane. Predictions are optimized using a novel outlier removal algorithm to remove incorrect LiDAR points.

Deep Learning Based Depth Completion

Conventional image processing techniques often do not work as well with sparse depth as they do with denser input. Therefore, recent depth completion methods typically apply deep learning to recover depth from sparse depth input.

Deep learning-based depth completion methods predict a dense depth map from a sparse map either with or without guidance from data of another modality, typically RGB. Uhrig et al. first introduced an unguided method incorporating deep learning [30]. They introduce a CNN composed of sparse convolution layers that considers the location of missing data when performing the convolution operation to predict depth given sparse laser scan data. The proposed CNN performs better than traditional CNNs on sparse data. Unguided methods often underperform as it is difficult to recover consistent boundaries without semantic cues from a corresponding color image.

Guided methods use RGB data in various ways: some methods are guided by surface normals to utilize geometric features. Zhang et al. introduced a two-stage method for completing depth maps from RGBD cameras. In the first stage they train a neural network to predict surface normals and occlusion boundaries from a monocular image and in the second stage they apply a global optimization to integrate the predictions from the first stage with pixel depth estimates from the RGBD camera [31]. This method works better for indoor scenes than outdoor scenes. Other methods extract features at different scales to preserve high-level information from deep layers and finer details from shallow layers, revisiting an idea explored in traditional monocular depth estimation. Li et al. proposed a multi-scale guided cascade hourglass network which extracts multiscale guidance information from a color image [32]. Each hourglass in the network receives guidance at different levels to predict depth at different scales, which are then integrated by residual connections. RGB-guided methods are more accurate than unguided methods but are more computationally expensive.

Guided depth completion models employ various data fusion strategies. Early fusion algorithms combine features extracted from the two modalities before passing them into the model for training. These methods take advantage of correlations between features from different modalities but do not factor in unique properties of each modality [21]. Late fusion algorithms model data from each modality separately and integrate the outputs to generate a final prediction.

These methods avoid overfitting but are more computationally intensive and challenging in terms of data alignment.

Depth estimation and completion algorithms all have a similar goal of obtaining a more complete understanding of the depth in the scene. Monocular depth estimation algorithms generally predict relative depth and cannot return the true depth in the scene. Research in depth estimation has highlighted useful concepts such as making use of scene geometry. By employing these ideas with the addition of depth data from sensors such as RGBD cameras and laser scanners, researchers have developed methods that predict complete depth maps at true scale with greater accuracy given sparse input. Deep-learning based methods can produce accurate results but require a large amount of data to perform well and are computationally expensive. Methods making use of traditional techniques are more easily interpretable and some methods have competitive performance.

Depth completion algorithms are designed to complete missing portions of depth maps but to our knowledge, no work has specifically focused on extending the depth range of a sensor. Existing work has found that successful depth estimation requires an understanding of scene geometry, and in this thesis I propose a method that exploits geometric cues to extend the depth range of a depth camera. It does not require a large quantity of training data or extensive computational resources. The method takes as input a color image and corresponding sparse depth data and applies geometric constraints to determine the metric depth and extend it in long range scenes. Chapter 4 describes the proposed method.

3 Datasets

We are interested in the application of the proposed method to large indoor spaces. To evaluate the depth scaling algorithm we require a dataset of relatively large indoor scenes, containing RGBD data as well as ground truth depth data, e.g. from LiDAR, for evaluation. Several datasets have been created for evaluation of depth estimation and completion methods. Various depth sensors are used to collect RGBD images such as structured light cameras and stereo camera systems. Ground truth depth is collected using LiDAR or determined through algorithms leveraging multiple views of the same scene. The next section lists some existing depth datasets.

3.1 Survey of Datasets

Among the most commonly used datasets are the Matterport3D, NYUv2, and ScanNet datasets. The Matterport3D dataset is a multiview RGBD dataset containing 194,400 images of scenes from 90 buildings and annotations for surface reconstructions and semantic segmentation captured with the Matterport Pro 3D camera [33]. NYUv2 contains RGBD images from 464 indoor scenes collected with Kinect v1 with labels for semantic segmentation [34]. ScanNet contains 2.5M views of 1513 scenes collected using the Structure sensor (a structured light sensor) and reconstructed 3D models with annotated objects [35]. These datasets have enough images for training machine learning models, but do not have ground truth that reflects the true depth in the scenes, and the images are mostly of short-range indoor scenes.

Most datasets of long-range scenes that include ground truth LiDAR scans are outdoor datasets, e.g. the KITTI dataset, a popular benchmark dataset consisting of longer range traffic scenes collected using a multiple-sensor setup including GPS/IMU, LiDAR, grayscale and color cameras [36]. The KITTI-Depth dataset was created to evaluate depth prediction and completion tasks and contains sparse depth maps in addition to raw Velodyne LiDAR scans and RGB images but these depth maps are not independent from the raw depth data and were created from comparing projected LiDAR point clouds to depth estimated from stereo cameras using semi-global matching (SGM) [30]. ETH3D is a dataset of man-made and natural scenes; it has DSLR image sequences from four synchronized global-shutter cameras forming two stereo pairs, with ground truth from the Faro Focus X 330 laser and sparse depth maps rendered from the LiDAR point clouds [37]. However only two out of the five sets of stereo data are of short-range indoor

scenes. ARKitScenes is currently the largest indoor scenes dataset [38]. It is a collection of IMU, RGB and depth images collected using the 2020 iPad Pro which comes equipped with a TOF 3D LiDAR sensor, with corresponding ground truth point clouds from Faro Focus S70. The images are of 1661 indoor household scenes. The indoor images in ETH3D and ARKitScenes do not cover a wide enough range for our purposes.

A dataset coming close to meeting our requirements is the indoor OdomBeyondVision dataset [39]. Data were collected on three platforms: handheld, UAV (Unmanned Aerial Vehicle) and UGV (Unmanned Ground Vehicle) in different indoor scenes under varied illumination. For our purposes the UAV sequences are most useful. The UGV platform consists of an Intel RealSense D435i stereo camera and Velodyne HDL-32E LiDAR as well as a thermal camera, radar and IMU mounted on a Turtlebot [39]. The recorded sequences contain data collected as the Turtlebot moved through different scenes. Since this dataset was created to test ego-motion estimation methods, there is not a lot of variety in the types of scenes, most images are of similar-looking hallways. The scenes also do not cover a very wide range of depths; the maximum range in the scenes is around 20 m, but most parts of the scene are within 6 m from the UGV platform. The dataset is still in development and in my experiments I found large errors in the extrinsic calibration.

Preliminary experiments were conducted on the Matterport3D dataset, supplemented with "ground truth" from the Deep Depth Completion dataset [31]. Matterport3D contains multiple views of each scene. In [31] the authors use the multiple views to create a global reconstruction via Poisson surface reconstruction. They then generate a completed depth map for a selected RGB image by rendering a reconstructed mesh. The resulting dataset contains 117,516 RGB-D images with complementary aligned completed depth images from 72 real-world indoor environments. 64.6% of the missing pixels in the raw depth image are filled by reconstruction and the completed depth images have better resolution for distant surfaces and are less noisy than the raw depth maps. Two aspects of this dataset leave it ill-suited to the problem of extending depth range explored in this thesis. First, the range of depths is limited and the maximum depth is between 5 and 10 m for most images, and many images only show a portion of a wall or ceiling. Second, while the reconstruction process in [31] is useful for multi-view datasets where no ground truth data was collected and is less costly and time-consuming than collecting data

with another depth sensor with better accuracy, the ground truth in the completed depth maps is not independent of the depth estimates and this could bias evaluation.

Many long-range datasets with supporting LiDAR ground truth capture outdoor scenes via a car set up with sensors driving through the area [36, 40], but there are few long-range indoor datasets. Since I could not find a dataset that satisfactorily met my criteria for evaluation (indoor scenes with large range of depth, with ground truth depth independent of camera returns), I collected a small dataset, which I will call the York-RGBD-LiDAR dataset, as part of my thesis to carry out experiments.

3.2 York-RGBD-LiDAR Dataset

Data were collected using the Dingo robot platform (Figure 1.1) with four RealSense D455 cameras and an Ouster OS1 LiDAR [41] mounted on it; the setup is shown in Figure 3.1.

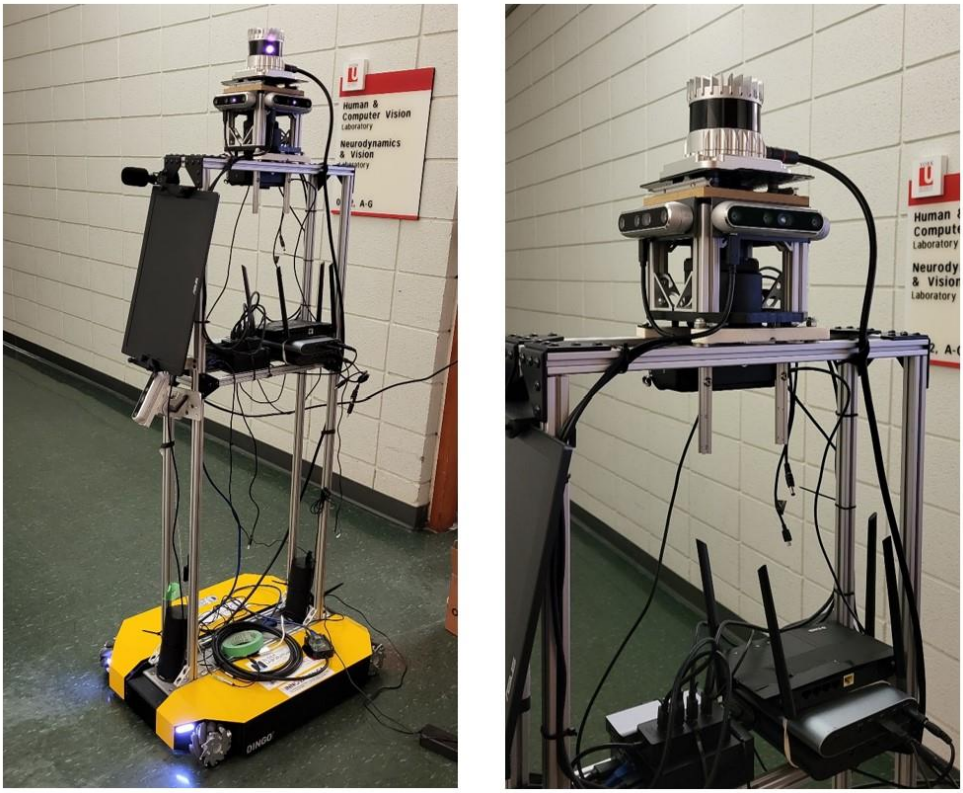


Figure 3.1: Dingo robot platform with Ouster OS1 LiDAR mounted on it.

The LiDAR is placed above the cameras. The center of the LiDAR sensor is 11 cm vertically above the plane through the center of each camera sensor and each camera is 8 cm

horizontally in front of the center of the LiDAR. The Ouster OS1 has a range of 0.5 to 90 m, 360° horizontal field of view and 45° vertical field of view, and up to ± 0.7 cm precision [41]. The range accuracy is quoted to be ±3 cm for lambertian targets and ±10 cm for retroreflectors.

The RealSense cameras have an ideal range of 0.6 to 6m and nominal depth error of less than 2% at 4m. The depth horizontal and vertical fields of view are 87° and 58° respectively and the color sensor's horizontal and vertical fields of view are 90° and 65° respectively [42]. The depth start point, or the starting point or plane where depth is equal to 0 m, is referenced from 4.55 mm behind the front of the camera cover glass (Figure 3.2) [56]. The RealSense camera contains wide stereo left and right imagers placed on either end of the camera, a wide infrared projector, an RGB color sensor and a 6 degrees of freedom inertial measurement unit (IMU) [56]. The origin of the depth channel coordinate frame is aligned with the left imager and 4.55 mm behind the front cover glass (Figure 3.3). The offsets between the depth and RGB sensors are accounted for in the pre-specified intrinsic parameters included in the camera API. The camera compares the two images from the left and right imagers and uses the distance between the imagers to derive depth from disparity, similarly to how humans perceive depth by stereopsis. The infrared projector improves the stereo camera system's ability to measure depth in low lighting; it projects a static infrared pattern on the scene to increase texture on scenes with low texture. The color sensor produces a color image and provides texture information. The camera performs relatively well in indoor settings, but the output depth maps have missing portions where there are transparent and near/distant surfaces beyond the camera range. Shiny and reflective surfaces also do not generate depth returns because they saturate the IR sensor.

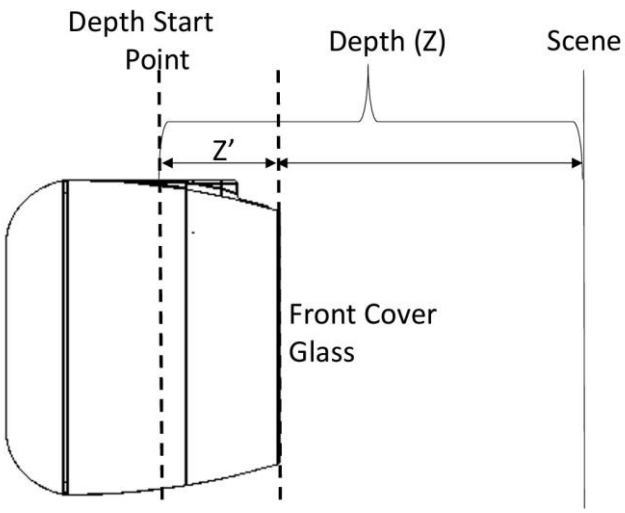


Figure 3.2: Side view of RealSense D455 camera; depth is measured from a distance Z' = 4.55 mm inside the front cover glass [pg. 89, 56].

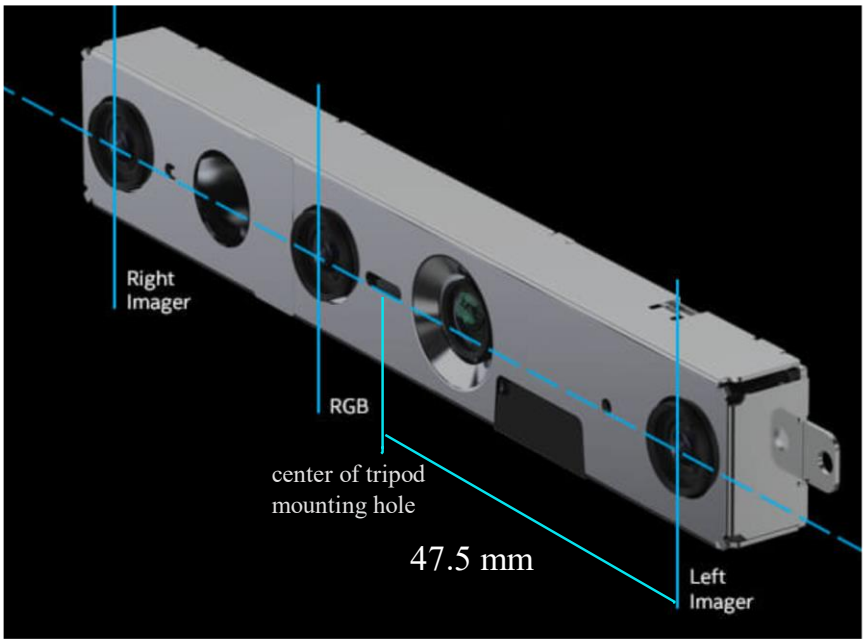


Figure 3.3: RealSense D455 camera: the optical center for the depth camera is 4.55 mm behind the left imager, which lies 47.5 mm to the left of the central tripod mounting hole [pg. 92, 56]. Picture adapted from [42].

The four cameras and LiDAR were launched from the robot's onboard computer using ROS and data (time-synchronized color and depth, LiDAR point clouds and camera intrinsic parameters) were collected from all sensors simultaneously in rosbag format. Each sensor has its own clock and the components of the data collection system are connected via ROS and USB.

Perfect time synchronization is difficult to achieve in such a system, so for the purpose of collecting this dataset we assumed a static world, stopped the robot and recorded data within a small temporal window. Color, depth and LiDAR point cloud topics were extracted from rosbags for each scene according to timestamp within a small uncertainty margin. The RealSense color and depth sensors have slightly different origins and fields of view and to account for this the depth map is aligned to the color image through an internal step in a ROS nodelet. The raw depth map is aligned to the color image by reprojecting the depth map to the color sensor's coordinate frame in two steps: first the depth map is converted to a point cloud using the depth camera intrinsics, then the point cloud is converted back to a depth map using the color camera intrinsics. In the evaluation of our method, the LiDAR data is projected into the camera's reference frame using the extrinsic calibration described later in this section.

Data were collected inside buildings on York University's campus. The robot was moved to each location, positioned to capture images with the greatest possible range, and left stationary while recording images and LiDAR scans. The dataset contains color and depth images of 1280×720 resolution and corresponding LiDAR point clouds for 25 different indoor scenes, with the maximum range of scenes ranging between 30m and 90m. While the RealSense cameras can record depth beyond their quoted range of 6m, these depth estimates are noisy and inaccurate thus we filtered out raw depth estimates beyond 6m. 11 images from the front, 7 images from the right and 7 images from the left cameras are included in the dataset; images captured by the back camera were mostly short-range and therefore are excluded from the dataset. Figure 3.4 displays three images from the dataset.

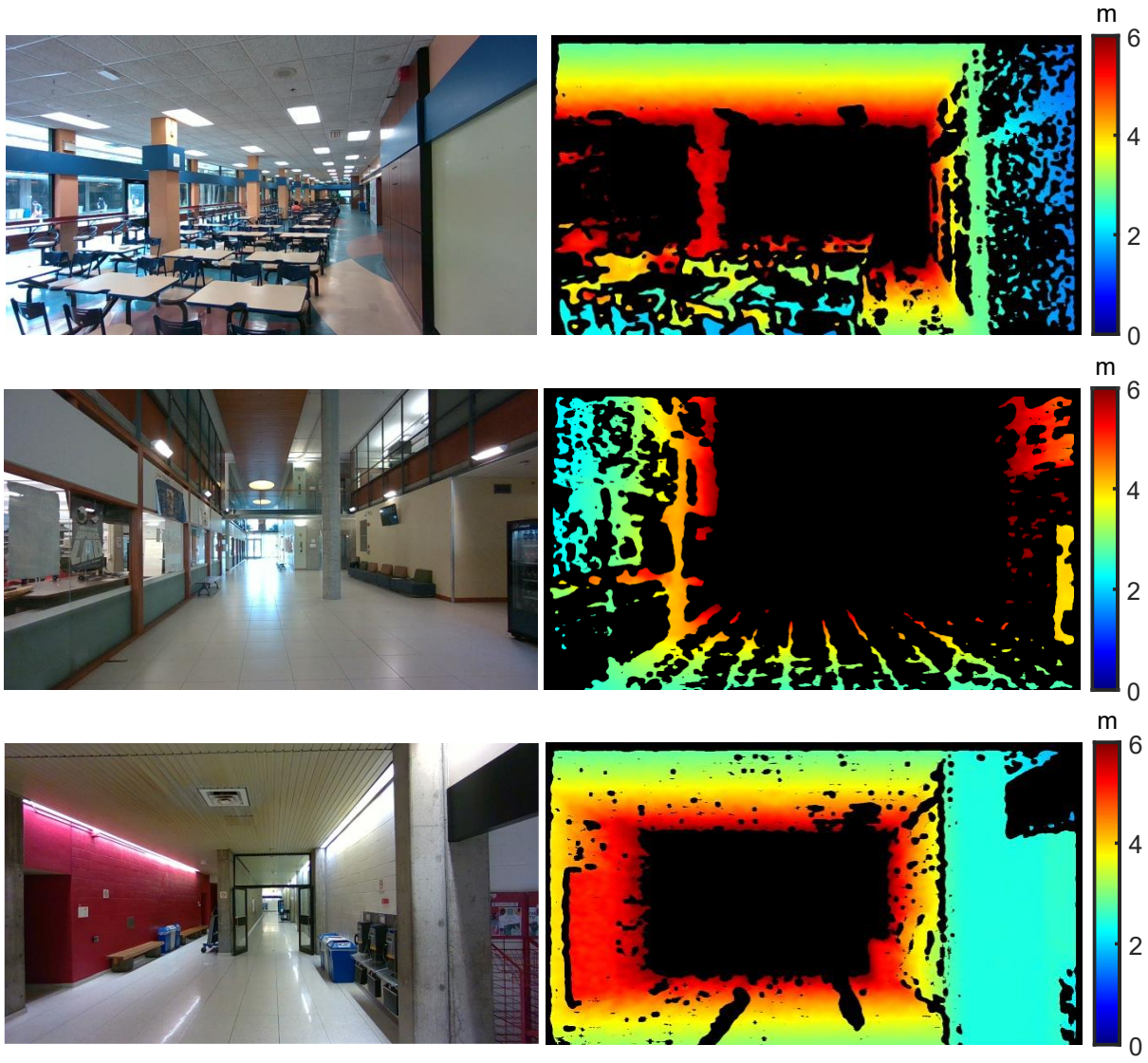


Figure 3.4: Images (color and depth) of Scene 3, 5, and 8 from the York-RGBD-LiDAR dataset (front camera).

The accuracy of the depth measurements of both the RealSense camera and LiDAR was validated through experimentation. For each camera, color, depth and LiDAR point clouds of a checkerboard (serving as a reference object) were captured over a range of distances between 0.5 and 35 m and the ground truth depth was measured using a BOSCH GLM 50 C spot laser with quoted measuring accuracy ±1.5 mm and range of up to 50 m [51]. The ground truth depth was measured from the center of the RealSense camera sensor to the center of the checkerboard. The measurements were taken in an indoor environment free from high intensity infrared sources and

there were no specularities on the target checkerboard. Multiple images and point cloud pairs were recorded at each distance. To obtain a distance measure that does not rely on the absolute orthogonality of the sensor setup, for each set of image and point cloud data we took the depth values at the pixel locations within a patch at the center of the checkerboard on the depth map, converted them to 3D world coordinates using camera intrinsics, calculated the 3D Euclidean distance from the camera to the checkerboard, and then found the average distance to the patch over all points in the patch (Figure 3.5). Then we detected the 3D coordinates of points within that patch on the checkerboard plane in the LiDAR point cloud. We then subtracted the vertical and horizontal offset of the LiDAR from the camera to translate the points to the camera center and calculated the average Euclidean distance for all the points. Average distances over repeated measurements for the same distance were computed and then compared with the ground truth spot laser measurements. The accuracy of the spot laser was verified by comparing measurements from 0.5 to 30m against a tape measure and the deviation between readings was found to be insignificant. Figure 3.6 shows the error in distance measurements as a fraction of ground truth for each camera.

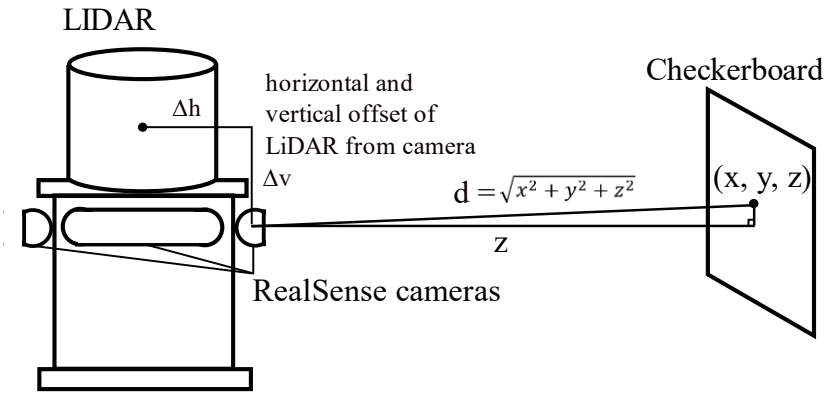
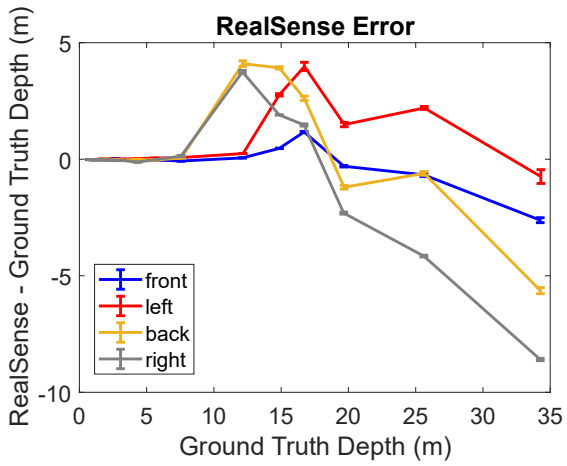
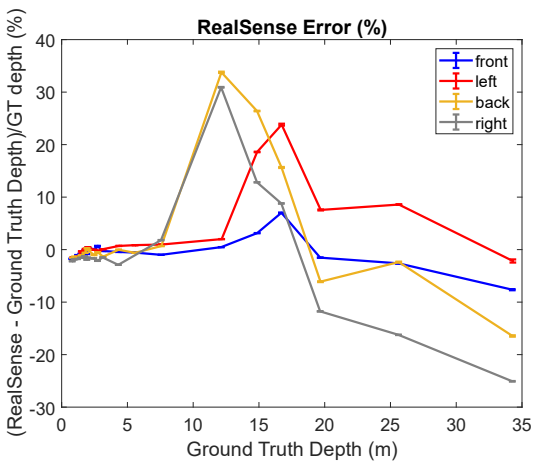


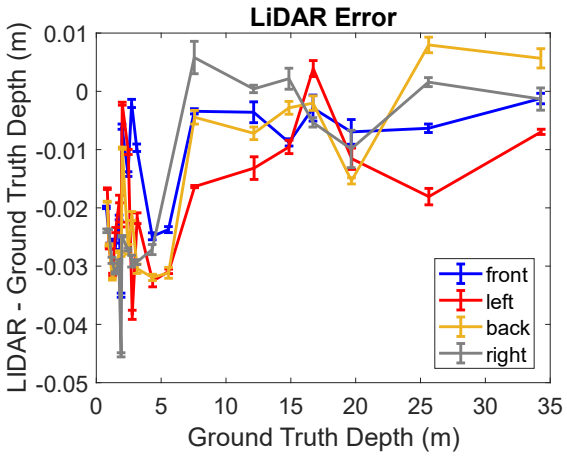
Figure 3.5: Euclidean distance d was measured from the center of the camera (0,0,0) to the center of the checkerboard (x,y,z) and compared to ground truth measurements obtained with a spot laser [51].



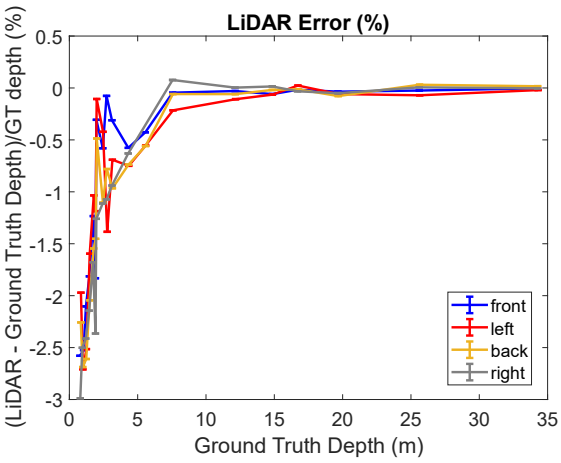
(a)



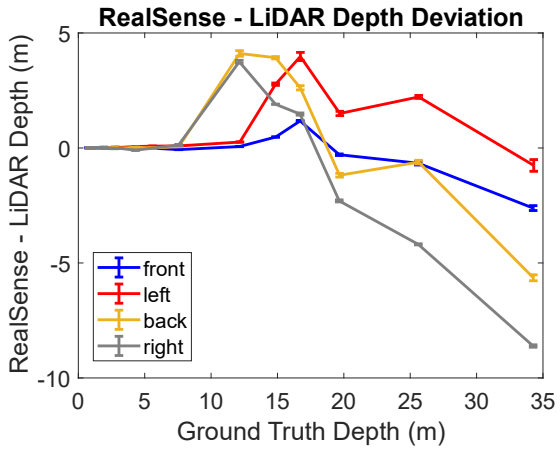
(b)



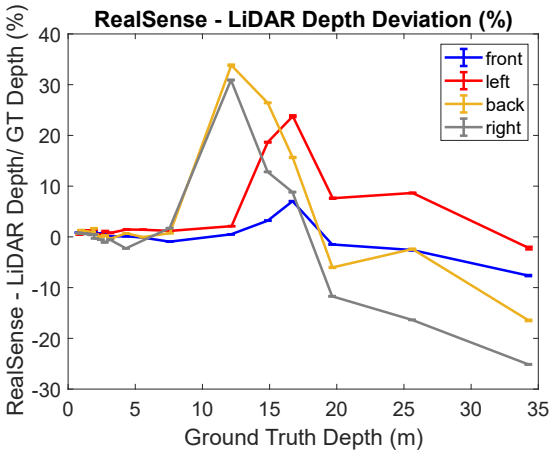
(c)



(d)



(e)



(f)

Figure 3.6: Error in RealSense and LiDAR sensors. The error bars show standard error over five measurements.

The plots in Figure 3.6 show that there is systematic bias in both the RealSense cameras and LiDAR sensor. For the RealSense cameras the error increases with distance; there is large deviation from ground truth beyond 6 m and this confirms that the cameras are not reliable beyond this range. Also the error is not consistent across the cameras. The LiDAR data is more accurate; the errors do not exceed 5 cm and are higher at closer distances (below 2m).

To prevent the systematic errors from affecting the evaluation of the proposed method, we correct the RealSense depth returns with respect to the LiDAR using a precomputed lookup table using the experimental depth measurements. This lookup table maps RealSense depth to LiDAR depth by linear interpolation, eliminating the intermediate step of converting both RealSense and LiDAR depths to spot laser ground truth.

The intrinsic parameters provided by the manufacturer were used for each RealSense camera. The extrinsic calibration between the LiDAR sensor and each RealSense camera was estimated using the MATLAB R2022a Lidar Camera Calibrator application [43]. 15 to 20 pairs of color images and LiDAR point clouds were collected by positioning a 6×7 checkerboard with square size 12.7 cm (checkerboard size: 88.9×76.2 cm) at various positions within the field of view of the RealSense camera. The checkerboard was held 1-1.3 m from the LiDAR-camera setup to get ideal measurements from both sensors. This was determined to be the ideal distance after repeating the data collection and calibration process with the checkerboard held at different distances away from the robot. The collected image-point cloud pairs were input into MATLAB's calibrator application which first detects the corners of the checkerboard in the image and converts the checkerboard corners to 3D world coordinates using camera parameters and checkerboard square size. Next it detects the checkerboard plane in the LiDAR point cloud using the RANSAC algorithm, then estimates the rigid transformation between the LiDAR and camera using constraints formed from 3D line and plane correspondences [44].

Figure 3.7 shows a screenshot of the calibrator app. To improve automatic detection of the checkerboard plane in the point cloud, settings on the app e.g. cluster threshold and dimension tolerance were fine-tuned, and the region in the LiDAR point cloud containing the checkerboard was manually selected using the 'Select Checkerboard' option (Figure 3.8).

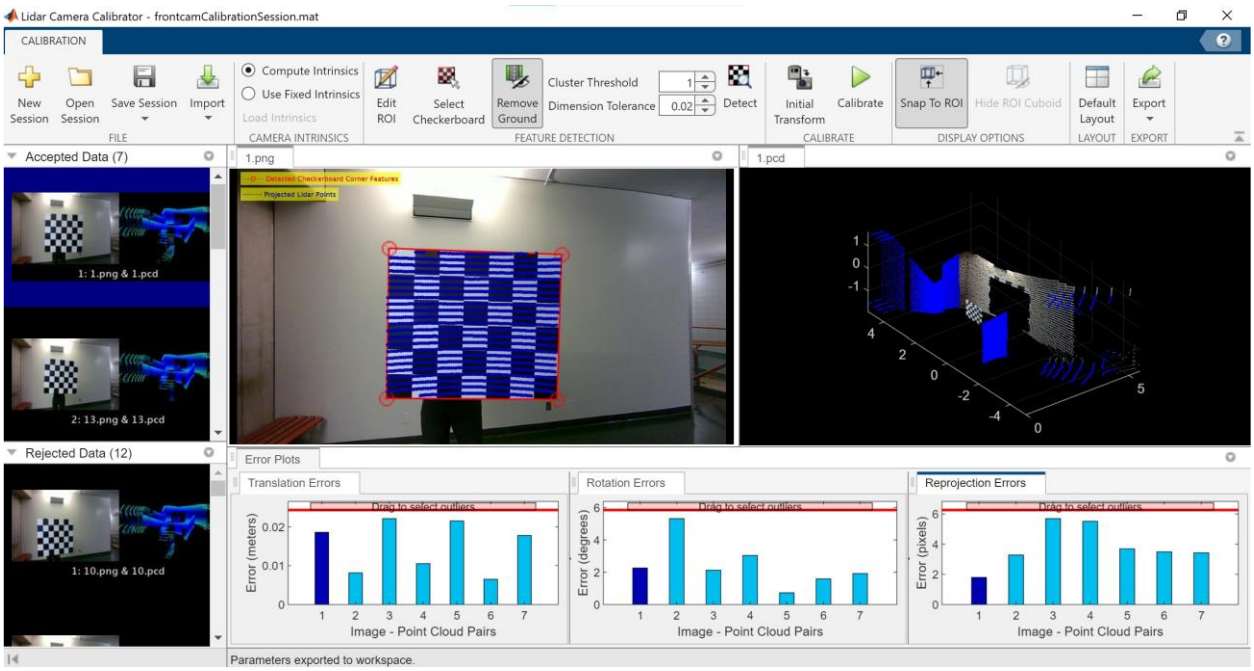


Figure 3.7: Screenshot of MATLAB's LiDAR Camera Calibrator app. Image-point cloud pairs in which the checkerboard is successfully detected are listed in 'Accepted Data'. The image in the center shows LiDAR checkerboard points projected to the 2D image. Plots at the bottom show errors for each accepted image-point cloud pair.

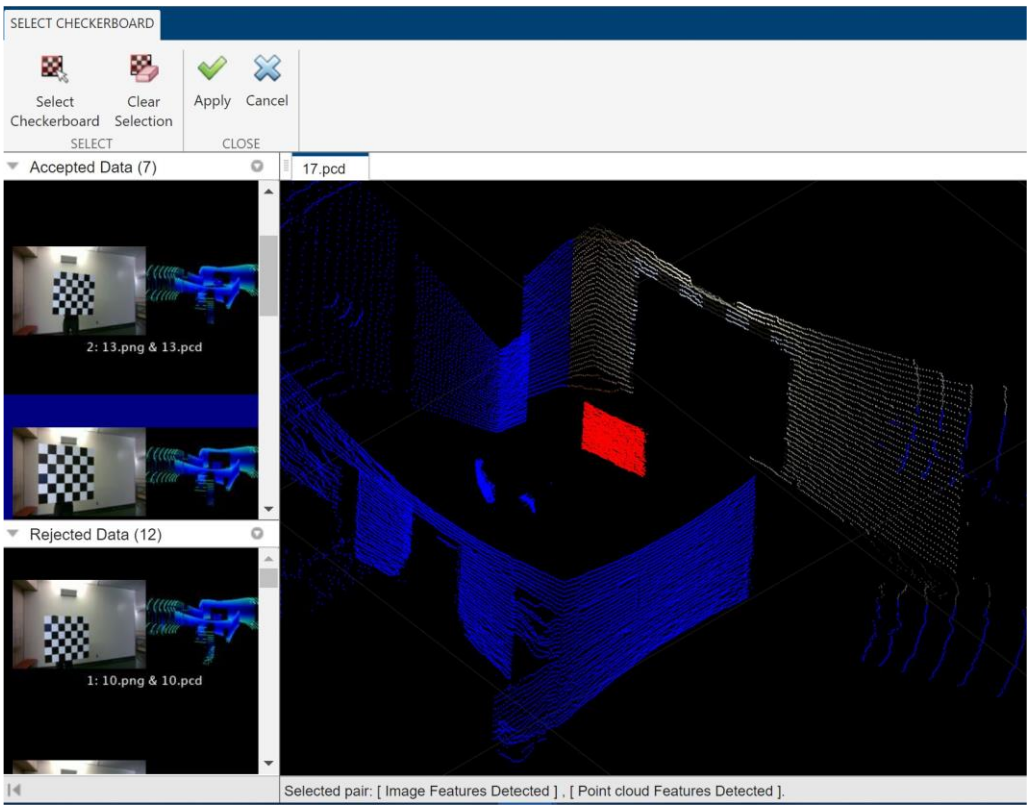


Figure 3.8: 'Select Checkerboard' option of MATLAB calibrator. The point cloud can be rotated and the user can select a square region of points by clicking and dragging the mouse. Here the red points have been selected.

Mean errors were calculated across the image-point cloud pairs used in the calibration (plots at bottom of Figure 3.7), where the errors are defined as:

- Translation error: the difference, in meters, between the centroid of the checkerboard plane in the LiDAR point cloud and the centroid of the plane formed from the 3D checkerboard corners from the image.
- Rotation error: the difference, in degrees, between the normal of the checkerboard plane in the LiDAR point cloud and the plane formed from the 3D checkerboard corners from the image.
- Reprojection error: the distance, in pixels, between the projected centroid of the checkerboard plane in the LiDAR point cloud and the checkerboard centroid in the 2D image. The 3D centroid of the checkerboard is found in the image (from corner points converted to 3D world points) and the 3D centroid of the LiDAR checkerboard plane is

found, both centroids are reprojected to the image and the distance between the two is calculated.

We obtained the best extrinsic calibration for the front camera with 0.015m mean translation error, 2.42° mean rotation error and 3.84 pixel mean reprojection error across 7 accepted pairs of images and corresponding point clouds. Errors for all cameras are reported in Table 3.1. From Figure 3.6 we see that the LiDAR has a systematic offset of around 3 cm (less than ground truth). This together with miscalculation between the LiDAR and RealSense camera could explain the high extrinsic error.

Table 3.1: Mean errors across image-point cloud pairs used in the calibration for each camera. Average errors were computed across 7 pairs for the front camera, 9 pairs for the right camera and 7 pairs for the left camera.

Camera	Translation Error (m)	Rotation Error (°)	Reprojection Error (pixels)
Front	0.015	2.42	3.84
Right	0.013	1.48	4.35
Left	0.023	2.36	9.75

Despite our best efforts, the extrinsic calibration errors using the MATLAB calibrator are relatively high. As future work, we will refine our calibration process to reduce errors. Considering the RealSense camera, we will review the camera API ROS nodelet to check for parameter errors that might affect experimental results, and verify through experiments the depth start point of the camera. We used intrinsic parameters provided by the manufacturer here, instead we can calibrate the camera in-house and obtain new intrinsics. We could also repeat the sensor accuracy tests and more carefully align the spot laser with the depth camera depth start point when taking measurements.

We also plan to try alternative methods to improve extrinsic camera calibration. Here we utilized only color images from the camera for calibration. We can make use of the camera depth data to refine the calibration by gradient descent. The extrinsic matrix provided by the MATLAB calibrator can be used as an initial estimate for the 6 DOF rigid transformation between the camera and LiDAR. We can use this initial matrix to transform the part of the LiDAR point cloud within the FOV of the camera into the camera frame. Then we can project the LiDAR

points to the image plane using intrinsic parameters. The RealSense depth estimates would then be bilinearly interpolated to obtain a camera depth estimate at each projected LiDAR point. We would define the sum of the squared residuals between the LiDAR depth estimates and RealSense depth estimates as our objective function, and minimize this objective function by gradient descent. Finally we can calculate the gradient of the squared residual with respect to the 6 DOF transformation, and update the transformation iteratively until the objective function converges to obtain a refined extrinsic matrix.

4 Linear Perspective Algorithm

The proposed method to extend depth has four steps. The algorithm takes an RGB image and a depth map as input, identifies line segments in the image, estimates the 3D orientation of these segments, and scales each 3D segment using the depth data.

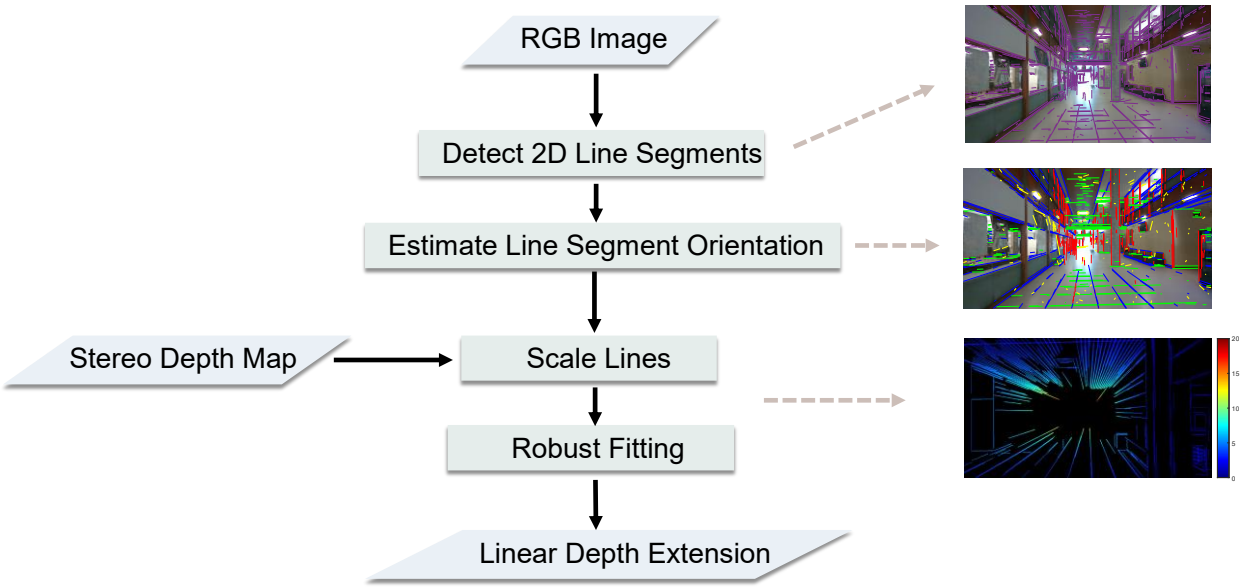


Figure 4.1: Flowchart of Linear Perspective Algorithm with visual output of intermediate stages.

4.1 Step 1: 2D Line Segment Detection

The method can use any algorithm that detects line segments in a color image. The method was tested using the MCMLSD [45] and LSD [46] line detectors. The MCMLSD algorithm detects line segments in two stages. In the first stage, a global probabilistic Hough approach is used to detect global lines. In the second stage each line is analyzed in the image domain to find the visible segments within each line [45]. The LSD line detector groups image regions with significant gradient magnitude based on gradient orientation in linear time; pixels are selected using a region growing process and each region is validated using an *a-contrario* approach [46]. MCMLSD is more accurate, with an average precision (AP) of 21.98% on the York Urban Dataset while LSD has an AP of 17.33%. LSD provides benefits in terms of runtime; [47] found LSD had a runtime of 36.51±1.60 ms to process a 640 x 480 image on Intel Core i7 compared to MCMLSD's runtime of 4.68±1.78 s.

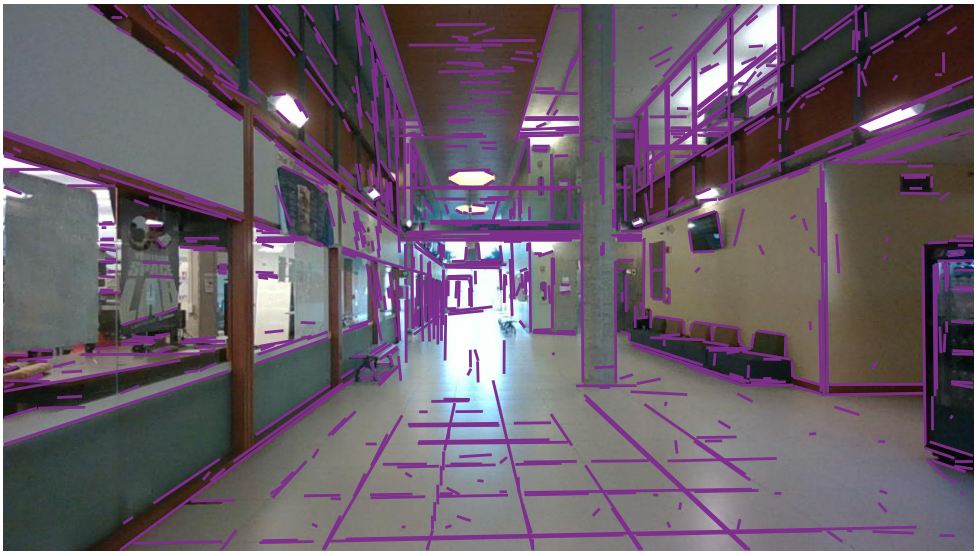


Figure 4.2: Image from York-RGBD-LiDAR dataset overlayed with line segments (purple) detected with MCMLSD algorithm [45].

4.2 Step 2: Estimation of 3D Line Segment Orientation

The Manhattan constraint [48] assumes that the scene contains three dominant and mutually orthogonal sets of parallel line segments [1]. This is particularly applicable to indoor scenes and the built environment. In step 2 each 2D line segment from step 1 is associated with one of three Manhattan directions and a background non-Manhattan process, computed using the $f\mathbf{R}$ algorithm [49] (Figure 4.2).

$f\mathbf{R}$ is a probabilistic Manhattan method for online camera rotation and focal length estimation. It assumes a camera-centered world frame and central principal point and estimates both focal length f and camera rotation relative to the Manhattan structure of the scene. Parallel lines in 3D converge on vanishing points in 2D and these vanishing points provide the Manhattan orientations. When the camera intrinsics and rotation are known, the i^{th} vanishing point $\mathbf{x_i}$ can be computed from $\mathbf{x_i} = \mathbf{KR_i}$ where $\mathbf{K}$ is the intrinsic matrix and $\mathbf{R_i}$ is the i^{th} column of rotation matrix $\mathbf{R}$. The $f\mathbf{R}$ algorithm uses line segments as features in a mixture model that generates the likelihood of each line under the parameter set consisting of f and $\mathbf{R}$. The optimal solution is found by maximizing the sum of log likelihoods weighted by line segment length. The final parameter estimates are obtained through a grid search followed by a nonlinear iterative search. The resulting parameter set is used to determine the vanishing point directions [49]. The

principal point and focal length are known in our York-RGBD-LiDAR dataset and to achieve better results we input these parameters into the $f\mathbf{R}$ algorithm and estimate only the rotation matrix.

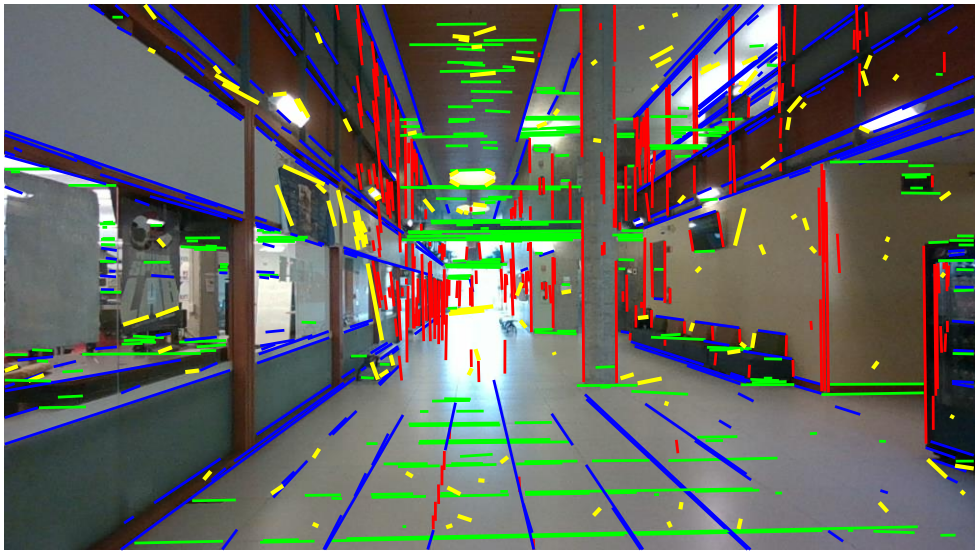


Figure 4.3: Image from York-RGBD-LiDAR dataset overlayed with line segments detected with MCMLSD algorithm [45] and labelled according to Manhattan orientation with the $f\boldsymbol{R}$ algorithm [49]. Red lines are in the vertical Manhattan direction, blue/green lines are in the horizontal Manhattan directions and yellow lines are non-Manhattan.

4.3 Step 3: Scaling Lines using Depth Data

We know the 3D orientation of the line segments following Step 2, but we do not know their true scale or depth. We make use of sparse depth estimates from the depth map returned by the stereo camera to calculate the scaling parameter of the lines.

Due to noise, the 2D line segments do not perfectly align with their associated Manhattan directions, so they are rectified to lie in the exact direction. Assuming a camera-centered coordinate system, a rectified line is represented as $(1-\alpha)\lambda X_1 + \alpha\lambda X_2$ where $0 \leq \alpha \leq 1$, $X_1, X_2 \in \mathcal{R}^3$ are the nominally-scaled 3D endpoints of the segment and λ is the unknown scaling factor. For each 2D line segment, we collect depth estimates from the D channel of the RealSense camera within $\sqrt{2}$ pixels of the line segment; this is the maximum distance between a pixel and its 8 neighboring pixels. We convert the pixel depth estimates to 3D world coordinates

by multiplying by the inverse camera intrinsic matrix, represented as green points in Figure 4.4. Next, we compute the scaling parameter for the line.

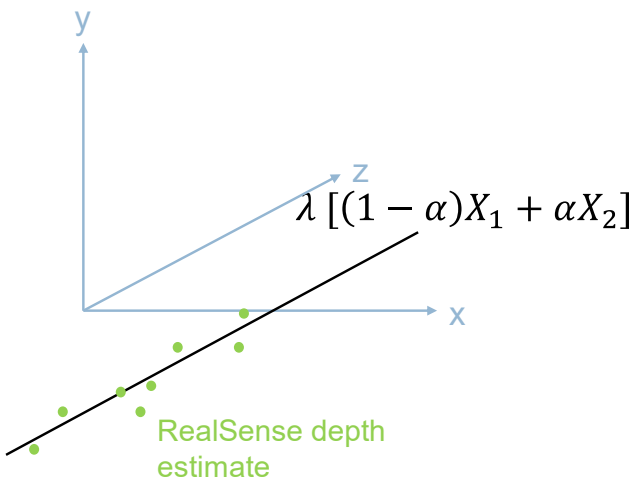


Figure 4.4: Scaling a 3D line segment using the mean distance of camera depth estimates (green points) from the line.

4.3.1 Scaling Each Line Segment

Our objective is to find the scaling factor λ that minimizes the total squared deviation of 3D depth estimates from the scaled 3D line segment (Figure 4.5). The equation for the scaling parameter λ can be derived from the equation for the distance d_i of a 3D point X_i from a 3D line (equation 1) (Figure 4.6) [50]:

$$d_i = \frac{|(X_i - \lambda X_1) \times (X_i - \lambda X_2)|}{\lambda |X_2 - X_1|} \propto |(\mathrm{X_2} - \mathrm{X_1}) \times \mathrm{X_i} + \lambda \mathrm{X_1} \times \mathrm{X_2}| \tag{1}$$

Taking a partial derivative with respect to λ and setting to 0 gives:

$$\sum_i \frac{\partial \mathrm{d_i^2}}{\partial \lambda} \propto (\mathrm{X_1} \times \mathrm{X_2}) \cdot \sum_{\mathrm{i}} \big((\mathrm{X_2} - \mathrm{X_1}) \times \mathrm{X_i} + \lambda \mathrm{X_1} \times \mathrm{X_2}\big) = 0 \tag{2}$$

Solving for λ:

$$\lambda = \frac{(\mathrm{X_1} \times \mathrm{X_2}) \cdot \big((\mathrm{X_1} - \mathrm{X_2}) \times \overline{\mathrm{X}}\big)}{|\mathrm{X_1} \times \mathrm{X_2}|^2} \tag{3}$$

Where $\overline{\mathrm{X}} = \frac{1}{\mathrm{N}}\sum_i X_i$ is the 3D centroid of the associated depth estimates.

Equation (3) can be further simplified by substituting:

$$\mathrm{X}_2 = \mathrm{X}_1 + \beta V \tag{4}$$

for $\beta \in \mathcal{R}$, where V is the vanishing point direction.

$$\lambda = \frac{(\mathrm{X}_1 \times \mathrm{V}) \cdot (\overline{\mathrm{X}} \times \mathrm{V})}{|\mathrm{X}_1 \times \mathrm{V}|^2} \tag{5}$$

To solve for λ, we let $\mathrm{X}_1 = \mathrm{X}_1^{\mathrm{c}} = [\mathrm{x}_1, \mathrm{y}_1, f]^{\mathrm{T}}$, the 3D coordinate of the image pixel corresponding to X_1 in the camera-centered coordinate frame.

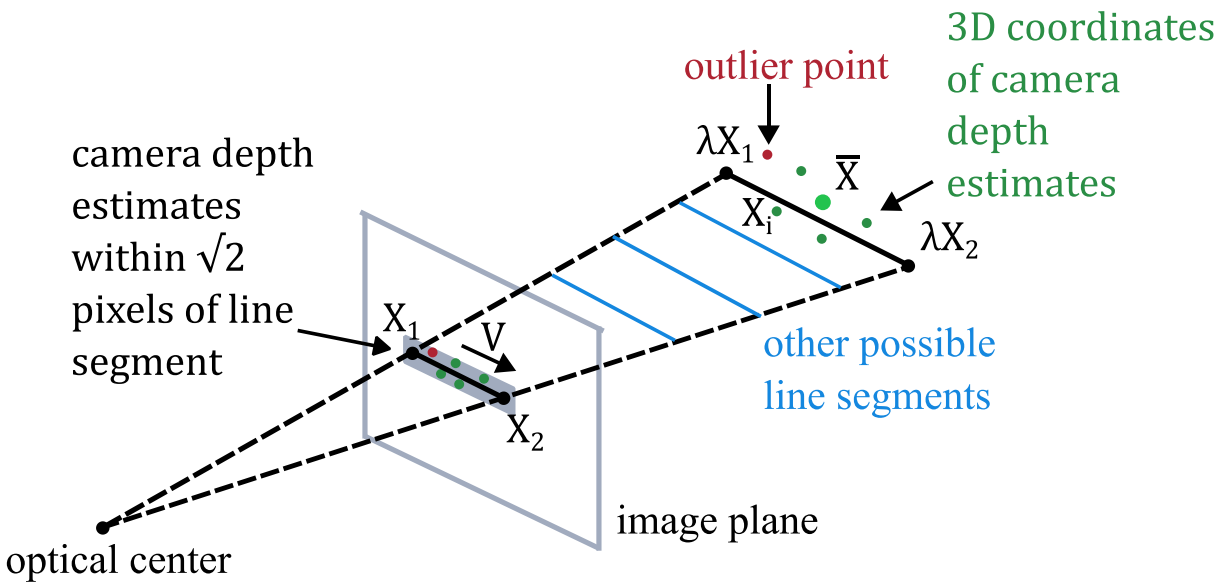


Figure 4.5: X_1 is the 3D coordinate of the image pixel corresponding to the first 2D endpoint and X_2 is the projection of the second scaled endpoint on the image plane. The black line connecting $X_1$and X_2 is the detected 2D line segment. Scaling with scaling parameter λ gives the line segment's true position in 3D space (black line connecting $\lambda X_1$and λX_2). If the scaling parameter is unknown, there are several possible 3D line segments, indicated by the blue lines. The green and red points in the shaded gray region represent camera depth estimates within $\sqrt{2}$ pixels of the

line segment and the red and green points near the scaled line represent these points projected to world coordinates. $\bar{X}$ is the centroid of these depth estimates (excluding the outlier).

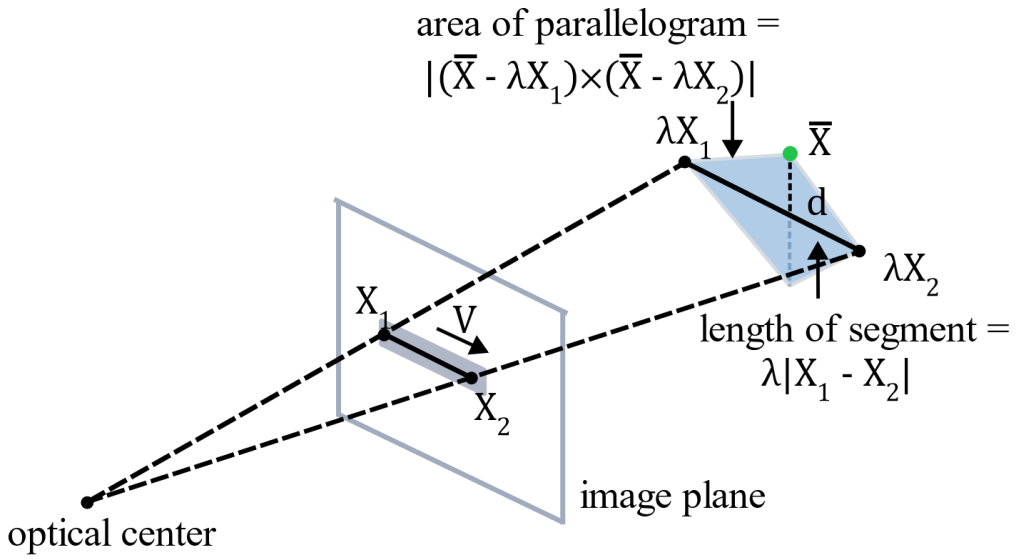


Figure 4.6: Distance d of centroid $\bar{X}$ from the 3D line segment $\lambda(X_2 - X_1)$ is given by equation (1) where the numerator is the area of the shaded parallelogram and the denominator is the length of the 3D segment $\lambda(X_2 - X_1)$, which forms the diagonal of the parallelogram.

To determine X_2 we consider the 3D line $\boldsymbol{v_1} = \lambda X_1 + \beta V$ and the line passing through X_2 and the optical center $\boldsymbol{v_2} = \gamma K^{-1}\bar{x}_2$. Here $\bar{x}_2$ is the image coordinate of the second detected endpoint in the 2D image in augmented form. The 2D line segment connecting X_1 and $\bar{x}_2$ is aligned with the Manhattan direction V. We solve for β by finding where these lines intersect (Figure 4.7). Given two lines $v_1 = p_1 + t_1 d_1$ and $v_2 = p_2 + t_2 d_2$, the lines intersect when $t_1 = \frac{(p_2 - p_1)\cdot n_2}{d_1 \cdot n_2}$ and $t_2 = \frac{(p_1 - p_2)\cdot n_1}{d_2 \cdot n_1}$, where $n_1 = d_1 \times n$, $n_2 = d_2 \times n$, and $n = d_1 \times d_2$ [50].

In our case we have $n_1 = V \times n$, $n_2 = (K^{-1}\bar{x}_2) \times n$, and $n = V \times (K^{-1}\bar{x}_2)$, giving

$$\beta = -\frac{\lambda X_1 \cdot n_2}{V \cdot n_2} \tag{6}$$

$$\gamma = \frac{X_1 \cdot n_1}{(K^{-1}\bar{x_2}) \cdot n_1}$$

We solve for β using equation (6) and then find X_2 using equation (4).

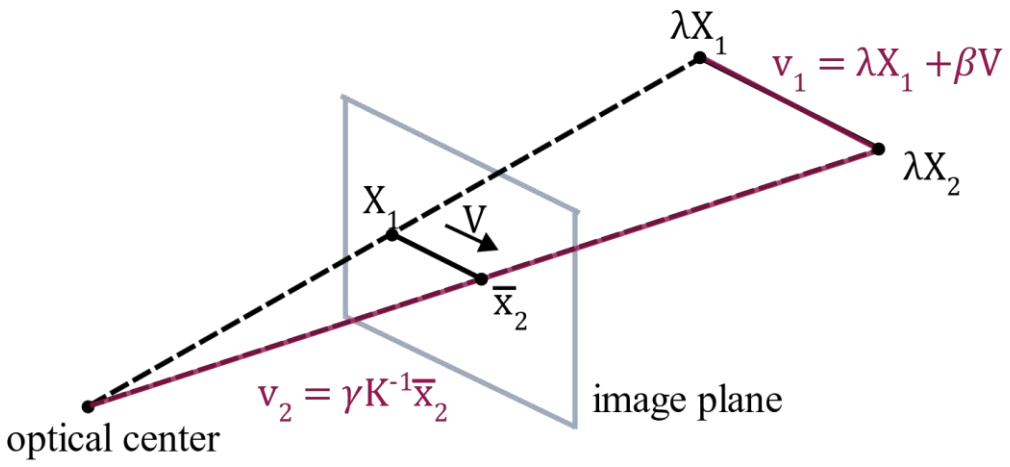


Figure 4.7: $\bar{x}_2$ is the image coordinate of the second detected endpoint. The detected 2D line segment has been rectified to be perfectly aligned to the Manhattan direction. β is found by finding where lines $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ intersect.

4.4 Step 4: Robust Fitting

In general, not all of the depth points within $\sqrt{2}$ pixels of a detected line segment (the maximum distance between a pixel and its 8 neighboring pixels) will lie on the surface generating that segment. To obtain a better estimate of the scaling parameter, we employ a robust fitting method. For each line segment, without loss of generality, we assume a 3D coordinate system with the z-axis aligned with the segment. The 3D points associated with the line are represented as deviation vectors (X_i, Y_i) from the line segment. This distribution of points can be modeled as independent and identically distributed zero-mean Gaussian with variance σ^2. The normalized squared distance $Z_i = \left(\frac{D_i}{\sigma}\right)^2$ of the points from the line then follows a $\chi^2(2)$ distribution [54]. We prune points that exceed the 95$^{\text{th}}$ percentile distance, i.e. points for which $D_i^2 > \sigma^2 F^{-1}(0.95,2)$, where $F^{-1}(p,k)$ is the inverse CDF of the $\chi^2(k)$ distribution and σ is estimated from $\sigma^2 \approx$

$\frac{1}{2k}\sum_{i=1}^{k} D_i^2$, are considered outliers and are removed. A new scaling parameter is computed with the pruned set of points. This process is repeated until no more outliers remain.

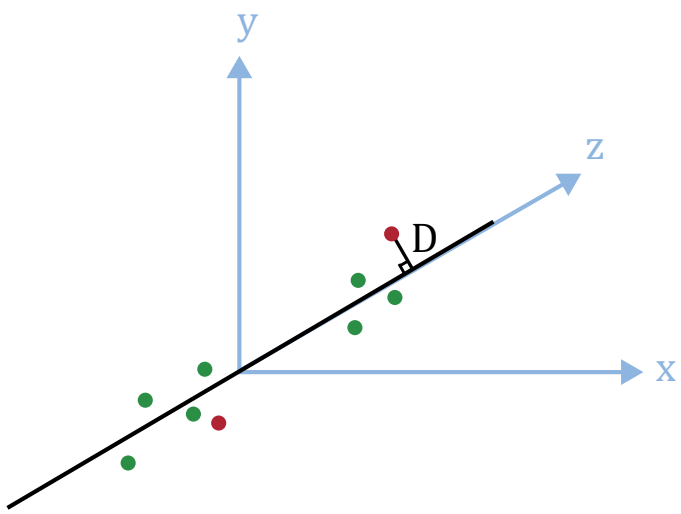


Figure 4.8: Red points represent outlier RealSense points for which distance D of the point from the line segment exceeds the 95th percentile distance. Green points are used for scaling.

Robust fitting removes noise and lowers the RMSE between the range estimates from the depth camera being used to scale the line and the scaled line produced by the depth scaling algorithm. The histograms in Figure 4.9 provide an example of how our algorithm removes outliers to produce a more normal distribution.

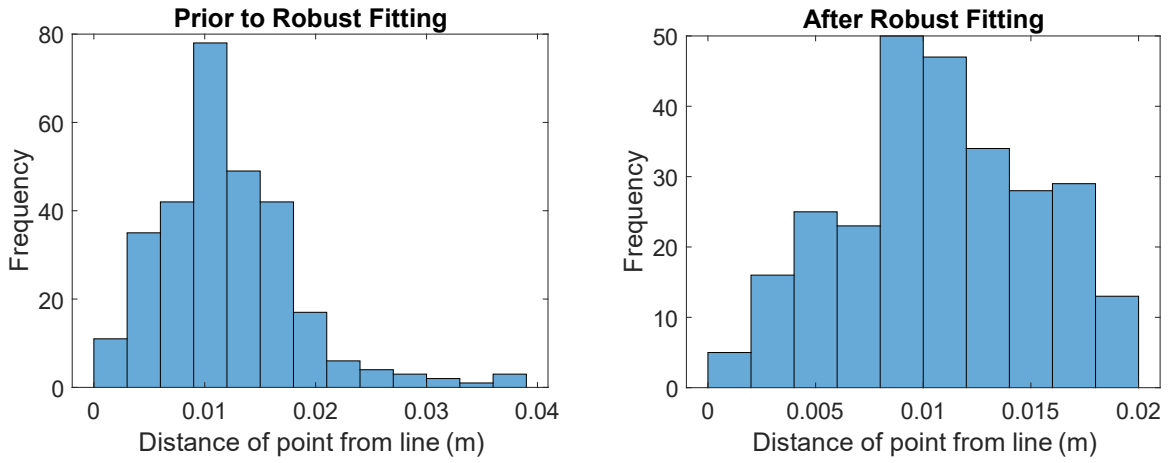


Figure 4.9: Distribution of deviation of points from a detected line segment before and after robust fitting. The RMSE before and after robust fitting is 1.35 cm and 1.17 cm respectively.

Figure 4.10 shows qualitative results after running the algorithm on a single image. Here the scaled 3D lines have been reprojected to the image and overlayed on the original input depth map. In this example, given camera depth estimates up to 6m, the method successfully extrapolates up to 20 m for certain line segments. The extent of extrapolation is most obvious in the central dark region of the bottom right depth map where originally there is no camera depth information.

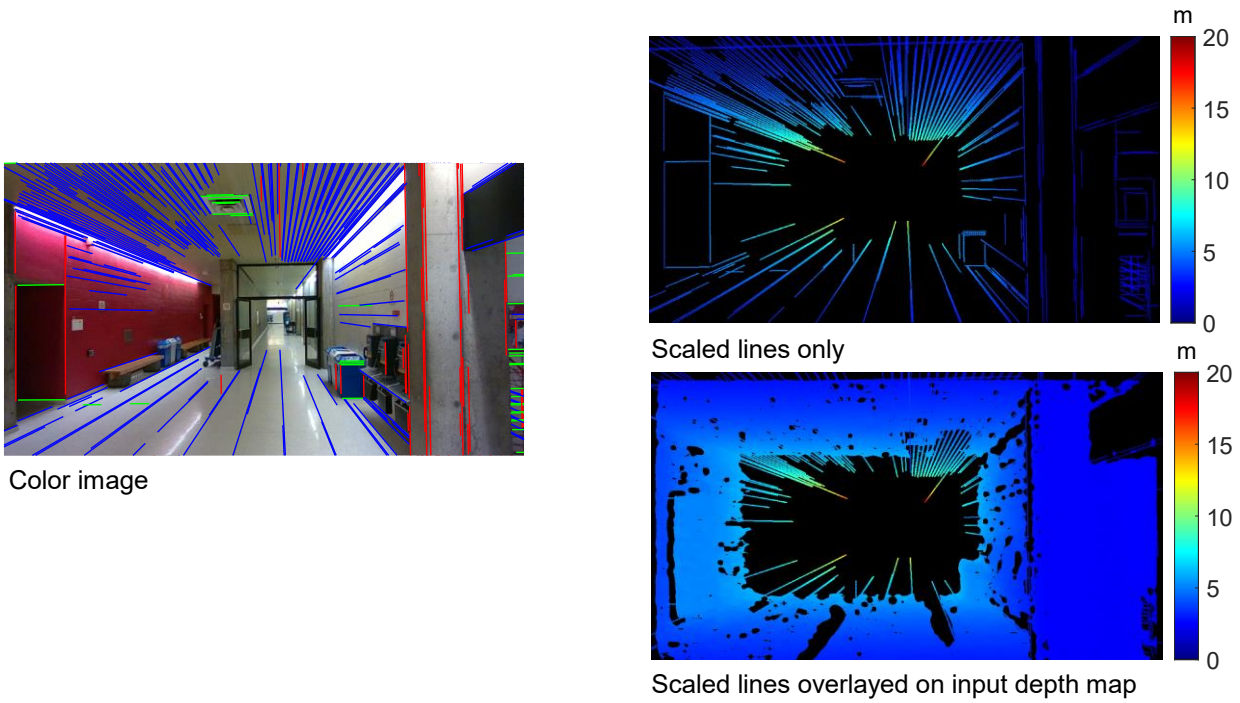


Figure 4.10: Example of linear depth completion in one image. The top right image shows scaled 3D lines reprojected to the image and the bottom right image shows the reprojected scaled lines overlayed on the original input depth map.

5 Experiments and Evaluation

5.1 Evaluation Measures

Experiments were designed to test the extent of interpolation and extrapolation produced by the proposed depth scaling algorithm, as well as the accuracy of depth estimated along the lines. Evaluation metrics used in the literature to assess depth completion algorithms include root mean square error (RMSE), root mean square log error (RMSLE), and absolute and square relative errors. We also introduce two new metrics to measure the extent of interpolation and extrapolation along the scaled line segment: percentage 2D interpolation and percentage 2D extrapolation.

To define these metrics, we split each line segment into depth-supported, interpolated and extrapolated intervals (Figure 5.1). For each line segment, we consider the RealSense points (pixel points with depth values) after robust fitting within $\sqrt{2}$ pixels of the segment. First to define the depth-supported intervals, we project the RealSense points onto the line. Without loss of generality we set one extreme point projection as the start point and the other as the end point, and sort the projections accordingly. Then we calculate the distance between consecutive projections along the line. Consecutive projections that are within $\sqrt{2}$ of each other are considered to belong to depth-supported intervals. Extrapolated intervals are sections at either end of the line segment between the scaled endpoint and the first RealSense point projection of the depth-supported interval closest to that endpoint. The remaining portions of the line, along which there are no point projections within a threshold of $\sqrt{2}$ pixels between consecutive projections, are interpolated intervals. We then define our two new metrics:

- Percentage 2D interpolation: the ratio of the sum of the lengths of all interpolated intervals to the length of the full line segment.
- Percentage 2D extrapolation: the ratio of the sum of the lengths of the extrapolated intervals to the length of the full line.

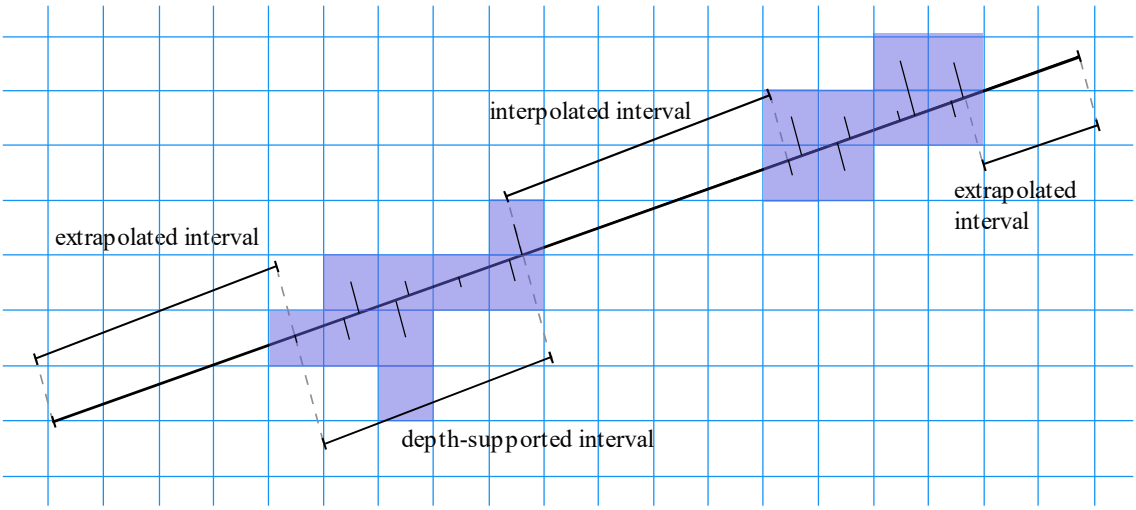


Figure 5.1: For pixels with depth returns (shaded blocks) within $\sqrt{2}$ pixels of the line segment, if the orthogonal projections of the pixels (thin black lines) are within $\sqrt{2}$ pixels of each other they fall within a depth-supported interval. Sections of the segment with no projections are interpolated intervals.

5.2 Percentage Extrapolation and Interpolation

The figures below show the results of our depth scaling algorithm for one image. Figure 5.2a) shows detected line segments. The lines are colored according to Manhattan orientation. Figure 5.2b) shows the depth map of the scene overlayed with the scaled lines. This figure highlights the extrapolation in the lines beyond the range of existing depth data (there are no depth returns for the black pixels in the depth map). Figure 5.2(c) zooms into lines in the image; the points circled in cyan are within $\sqrt{2}$ pixels of the line and are used to scale the line segment; depth points along the line falling within this threshold that are not circled were identified as outliers in the robust fitting step and are not used for scaling.

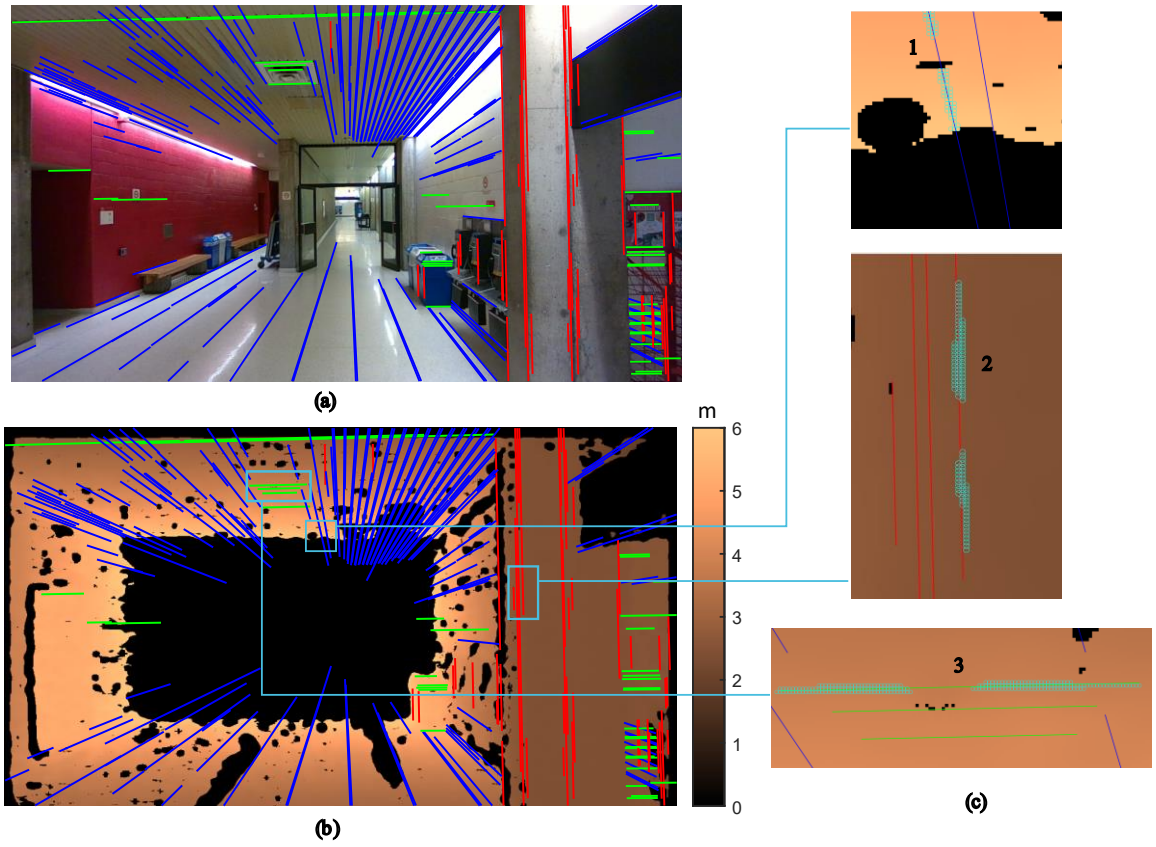


Figure 5.2: (a) Example of line segments detected and classified according to 3D orientation (red: vertical, blue/green: horizontal) by the fR algorithm [49]. (b) Depth map overlayed with scaled line segments. (c) Zoomed-in views of three lines. Depth points (excluding outlier points identified via robust fitting) used to scale each line are circled in cyan.

Figure 5.3 shows the interpolated and extrapolated intervals in a line segment of each Manhattan orientation, and Table 5.1 reports the amount of interpolation and extrapolation in each line.

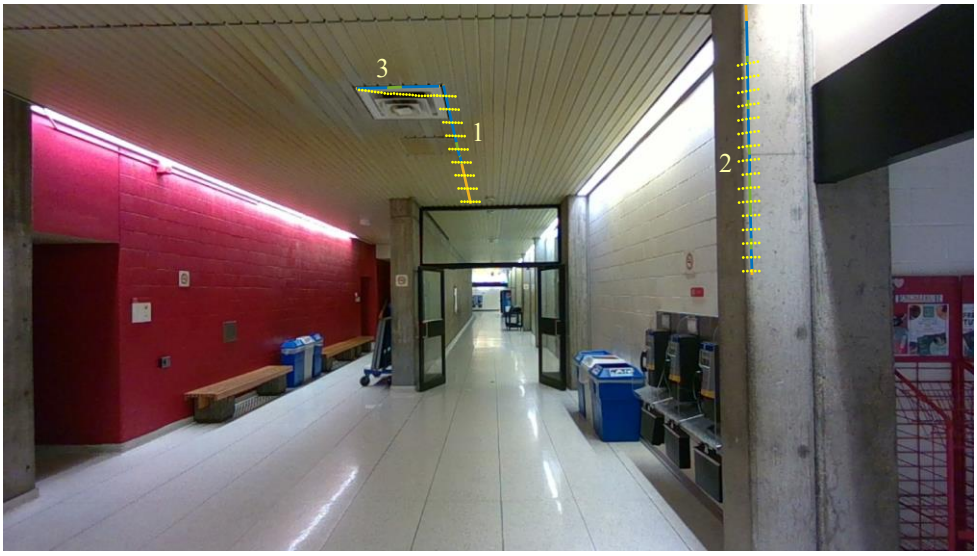


Figure 5.3: Interpolated (green), extrapolated (orange) and RealSense-depth supported (blue) intervals for three line segments. LiDAR points that fall within the region of interest (ROI) of each line segment are displayed as yellow points.

Table 5.1: % interpolation and extrapolation for lines in Figure 5.3 and RMSE between scaled line and associated RealSense depths before and after robust fitting.

Line	% interpolation	% extrapolation	RMSE before robust fitting (cm)	RMSE after robust fitting (cm)
1 (blue)	7.44	34.12	1.35	1.17
2 (red)	28.53	8.10	1.48	0.62
3 (green)	17.46	0.92	3.30	1.82

Figure 5.4 shows the distribution of interpolated and extrapolated intervals in all line segments over all images from all camera and Figure 5.5 presents bar charts of the mean percentage interpolation and extrapolation over all the images from each camera. The line segments on average are extrapolated more than they are interpolated.

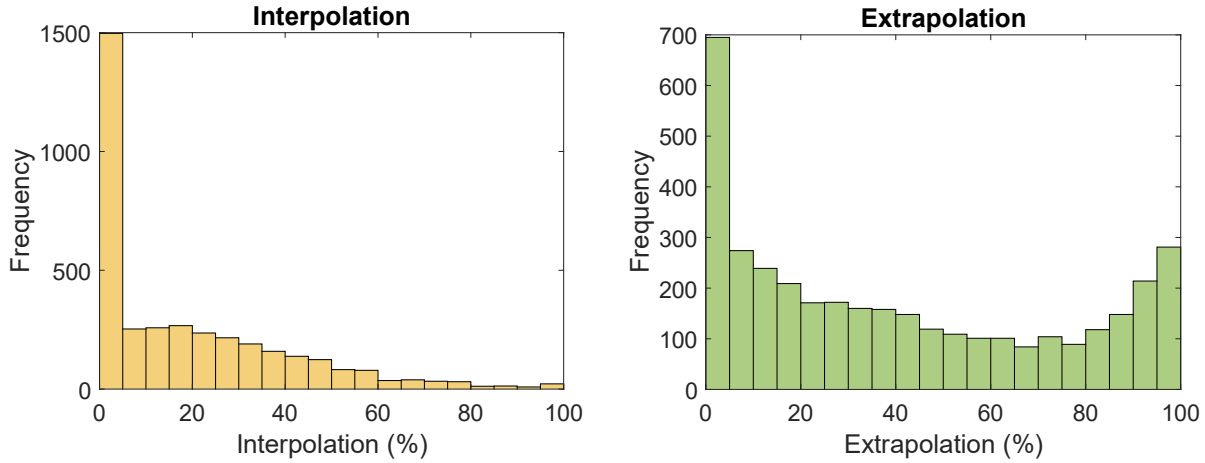


Figure 5.4: Histograms of percentage interpolation and extrapolation in all images from all three camera.

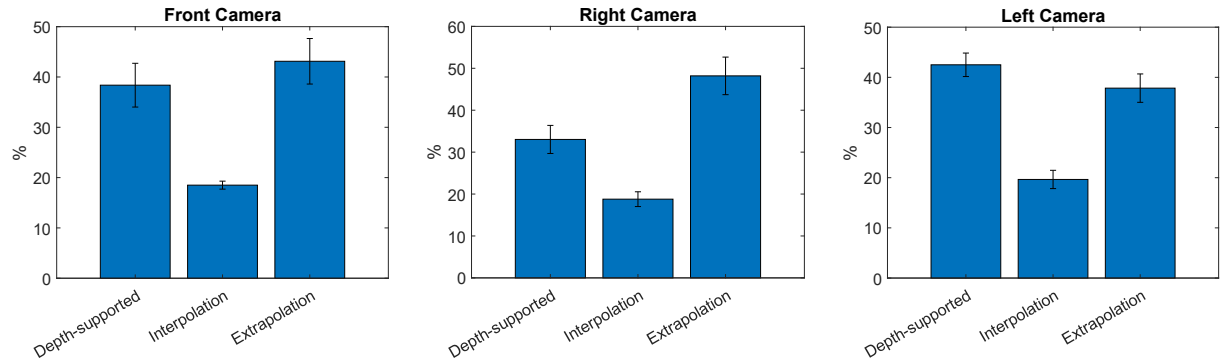


Figure 5.5: Mean percentage interpolation, extrapolation and percentage depth-supported intervals in lines over all images for each camera. The error bars represent standard error.

5.3 Accuracy Evaluation

We evaluate the accuracy of our depth completion algorithm on depth-supported, extrapolated and interpolated components of each line segment individually using LiDAR data. The LiDAR scans for each scene have a 360° field of view and a different coordinate frame than the RealSense camera, so the point clouds were preprocessed as follows to compare with the results of depth scaling:

1. The LiDAR point cloud was transformed to the depth camera coordinate frame using the estimated extrinsic parameters obtained from LiDAR-camera calibration. The 3x4

extrinsic matrix containing the 3x3 rotation and 3x1 translation matrices was multiplied with the LiDAR coordinates to transform them to camera coordinates.

2. The portion of the LiDAR point cloud within the horizontal and vertical field of view of the camera was selected.

The evaluation procedure has the following steps:

1. We determine the subpixel image of each LiDAR point using the intrinsic parameters of the RealSense camera.
2. We define a region of interest (ROI) around each line segment to identify ground truth LiDAR points to account for extrinsic calibration error. We note that the ROI should be flexible since there can be considerable offset between LiDAR and RealSense point estimates due to error in calibration (Figure 5.6).
3. We consider the LiDAR points within this ROI around the scaled line segment. To get depth predictions along the scaled line segment, we sample points equidistant from each other along the 3D scaled segment and project them to 2D. We then find the 2D point on the scaled line segment that is closest to each LiDAR point, calculate the difference in the depth and take the average.

Tables 5.2 and 5.3 show errors using a relatively large ROI (14.3 pixel threshold). Note that the larger mean/median extrapolation errors (Table 5.2) are due in part to the greater distances being estimated; this is normalized out in the percentage error evaluation (Table 5.3).

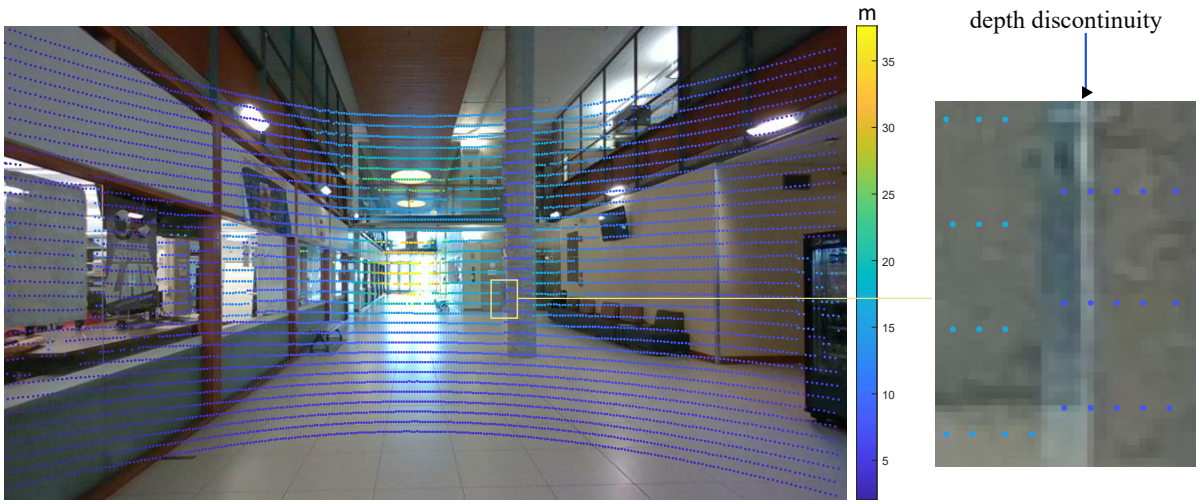


Figure 5.6: LiDAR points projected to 2D and plotted according to depth value over a front camera image. Zoomed-in portion highlights reprojection error between lidar points and RGB points.

We performed paired t-tests over scenes to determine whether the difference in mean error between interpolated/extrapolated intervals and depth-supported intervals is statistically significant (Table 5.4). From the results we see a statistically significant p-value in error for extrapolation. For interpolated intervals, the p-value is greater than 0.05, suggesting that the difference in errors between the interpolated intervals and depth-supported intervals is statistically insignificant and likely due to random sample error.

Table 5.2: Average mean absolute error (MAE) and median error (ME) with standard error in interpolated, extrapolated and RealSense depth-supported intervals of all lines over all scenes for each camera. Here depth predictions are evaluated against LiDAR points within a threshold of 14.3 pixels from the line segment.

Camera	Intervals with Depth		Interpolated Intervals		Extrapolated Intervals	
	MAE (cm)	Median Error (cm)	MAE (cm)	Median Error (cm)	MAE (cm)	Median Error (cm)
Front	31.09±4.31	15.10±1.19	31.26±5.38	14.96±1.41	48.62±8.26	22.30±2.88
Right	33.82±4.20	20.83±2.98	33.93±4.88	21.34±3.63	50.93±6.67	29.09±4.60
Left	32.53±6.16	7.48±2.00	44.30±9.92	8.16±1.83	62.16±13.85	15.37±2.23

Table 5.3: Average percent mean absolute error and median percent error with standard error in interpolated, extrapolated and RealSense depth-supported intervals of all lines over all scenes for each camera. Here depth predictions are evaluated against LiDAR points within a threshold of 14.3 pixels from the line segment.

Camera	Intervals with Depth		Interpolated Intervals		Extrapolated Intervals	
	MAE (%)	Median Error (%)	MAE (%)	Median Error (%)	MAE (%)	Median Error (%)
Front	6.16±0.50	3.70±0.19	5.77±0.44	3.81±0.31	7.99±0.88	4.52±0.44
Right	6.70±0.54	4.63±0.51	6.91±0.61	4.94±0.58	8.44±0.72	5.66±0.60
Left	6.18±1.47	2.17±0.37	7.70±1.97	2.60±0.60	10.53±1.86	4.20±0.73

Table 5.4: Average percent mean absolute error for all lines in dataset and results of paired t-test comparing MAE in interpolated and depth-supported intervals and extrapolated and depth-supported intervals.

MAE (%)			Paired t-test results (interpolated)			Paired t-test results (extrapolated)		
Depth-supported	Interpolated	Extrapolated	p-value	t-value	DOF	p-value	t-value	DOF
6.32±0.47	6.63±0.60	8.83±0.68	0.17	-1.42	24	3.56e-05	-5.06	24

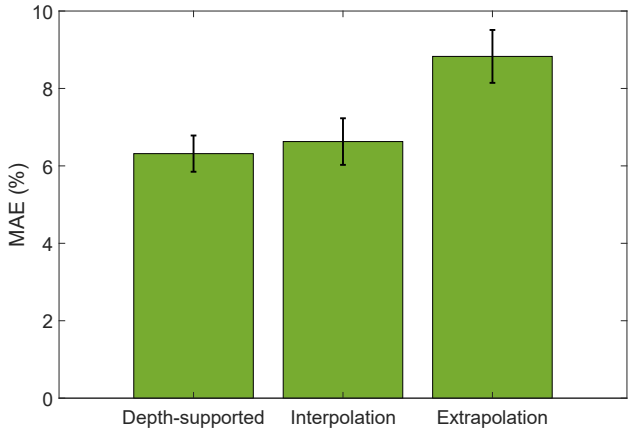
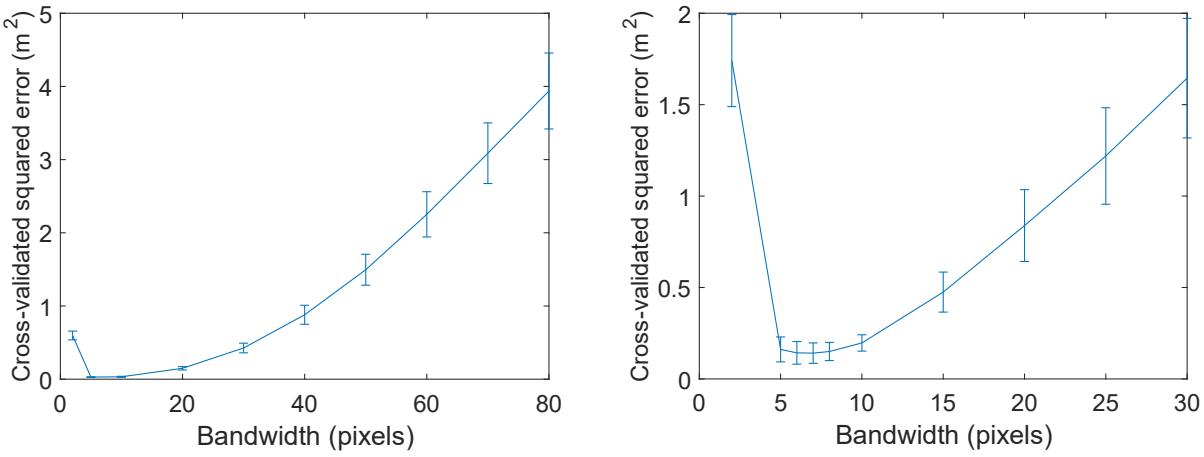


Figure 5.7: Bar chart depicting average percentage mean errors over all lines in the dataset images.

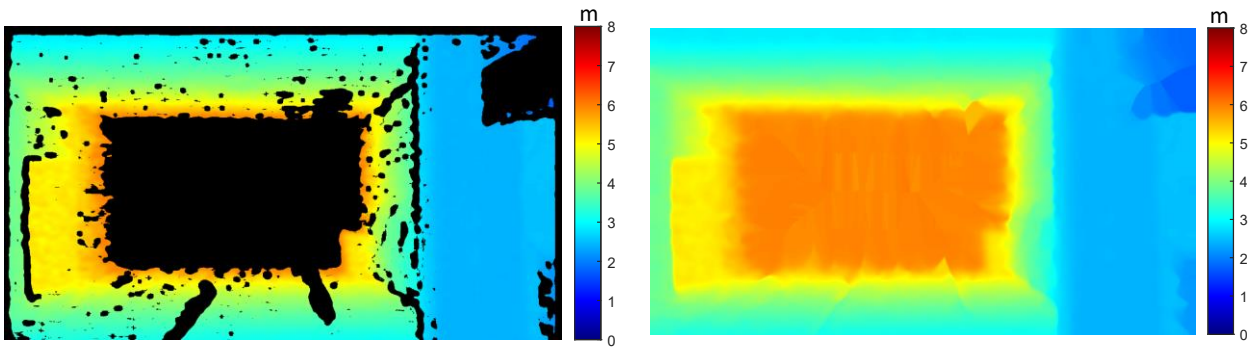
5.4 Comparison with Baseline Method

We compare our method to a baseline method that can be used to fill in depth maps. The method uses Gaussian kernel regression to predict the depth at points on the image by taking a weighted average of surrounding points. It is implemented in MATLAB by convolving the input depth map with a Gaussian kernel in the x and y directions. To determine the optimal bandwidth for the kernel we performed leave-one-out cross-validation over a range of bandwidths. We calculated the sum of squared deviations between predicted depth and the original RealSense depth at test points. Figure 5.8 shows a plot of the average results across all images. Based on these results we select 6 pixels as the optimal bandwidth for kernel regression.



(a) (b)

Figure 5.8: Leave-one-out cross-validation results for selecting optimal bandwidth. (a) average sum of squared deviation in depth values in all images calculated over a large bandwidth range; error bars indicate standard error of the mean, (b) errors calculated over a narrower bandwidth range to find bandwidth generating lowest error.



(a) (b)

Figure 5.9: (a) Original depth map for the scene in Figure 5.2 and depth map predicted by (b) Gaussian kernel regression using a bandwidth of 6 pixels.

To compare the two methods we evaluated at all the LiDAR points falling within a ROI around the scaled interval for each interval type in the 2D image. We obtained depth predictions at the same points along the scaled intervals for both methods, then calculated the difference in LiDAR depth and predicted depth and the mean absolute error for the interval. We repeated for a range of ROIs (Figure 5.10). The percent errors are high due to extrinsic calibration error, but this extrinsic error is affecting both methods in the same way. The plots in Figure 5.10 show that the proposed method performs comparably to the baseline method in interpolated intervals and intervals with depth for all cameras, and much better in extrapolated intervals for the front and right cameras. This meets expectations as the baseline method is primarily an interpolation method. Given camera depth estimates ranging up to 6m, the linear perspective method is able to extrapolate depth up to 30 m (Figure 5.11).

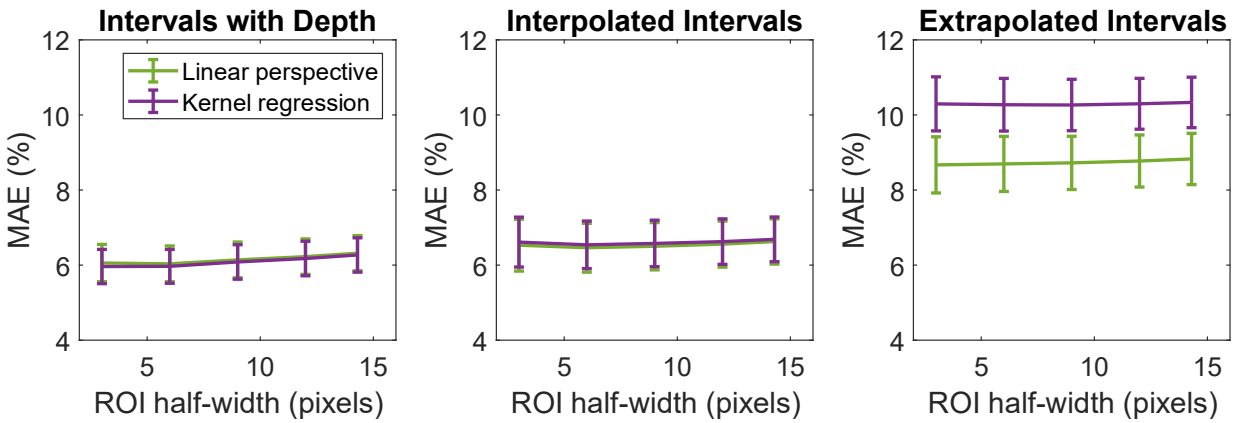


Figure 5.10: Average percent mean absolute error in depth predictions for each method over ROI ranging from 3 to 14.3 pixels for lines from all images for each interval type.

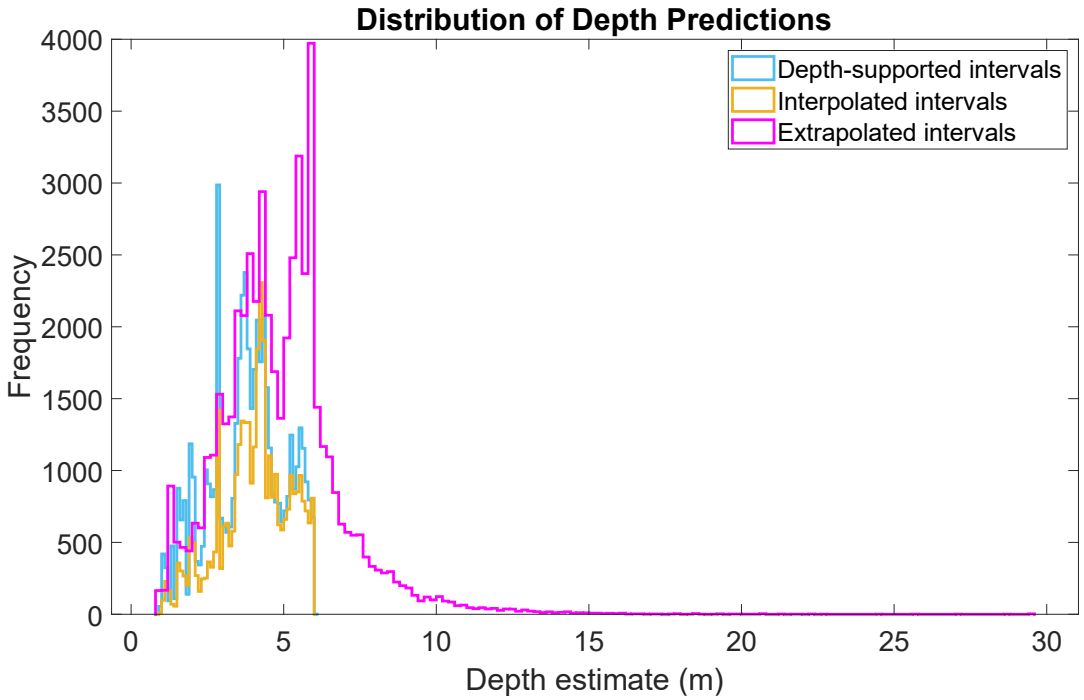


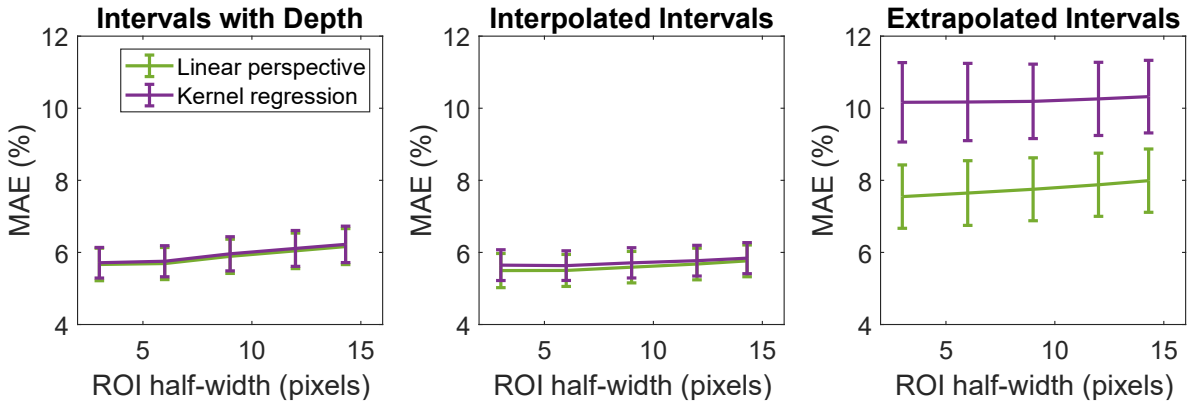
Figure 5.11: The distribution of predicted depths (generated by linear perspective method) at evaluation points along line segments. Depths are extrapolated up to 30 m.

Paired t-tests were performed to test if the difference in error in depth predictions estimated by the kernel regression method and by the linear perspective method is significant. Table 5.5 shows the resulting p-values from the test performed on mean errors over all images in the dataset for an ROI of 14.3 pixels. For extrapolated intervals, the difference in errors is statistically significant.

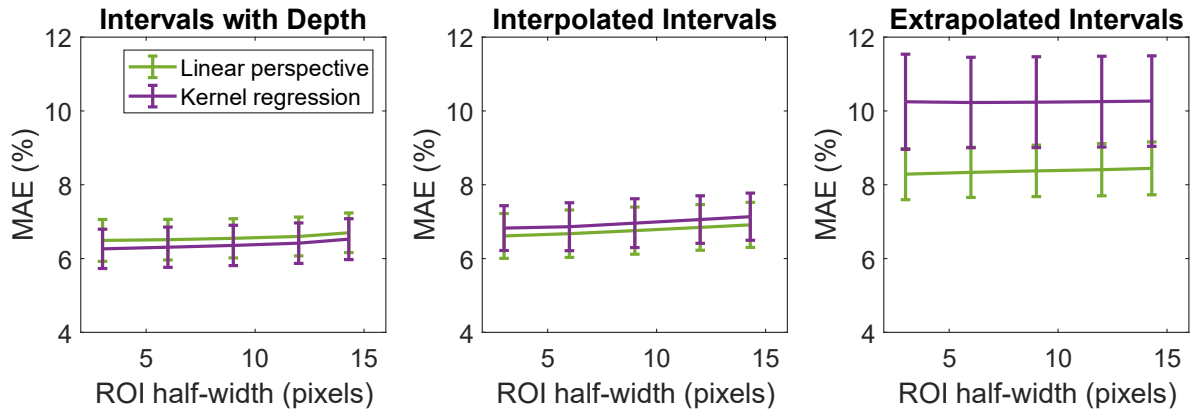
Table 5.5: Paired t-test results comparing MAE over all dataset images between baseline method and the proposed method for all interval types for an ROI of 14.3 pixels.

Depth-supported interval			Interpolated Intervals			Extrapolated intervals		
p-value	t-value	DOF	p-value	t-value	DOF	p-value	t-value	DOF
0.441	0.784	24	0.368	-0.917	24	2.85e-4	-4.243	24

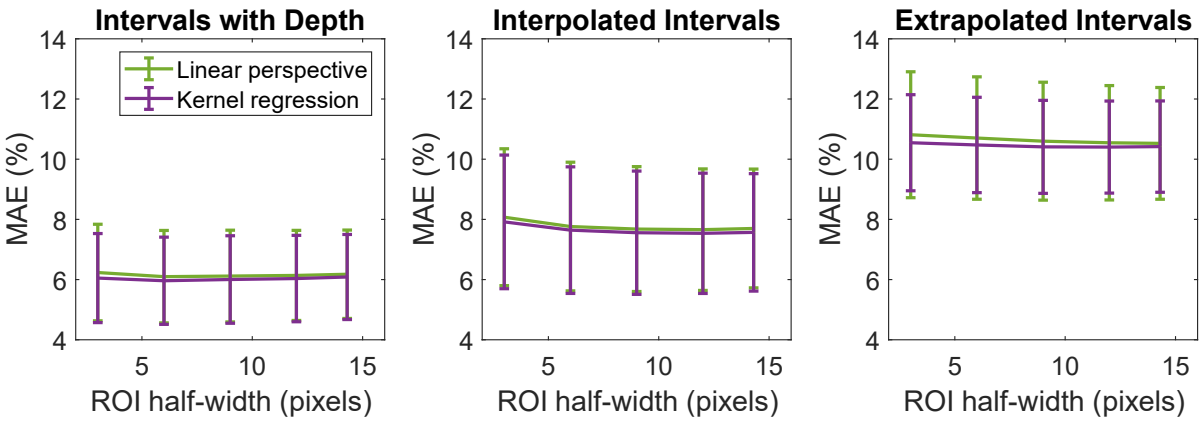
The difference in mean errors varied across cameras (Figure 5.12). The difference in extrapolation errors is higher for the images from the front camera compared to the left camera, and this could be because there was less extrapolation in the left camera images. Figure 5.13 shows the depth predictions in extrapolated intervals ranged up to 30m for the front camera, but only up to less than 15m for the left camera. The proposed method extrapolates better to longer ranges than the baseline method so the errors in longer-range depth predictions were lower compared to baseline predictions.



(a) Results for front camera



(b) Results for right camera



(c) Results for left camera

Figure 5.12: Average percent mean absolute error in depth predictions for each method over ROI ranging from 3 to 14.3 pixels for lines from all images for each interval type.

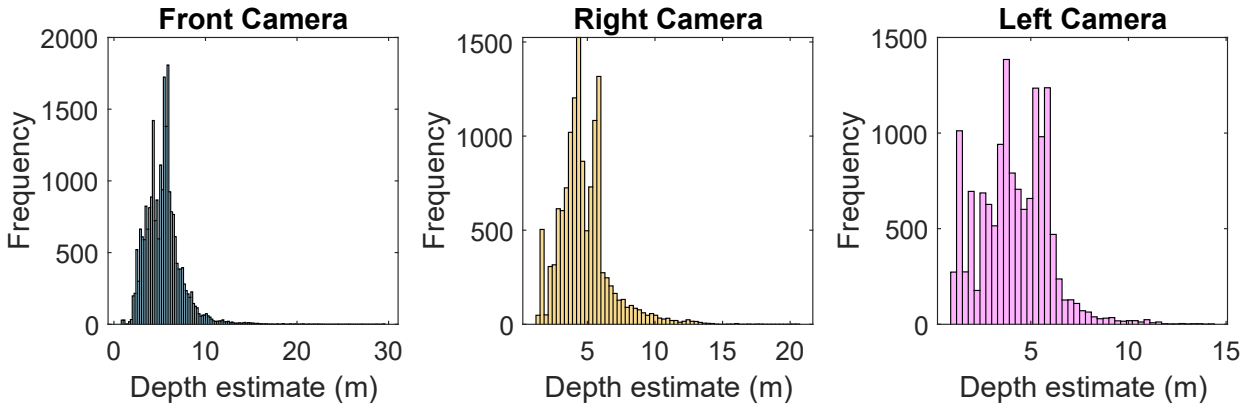


Figure 5.13: Distribution of depth predictions (generated by linear perspective method) at evaluation points along line segments in extrapolated intervals across images from each camera.

5.5 Results

With a distance threshold of 14.3 pixels (Table 5.3 and 5.4), percent error in depth predictions generated by the proposed method ranges between 6-11%. These errors may partly be attributed to points on the line segment being incorrectly matched with a LiDAR point on a different surface.

Limitations in the dataset contribute to some of the observed error as we know there is error in the extrinsic camera calibration. The higher average errors across the right and left camera images may be due to higher extrinsic calibration error for these cameras; from Table 3.1 the reprojection error for the front camera was 3.84 pixels whereas the error for the right and left cameras was 4.35 and 9.75 pixels respectively. Moreover, the LiDAR data is noisy and inaccurate in areas with shiny, transparent surfaces or where there is low reflectivity. The distribution of errors is positively skewed perhaps due to these effects, and so the median error may be more representative of performance than the mean. The median errors for all intervals and all cameras are lower than the mean errors.

Other errors arise from the shortcomings of the depth scaling algorithm. Some lines may be incorrectly detected and some may be associated with the wrong Manhattan orientation in step 2 of the algorithm; Figure 5.14 illustrates this. The line segment orientations are calculated by applying the Manhattan constraint, so the method works best for Manhattan scenes. The presence of non-Manhattan structures and clutter in the environment leads to higher errors. The RealSense

range estimates are also noisy and incorrect extrapolation occurs in cases where the robust fitting step fails to remove all outlier range points.

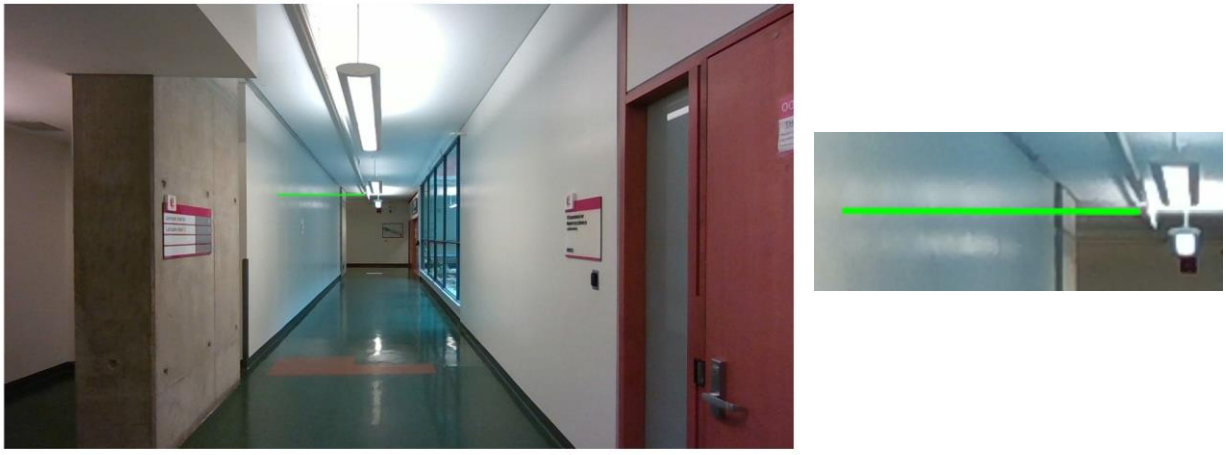


Figure 5.14: Example case where line has been incorrectly detected in the 2D image.

Comparison of the proposed method with the kernel regression baseline found lower extrapolation errors in the linear perspective method with respect to the baseline in sets of images where lines were extrapolated to greater distances (Figure 5.10 and 5.12).

6 Conclusions

Depth sensing is important for most robot applications. The quality of the raw data captured by depth sensors can be inadequate, and various algorithms have been proposed to improve data quality. Depth estimation and completion approaches introduced in the literature all attempt to get a more complete understanding of depth in an environment, and a reoccurring observation in existing work is that incorporating scene geometry leads to better depth recovery. A common approach in the literature is to fill in missing depth values and many existing approaches are fairly simple 2D interpolation (inpainting) methods. In this thesis I presented a method that exploits 3D geometric cues, specifically the monocular cue of linear perspective, to extend the depth range of a depth camera. The method does not require training and is not computationally expensive, unlike recent deep learning based methods.

To evaluate the depth scaling algorithm we created a new long-range dataset, the York-RGBD-LiDAR dataset. This dataset contains color and depth images of indoor scenes with ranges extending up to 90 m, with supporting LiDAR ground truth. We will make the dataset public so that it can be used by the research community to test and develop long-range monocular depth estimation and completion approaches.

From our evaluation experiments we found that our proposed method successfully interpolates and extrapolates detected lines in Manhattan scenes given sparse depth estimates, providing depth information in parts of the scene with no measured depth data from the sensor. Statistical evaluation found depth estimates in interpolated parts of a line to be of reasonable quality as those in parts supported by input depth, thus the algorithm can reliably estimate depth in intervals with and without support. The proposed approach performs slightly better than the kernel regression interpolation technique when predicting depth in depth-supported and interpolated intervals of lines and is much better at extrapolating depth than this method.

7 Future Work

There are several directions for future work:

1. Calibration errors affect our evaluation results. Here we mapped LiDAR points to camera coordinates to estimate the extrinsic matrix. A different method of LiDAR-camera calibration making use of the RealSense depth estimates can be tested to see if it produces more accurate results.
2. We can collect additional long-range scenes to expand the dataset.
3. Another research direction would be to test the effect of fixing the Manhattan orientations in our depth scaling approach and see whether this significantly improved performance. Testing this would involve creating a variation of the algorithm in which the orientations of the lines are not provided and both line orientation and scale are estimated from the input depth data.
4. Groups of line segments can be connected to produce Manhattan tree representations by applying the Gestalt principle of proximity. A set of line segments in a Manhattan tree can be scaled using a single scaling parameter. We then only require depth estimates for any one line segment in the Manhattan tree, enabling us to retrieve more complete depth information given sparse depth input. Preliminary work has been done to test this idea and extend the proposed method by implementing part of the LS3D [1] pipeline that generates Manhattan trees. Future work will involve evaluation of this extension to the method as well as further development for improved performance.
5. The proposed method is not directly comparable to existing depth completion algorithms which generally return complete depth maps. As future work the algorithm can be extended to generate completed depth maps from the scaled lines. Then depth estimation results can be compared against published results. The proposed approach can potentially be combined with an existing deep learning-based depth completion algorithm to test whether the added information on scene geometry provided by the depth scaling algorithm improves the overall performance.
6. Finally, the long-term objective of this research project is to implement the depth scaling algorithm on the Dingo robot platform and test if it enhances the robot's scene

understanding capabilities. The work done for this thesis can then be integrated with the work ongoing to develop the Dingo robot platform into a social robot.

Bibliography

1. Qian, Y., Ramalingam, S., & Elder, J. H. (2018, December). LS3D: Single-view Gestalt 3D surface reconstruction from Manhattan line segments. In *Asian Conference on Computer Vision* (pp. 399-416). Springer, Cham.
2. *Dingo Indoor Mobile Robot*. Clearpath Robotics. (2022, December 21). Retrieved January 9, 2023, from https://clearpathrobotics.com/dingo-indoor-mobile-robot/
3. Labbé, M., & Michaud, F. (2011, September). Memory management for real-time appearance-based loop closure detection. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1271-1276). IEEE.
4. Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
5. Mertan, A., Duff, D. J., & Unal, G. (2022). Single image depth estimation: An overview. *Digital Signal Processing*, 103441.
6. Kalloniatis, M., & Luu, C. (2005). The Perception of Depth. In H. Kolb (Eds.) et. al., *Webvision: The Organization of the Retina and Visual System*. University of Utah Health Sciences Center.
7. Hoiem, D., Efros, A. A., & Hebert, M. (2005, October). Geometric context from a single image. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* (Vol. 1, pp. 654-661). IEEE.
8. Saxena, A., Chung, S., & Ng, A. (2005). Learning depth from single monocular images. In *Proceedings of the 18th International Conference on Neural Information Processing Systems (NIPS'05)* (pp. 1161-1168).
9. Saxena, A., Sun, M., & Ng, A. Y. (2008). Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(5), 824-840.
10. Delage, E., Lee, H., & Ng, A. Y. (2006, June). A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (Vol. 2, pp. 2418-2428). IEEE.

11. Liu, B., Gould, S., & Koller, D. (2010, June). Single image depth estimation from predicted semantic labels. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 1253-1260). IEEE.
12. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., & Navab, N. (2016, October). Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth International Conference on 3D Vision (3DV)* (pp. 239-248). IEEE.
13. Alhashim, I., & Wonka, P. (2018). High quality monocular depth estimation via transfer learning. *arXiv preprint arXiv:1812.11941*.
14. Yin, W., Liu, Y., Shen, C., & Yan, Y. (2019). Enforcing geometric constraints of virtual normal for depth prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 5684-5693).
15. Khan, F., Salahuddin, S., & Javidnia, H. (2020). Deep learning-based monocular depth estimation methods—A state-of-the-art review. *Sensors*, *20*(8), 2272.
16. Kuznietsov, Y., Stuckler, J., & Leibe, B. (2017). Semi-supervised deep learning for monocular depth map prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6647-6655).
17. Bauer, Z., Li, Z., Orts-Escolano, S., Cazorla, M., Pollefeys, M., & Oswald, M. R. (2021, December). NVS-MonoDepth: Improving Monocular Depth Prediction with Novel View Synthesis. In *2021 International Conference on 3D Vision (3DV)* (pp. 848-858). IEEE.
18. Godard, C., Mac Aodha, O., & Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 270-279).
19. Godard, C., Mac Aodha, O., Firman, M., & Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 3828-3838).
20. Guizilini, V., Ambrus, R., Pillai, S., Raventos, A., & Gaidon, A. (2020). 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2485-2494).
21. Xie, Z., Yu, X., Gao, X., Li, K., & Shen, S. (2022). Recent Advances in Conventional and Deep Learning-Based Depth Completion: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*. 1-21.

22. Cho, J. M., Park, S. Y., & Chien, S. I. (2020). Hole-Filling of RealSense Depth Images Using a Color Edge Map. *IEEE Access*, *8*, 53901-53914.
23. Telea, A. (2004). An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, *9*(1), 23-34.
24. Tomasi, C., & Manduchi, R. (1998, January). Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)* (pp. 839-846). IEEE.
25. Chen, L., Lin, H., & Li, S. (2012, November). Depth image enhancement for Kinect using region growing and bilateral filter. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)* (pp. 3070-3073). IEEE.
26. Ku, J., Harakeh, A., & Waslander, S. L. (2018, May). In defense of classical image processing: Fast depth completion on the cpu. In *2018 15th Conference on Computer and Robot Vision (CRV)* (pp. 16-22). IEEE.
27. Criminisi, A., Perez, P., & Toyama, K. (2004). Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, *13*(9), 1200–1212.
28. Patel, J., & Sarode, T. K. (2014). Exemplar based image inpainting with reduced search region. *International Journal of Computer Applications*, *92*(12).
29. Zhao, Y., Bai, L., Zhang, Z., & Huang, X. (2021). A surface geometry model for LiDAR depth completion. *IEEE Robotics and Automation Letters*, *6*(3), 4457-4464.
30. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., & Geiger, A. (2017, October). Sparsity invariant cnns. In *2017 International Conference on 3D Vision (3DV)* (pp. 11-20). IEEE.
31. Zhang, Y., & Funkhouser, T. (2018). Deep depth completion of a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 175-185).
32. Li, A., Yuan, Z., Ling, Y., Chi, W., & Zhang, C. (2020). A multi-scale guided cascade hourglass network for depth completion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 32-40).

33. Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., ... & Zhang, Y. (2017). Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*.
34. Silberman, N., Hoiem, D., Kohli, P., & Fergus, R. (2012, October). Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision* (pp. 746-760). Springer, Berlin, Heidelberg.
35. Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., & Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5828-5839).
36. Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, *32*(11), 1231-1237.
37. Schops, T., Schonberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., & Geiger, A. (2017). A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3260-3269).
38. Baruch, G., Chen, Z., Dehghan, A., Dimry, T., Feigin, Y., Fu, P., ... & Shulman, E. (2021). ARKitScenes--A Diverse Real-World Dataset For 3D Indoor Scene Understanding Using Mobile RGB-D Data. *arXiv preprint arXiv:2111.08897*.
39. Li, P., Cai, K., Saputra, M. R. U., Dai, Z., & Lu, C. X. (2022). OdomBeyondVision: An Indoor Multi-modal Multi-platform Odometry Dataset Beyond the Visible Spectrum. *arXiv preprint arXiv:2206.01589*.
40. Ma, Y., Zhu, X., Zhang, S., Yang, R., Wang, W., & Manocha, D. (2019, July). Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 6120-6127).
41. *High-resolution OS1 LIDAR SENSOR: Robotics, trucking, mapping*. Ouster. (n.d.). Retrieved January 9, 2023, from https://ouster.com/products/scanning-lidar/os1-sensor/.
42. *Introducing the Intel® realsense™ depth camera D455*. Intel® RealSense™ Depth and Tracking Cameras. (2022, December 5). Retrieved January 9, 2023, from https://www.intelrealsense.com/depth-camera-d455/

43. *Lidar and camera calibration*. Lidar and Camera Calibration - MATLAB & Simulink. (n.d.). Retrieved January 9, 2023, from https://www.mathworks.com/help/lidar/ug/lidar-and-camera-calibration.html
44. Zhou, L., Li, Z., & Kaess, M. (2018, October). Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5562-5569). IEEE.
45. Almazan, E. J., Tal, R., Qian, Y., & Elder, J. H. (2017). Mcmlsd: A dynamic programming approach to line segment detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2031-2039).
46. Von Gioi, R. G., Jakubowicz, J., Morel, J. M., & Randall, G. (2008). LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(4), 722-732.
47. Suárez, I., Buenaposada, J. M., & Baumela, L. (2021). Elsed: Enhanced line segment drawing. *arXiv preprint arXiv:2108.03144*.
48. Coughlan, J. M., & Yuille, A. L. (2003). Manhattan world: Orientation and outlier detection by bayesian inference. *Neural Computation*, *15*(5), 1063-1088.
49. Qian, Y., & Elder, J. H. (2022). A Reliable Online Method for Joint Estimation of Focal Length and Camera Rotation. In *European Conference on Computer Vision* (pp. 249-265). Springer, Cham.
50. "Wolfram Mathworld: The web's most extensive Mathematics Resource," *Wolfram MathWorld: The Web's Most Extensive Mathematics Resource*. [Online]. Available: https://mathworld.wolfram.com/. [Accessed: 15-Sept-2022].
51. *165 ft. laser measure*. GLM 50 C | 165 Ft. Laser Measure | Bosch Power Tools. (n.d.). Retrieved March 2, 2023, from https://www.boschtools.com/ca/en/boschtools-ocs/laser-measures-glm-50-c-128522-p/
52. Behrens, R. R. (1999). Adelbert Ames, Fritz Heider, and the chair demonstration. *Gestalt Theory*, *21*(3), 184-190.
53. Wong, B. (2010). Points of view: Gestalt principles (Part 1). *Nature Methods*, *7*(11), 863.
54. *Chi-square distribution*. Encyclopedia. (n.d.). Retrieved March 6, 2023, from https://encyclopedia.pub/entry/34125

55. Coughlan, J. M., & Yuille, A. L. (1999, September). Manhattan world: Compass direction from a single image by bayesian inference. In *Proceedings of the Seventh IEEE International Conference on Computer Vision* (Vol. 2, pp. 941-947). IEEE.
56. Intel. (2022, November). *Intel® RealSense™ Product Family D400 Series Revision 014* [Fact sheet]. Retrieved March 28, 2023, from https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet