ASSESSMENT OF CAMERA POSE ESTIMATION USING GEO-LOCATED IMAGES FROM SIMULTANEOUS LOCALIZATION AND MAPPING

THESIS

David W. Beargie, Capt, USAF
AFIT/ENG/19M

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/ENG/19M

ASSESSMENT OF CAMERA POSE ESTIMATION USING GEO-LOCATED IMAGES FROM SIMULTANEOUS LOCALIZATION AND MAPPING

THESIS

Presented to the Faculty

Department of Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

David W. Beargie, B.E.E.E.

Capt, USAF

February 26, 2019

AFIT/ENG/19M

ASSESSMENT OF CAMERA POSE ESTIMATION USING GEO-LOCATED IMAGES FROM SIMULTANEOUS LOCALIZATION AND MAPPING

THESIS

David W. Beargie, B.E.E.E.
Capt, USAF

Committee Membership:

Capt Aaron J. Canciani, PhD
Chair

Robert C. Leishman, PhD
Member

John F. Raquet, PhD
Member

Abstract

There is a need for a robust and easy to use indoor truthing system based on camera pose estimation. This research proposes a method for camera pose estimation using a one-time use of Simultaneous Localization and Mapping (SLAM) with high Size, Weight, Power and Cost (SWAP-C) sensors in order to enable future robust localization of low SWAP-C cameras.

Determining position and orientation requires an accurate map of the environment. However, creating maps requires known positioning. Because indoor localization is difficult, creating indoor maps is also difficult. SLAM is a common method for resolving this paradox, solving for the position and map simultaneously. Using low SWAP-C cameras alone presents an additional set of problems. Errors stemming from repetitive patterns, featureless environments, or incorrect loop closures can all result in unstable, or worse, completely divergent solutions.

The pivotal concept for this research is to combine the low SWAP-C cameras with an easy-to-use high SWAP-C mapping method. This truthing solution becomes feasible by making the initial SLAM pass as successful as possible, instead of trying to perfect vision-only SLAM with every low SWAP-C camera during every test. The problem of correct data association is solved by using uniquely-identifiable ArUco markers, resulting in optimal loop closures and landmark identification. Since the SLAM step is only performed once, additional sensors, in this case Light Detection and Ranging (LIDAR), can be used to improve accuracy.

With an accurate map of images, the indoor navigation paradox is resolved. Vision navigation using the mapped images can be done for any camera by recovering a relative translation and rotation between images. This enables position truthing without

permanent infrastructure, and without the complications of performing vision-based SLAM with low SWAP-C cameras.

In summary, this research demonstrates a method for a robust indoor truthing system based on camera pose estimation. It is based on a one-time use of high SWAP-C SLAM to enable future robust localization for low SWAP-C cameras.

Acknowledgements

I would like to thank everyone who has been part of my AFIT journey. The support from friends and family has been indispensable. I am particularly grateful to the ANT Center faculty and staff who made this thesis possible, especially my advisor, the committee, and their countless hours of help. I also need to thank my fellow Captains for their help, in and out of the lab. Most importantly, I wish to thank my wife and her unwavering support that made this all possible.

David W. Beargie

Table of Contents

Page

Abstract iv

Acknowledgements vi

List of Figures x

List of Tables xiv

I. Introduction 1

1.1 Background 1
1.2 Problem Statement 2
1.3 Research Goals 2
1.4 Contributions 2
1.5 Thesis Overview 3

II. Background and Related Research 4

2.1 Relevant Research 4
2.2 Simultaneous Localization and Mapping 5
2.3 Iterative Closest Point (ICP) Matching 6
LIDAR Scanners 7
Additional SLAM Methods 8
Georgia Tech Smoothing And Mapping (GTSAM) toolbox 10
Levenberg-Marquardt Optimization 10
2.4 Data Collection Methods 11
2.5 Optical Feature Descriptors 12
SIFT 12
SURF 12
ORB 13
2.6 Open Source Computer Vision Library (OpenCV) 13
2.7 Camera Characteristics 14
Pinhole Camera Model 14
Extrinsic Parameters 14
Intrinsic Parameters and Calibration 14
2.8 Multiple-Camera Registration 15
Essential and Fundamental Matrices 16
2.9 Sensor Synchronization 18
Frame Synchronization 18
External Timing 18

Page

III. Methodology . . . 19

3.1 Objective . . . 19
3.2 GTSAM Setup . . . 19
Prior Factors . . . 20
Odometry Factors . . . 21
Bearing-Range Factors . . . 22
Optimizer . . . 23
3.3 Equipment and Sensors . . . 23
ArUco Markers . . . 23
Velodyne HDL-64E S2 LIDAR . . . 26
FLIR Ladybug3 1394b Spherical Camera . . . 26
Mobile Cameras . . . 28
3.4 Coordinate Systems and Reference Frames . . . 28
3.5 Physical Characterization . . . 32
LIDAR to Camera . . . 35
Camera to Camera . . . 39
Experiment Overview . . . 40
3.6 Experiment 1 - Calculate Range Data from an Image . . . 40
3.7 Experiment 2 - Generate a Set of Geo-Referenced Images . . . 42
3.8 Experiment 3 - Collect Imagery in the Same Environment . . . 47
3.9 Experiment 4 - Extract Relative Pose from Matched Images . . . 48

IV. Results and Analysis . . . 51

4.1 Experiment 1 - Calculate Range Data from an Image . . . 51
Initial Test . . . 51
Verification Test . . . 57
4.2 Experiment 2 - Generate a Set of Geo-Referenced Images . . . 61
Data Collection . . . 62
Image Processing . . . 63
LIDAR Processing . . . 65
Optimization and Pose Extraction . . . 67
4.3 Experiment 3 - Collect Imagery in the Same Environment . . . 74
4.4 Experiment 4 - Extract Relative Pose from Matched Images . . . 75
Linked Image Database . . . 75
Matching a Test Image . . . 77
Pose Recovery . . . 83

Page

V. Conclusion 88

5.1 Overview 88
5.2 Conclusions 88
5.3 Research Significance 89
5.4 Future Work 90
Equipment Changes 90
Code Optimization 92
Machine Learning Approach 93
UcoSLAM 93

Appendix A. ArUco Marker To Bearing and Range Factor Script 95

Appendix B. GTSAM Script 97

List of Figures

Figure		Page
1.	Illustration of the iterative closest point method to align two lines [48].	8
2.	Epipolar geometry.	16
3.	Example factorgraph with pose and landmark factors.	20
4.	ArUco markers used as unambiguous features.	24
5.	ArUco marker printed with marker ID.	25
6.	Velodyne point cloud viewer.	27
7.	Matlab point cloud viewer.	27
8.	Camera capture rig.	29
9.	Camera color and white balance calibration card.	30
10.	Test environment with origin point.	31
11.	Camera reference frame.	32
12.	Camera and LIDAR mounted together.	33
13.	Camera and LIDAR platform on motorized ground vehicle.	34
14.	Translational offset between the LIDAR and the camera.	36
15.	Calibration object used for finding relative rotation offsets.	37
16.	Calibration object in each camera sensor.	38
17.	Rotational offset between the LIDAR frame and the s_1 frame.	38
18.	Checkerboard for determining intrinsic camera parameters.	40
19.	Subset of calibration images.	41
20.	Camera calibration.	41

Figure Page

21. ArUco marker number 1, 6x6. 43

22. Test setup for capturing images of ArUco markers at different distances. 44

23. ArUco marker placed at camera height. 47

24. Test environment, with landmark feature locations. 48

25. Training data used to model the area-distance relationship for an 82mm×82mm marker. 53

26. Curve fitting results with rational model for an 82mm×82mm marker. 54

27. Marker distance for 82mm marker 55

28. Distance error as a function of marker distance for 82mm×82mm marker. 55

29. Histogram and corresponding PDF when using the 82mm×82mm marker estimation function. 56

30. Training data used to model the area-distance relationship for an 55mm×55mm marker. 57

31. Curve fitting results with rational model for an 55mm×55mm marker. 59

32. Marker distance for 55mm marker 60

33. Distance error as a function of marker distance for 55mm×55mm marker. 60

34. Histogram and corresponding PDF when using the 55mm×55mm marker estimation function. 61

35. ArUco marker placement in the environment (left), and marker as viewed from s_1 (right). 62

36. LIDAR and camera frame synchronization. 63

37. ArUco keyframe selection 64

38. Output of the keyframe calculation function for all markers in the first pass. 65

Figure Page

39. Output of the keyframe calculation function for marker 1. 66

40. Estimated distance to each marker for each keyframe. 66

41. Plot of platform odometry in GTSAM without covariance. 68

42. Plot of platform odometry in GTSAM with noise model properly tuned. 69

43. Initialization of landmark factors in GTSAM. 70

44. GTSAM factorgraph after Levenberg-Marquardt optimization. 71

45. Landmark location and marginal before (solid line) and after optimization (dashed line). 72

46. Pose uncertainty decreases when passing a landmark. 73

47. Position and orientation of s_1 for frame 3189. 74

48. General path taken to capture mobile camera imagery. 75

49. Images captured from the DSLR (top), iPhone (middle), and GoPro (bottom). 76

50. Camera s_0 image with SIFT algorithm features shown. 78

51. Camera s_0 image with SIFT algorithm features shown. 79

52. Camera s_0 image with ORB algorithm features shown. 80

53. Rectified image from mobile camera, used to test image matching and pose recovery. 81

54. Detected SIFT features in the test image. 81

55. Detected SURF features in the test image. 82

56. Detected ORB features in the test image. 82

57. Top three image matches returned by computer. 83

58. SIFT features matched between images. 84

59. SURF features matched between images. 84

Figure Page

60. ORB features matched between images. 85

61. Pose recovery 86

List of Tables

Table		Page
1.	Epipolar geometry variables.	16
2.	GTSAM example factorgraph variable definitions.	21
3.	GTSAM odometry factor parameters.	22
4.	GTSAM bearing-range factor parameters.	22
5.	Velodyne HDL-64E S2 Specifications.	26
6.	FLIR Ladybug3 1394b Specifications.	28
7.	Bearing offsets between the spherical camera and LIDAR.	35
8.	Matlab calibration matrix variable definitions.	39
9.	Marker distance and area used to create a distance estimation model for a 82mm×82mm marker.	52
10.	Goodness of fit for different curve models based on RMSE of the 82mm×82mm marker data.	54
11.	Marker distance and area used to create a distance estimation model for a 55mm×55mm marker.	58
12.	Goodness of fit for different curve models based on RMSE of the 55mm×55mm marker data.	58
13.	Noise models for both marker sizes.	61

ASSESSMENT OF CAMERA POSE ESTIMATION USING GEO-LOCATED IMAGES FROM SIMULTANEOUS LOCALIZATION AND MAPPING

I. Introduction

1.1 Background

There is a need for a robust, easy to use indoor truthing system based on camera pose estimation. This truthing system would enable new advances in indoor and alternative navigation. Alternative navigation methods have been a subject of significant research in recent years. These methods attempt to augment or replace systems that rely on the Global Positioning System (GPS) and other satellite-based methods for navigation. This is especially important for indoor environments, where satellite signals are occluded by building infrastructure.

This thesis, along with the corresponding research at the Air Force Institute of Technology (AFIT) Autonomy and Navigation Technology (ANT) Center, attempts to develop a 6 Degree of Freedom (DOF) indoor navigation system for low Size, Weight, Power and Cost (SWAP-C) cameras that fulfills several key requirements:

1. There is no permanent infrastructure required to implement the system.
2. Localization and navigation must be performed without GPS.
3. The method must be sensor-agnostic, compatible with different camera types, brands, and characteristics.

The foundation of this method is an accurate map created by a robot, using a high SWAP-C camera and Light Detection and Ranging (LIDAR) Simultaneous

Localization and Mapping (SLAM). This map, which is difficult to make indoors, is then used to determine the position and orientation of many types of low cost cameras.

1.2 Problem Statement

This research answers several problems. First, can image features of known dimensions yield repeatably accurate range information using only monocular imagery? Next, can the relative poses of cameras with very different intrinsic characteristics be determined? Finally, can the true position and orientation of a camera in an environment be calculated within a reasonable amount of uncertainty?

1.3 Research Goals

The goal of this research is to recover the pose of a mobile camera within the environment, given an accurate map. The research attempts to collect monocular imagery data, establish a method of extracting bearing and range measurements from the data, and assesses the method's accuracy. It also uses the measurements to create a map using a SLAM solution. After data analysis, the thesis proposes methods for improving and refining the truthing process.

1.4 Contributions

This research contributes to the fields of mapping, navigation, and computer vision in these specific areas:

1. **Monocular ranging.** This research proposes a method for extracting range data from an image of a known feature.

2. **Method validation.** This research quantifies the accuracy of the distance estimation method using real world measurements.

3. **SLAM framework.** A SLAM optimization framework with specific factors and initial conditions is developed.

4. **Indoor navigation.** This research demonstrates an easily scalable navigation method.

5. **Computer vision feature algorithm evaluation.** This work compares and validates the use of different feature detection and matching algorithms to find the optimal setup for indoor navigation.

1.5 Thesis Overview

Chapter 2 contains the basic concepts and principles necessary to understand the thesis, including terminology and processes used for camera calibration, localization, and mapping. Chapter 2 also provides a review of previous work on robotic mapping and optical feature matching algorithms. Chapter 3 explains the specific equipment and SLAM factors used for the experiment. It also outlines the experiment methodology for this research; including the range extraction algorithm, data collection, pose optimization, feature detection and matching, and pose recovery.

Chapter 4 discusses and analyzes the results from the experiments in Chapter 3. It compares different feature matching algorithms and evaluates the different models for estimating range from feature size. Chapter 5 concludes this research by giving an overview of results and conclusions. It also outlines future work, including proposed equipment and methodology changes, that will develop and broaden the results of this research.

II. Background and Related Research

This chapter is divided into two parts. The first part is a study of relevant and related research. Next, there is a detailed explanation of the underlying principles and concepts used throughout the thesis, including SLAM, data collection, optical feature descriptors, computer vision libraries, camera characteristics, and sensor synchronization.

2.1 Relevant Research

Although the concept of using SLAM to determine localization using on-board sensors in unknown environments is already a proven concept, the mapping results from SLAM systems are not used often to enable localization with only optical sensors. It is common to use many different sensors and systems to perform SLAM. These sensor types include

1. Inertial Measurement Units (IMU) [10]
2. Wheel odometry [55]
3. Sound Navigation and Ranging (SONAR) [9]
4. Computer vision with stereo cameras [26]
5. LIDAR [1]

Bogoslavskyi, in [6] and [13], proposed new methods for improving the quality of matching consecutive LIDAR scans. Additionally, [41], and [51] have demonstrated the benefits of SLAM systems for robot localization.

Multiple methods of performing SLAM have been studied, including methods based on Extended Kalman Filters (EKF) [3, 42], Extended Information Filter (EIF)

SLAM [53], FastSLAM [34], grid-based SLAM [18], and graph-based SLAM [17]. Thrun, Montemerlo, and Stachniss have provided research papers on these SLAM techniques. In [33], Thrun presents a new method of SLAM, featuring a factoring algorithm to efficiently represent features. This factor graph approach is gaining popularity because the algorithm's complexity, compared to the Kalman-filter based approach, is reduced from $O(K^2)$ to $O(MlogK)$ time, where M is the number of particles, K is the number of features, and O is the algorithmic complexity.

2.2 Simultaneous Localization and Mapping

SLAM is a method for solving the localization paradox. If a position and orientation are precisely known, then generating a map of the local environment becomes trivial. Conversely, position and orientation can be determined quickly if the map is given. This interdependency can be circumvented by solving position updates concurrently with map updates, which is the foundation of SLAM. The idea of SLAM originated with Moravec and Elfes [40]. Smith, Self, and Cheeseman [49] improved the grid-based approach, while Durrant-Whyte and Leonard developed the feature-based SLAM method in [27].

SLAM can be separated into two sections. First, the "front end" comprised of the sensor data, landmark and environment information, and data association. The second section, or "back end", is where the information is processed into a optimal SLAM solution. Different front ends and back ends can be combined depending on the specific needs.

The SLAM acronym was first coined in a mobile robotics survey paper presented at the 1995 International Symposium on Robotics Research [15], where the mapping and localization problems were combined into a single estimation problem. The SLAM problem is a critical part of autonomous navigation. SLAM posits that it is possible

for a mobile robot to build up a map of an unknown environment while simultaneously determining its location within the map, with an unknown location in an unknown environment. Durrant-Whyte and Bailey's two tutorial papers on SLAM [54, 2] cover the basic principles and algorithms of SLAM in detail.

One key part of the algorithm development was the importance of the correlations between landmarks. The SLAM solution improves when the correlations between landmarks are stronger. Csorba developed much of the initial proof and theory of SLAM algorithm convergence in [11] and [12].

2.3 Iterative Closest Point (ICP) Matching

ICP is a method of matching two sets of corresponding point-clouds. The main goal of ICP is to determine a translation and rotation that minimize the sum of the squared error between corresponding point locations. The critical assumption behind ICP is that the correct correspondences are known. Without correct correspondences, the algorithm may terminate at a local minimum. If data association is an issue, having an accurate guess of rotation and translation can make the local minimum align with the global minimum.

ICP can trace its origins to the 2D template matching with the Singular Value Decomposition [56]. This method used a tiered approach, where a coarse registration reduced the computational intensity of the final cross-correlation calculation. The full ICP algorithm was applied to matching 3D shapes, curves, and surfaces in [5], and expanded to free-form shapes in [57]. A 2D example of ICP is shown in Figure 1. In the example, a set of points is chosen from each line. One of the point sets is iteratively moved and transformed to minimize the distance between each point set. Note that any inaccuracy in points chosen degrades the quality of the match.

LIDAR Scanners.

Laser scanners are a popular method for generating point clouds for SLAM, using either phase-shift or time of flight (TOF) methods. Measuring the doppler shift has the added benefit of providing the velocity of the reflecting surface along with the distance. TOF systems offer simplicity, measuring the time a laser pulse takes to travel to and from a surface. The travel time can be quickly converted to a distance using the equation:

$$d = \frac{1}{2}ct \tag{1}$$

where d is the distance, c is the speed of light, and t is the travel time of the laser pulse. Laser scanners have fast and accurate measurements compared to other sensors, each measurement only takes microseconds with an angular resolution of less than 0.5°. Angular resolution can be a disadvantage in feature-dense environments, since the laser pulse may not detect every object. Additionally, materials such as glass or water can have poor data returns. Sonar, which was originally an acronym for SOund Navigation And Ranging (SONAR), uses sound wave TOF in a method similar to the laser scanner.

A new method of ICP, proposed in [20], accounts for the local minimum trap by searching the point cloud data for corresponding features. These geometric features can be anything; including curvature, surface normal vectors, and point cloud density. The results demonstrated that the augmented ICP algorithm improves the convergence speed and the convergence interval, even without setting a proper initial estimate.

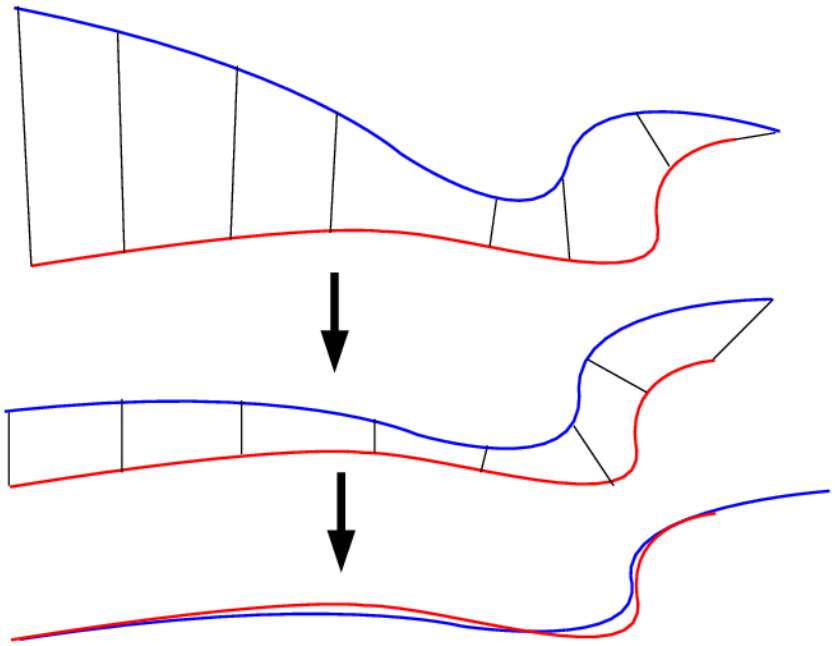

Figure 1: Illustration of the iterative closest point method to align two lines [48].

Additional SLAM Methods.

Since the introduction of SLAM in 1995, many different methods have been introduced. Some common types include Extended Kalman Filter (EKF) SLAM [50], FastSLAM 1 [34], FastSLAM 2 [32], Occupancy Grid SLAM [35], Oriented FAST and Rotated BRIEF (ORB) SLAM (ORB-SLAM) [37], and incremental Smoothing And Mapping (iSAM) [23].

EKF SLAM models a map of sensor and landmark states using a Gaussian variable. The two stages of EKF, prediction and correction, process the map. The prediction stage propagates the Gaussian variable through the filter to estimate the map at the next stage. The correction stage brings in sensor data to improve the SLAM solution. EKF SLAM is causal, but a forward-backward smoother can be added to improve the solution even further. Solà, in [50], presents a good tutorial of EKF SLAM including practical examples.

FastSLAM 1 and FastSLAM 2 were first introduced by [34] and [32], respectively. They have quickly overtaken EKF SLAM in popularity. While very similar to EKF SLAM, FastSLAM 1 substitutes a Rao-Blackwellised particle filter for the EKF. FastSLAM 2 improves the SLAM solution by using a different proposal distribution than the original FastSLAM algorithm. These algorithms converge well for linear problems, but have difficulty representing highly non-linear systems [3].

Occupancy Grid SLAM reduces the impact of data-disassociation by representing the environment as a block of cells. As the sensors traverse the environment, sensor data is used to determine the probability that a cell is empty or full. These probability maps, called occupancy grid maps, were introduced to SLAM in [35]. Millstein's research in [31] demonstrates a thorough application of Occupancy Grid SLAM.

ORB-SLAM, introduced in 2015 [37], is a camera-based SLAM approach that uses image features for every aspect of SLAM, from mapping and localization to handling loop closures. ORB-SLAM has several advantages besides using one sensor for the entire system. It operates in real time, in a wide range of environments and scales, and only increases in size if the camera environment changes.

The iSAM method was introduced in 2008 [23] as an exact and efficient SLAM solution. It is very good for non-linear systems since all previous poses are kept as part of the estimation stage. The four characteristics that are met by iSAM are:

1. Exactness: no approximations can be made, even with nonlinear measurements.

2. Efficiency: data processing speed must be adequate for real-time use

3. Data associations: information for data association must be readily available.

4. Applicability: iSAM must be able to solve a wide range of problems on many different data types.

The effiency in iSAM is possible by only re-factoring when a large change is present, instead of every time a measurement is made.

Georgia Tech Smoothing And Mapping (GTSAM) toolbox.

GTSAM is based The GTSAM toolbox was developed at the Georgia Institute of Technology by Dellaert along with many of his students and collaborators [14]. GTSAM is a C++ library for solving factor-graph based problems. In addition to modeling and solving SLAM problems, it can also solve Structure From Motion (SFM) problems and more complex estimation problems. Along with the C++ library, GTSAM also provides a MATLAB interface, used extensively for this research, which allows for algorithm development, modeling, visualization, and user interaction.

Factor graphs are bipartite graphs with two components, factors and variables. Factors represent probabilistic information, derived from measurements or prior knowledge. Variables represent random variables. Koller and Friedman provide tutorials on factor graphs, along with other graph-based models including Bayes networks, in [24]. Georgia Tech has been making many strides in navigation and mapping, including SLAM with 3D and 2D Sensors [52] and analyzing the impact of sensor precision on SLAM accuracy [44].

Levenberg-Marquardt Optimization.

The Levenberg-Marquardt (LM) optimization algorithm has become a standard method for solving non-linear least-squares problems. It is based on the work of Levenberg [28] and Marquardt [30]. It is an iterative technique that locates the minimum of a multivariate function that is expressed as the sum of squares of non-linear real-valued functions. The LM algorithm can be considered as a combination of the gradient descent method and the Gauss-Newton method. If the multivariate

function is far from the optimal solution, the algorithm approximates the gradient descent algorithm. If the function is close to optimal, it approximates the Gauss-Newton method. The algorithm's implementation and theory are described in detail in [36].

2.4 Data Collection Methods

Different SLAM algorithms and methods can utilize a wide range of sensors. The most common type are exteroceptive sensors including laser scanners, sonar, and cameras. Combinations of these sensors are often used, or paired with interoceptive sensors such as wheel encoders or inertial sensors. These sensors only measure position indirectly, using velocity and acceleration, they do not interact with the external environment. Exteroceptive sensors, while expensive compared with wheel odometry and commercially-available inertial sensors, offer more accurate long-term measurements. Often, both types of sensors are used together because of their complimentary characteristics.

SLAM relies on an estimate of motion between successive poses, often referred to as odometry. This motion is measured using the sensors above. Odometry from wheel encoders measures the rotation of each wheel, using the wheel radius and rotation to determine the vehicle's motion. Inertial sensors include accelerometers and gyrometers, which measure linear and angular acceleration. These sensors drift significantly in the long-term, making them insufficient for SLAM by themselves. However, they are very stable in the short-term, smoothing the jitter inherent to many exteroceptive sensors.

2.5 Optical Feature Descriptors

Optical feature descriptors are features that can be repeatably found in images of a specific scene. These features can be identified even after changes in rotation, scale, and lighting conditions. Many different algorithms have been developed, each with their own benefits and problems. This section details the specific algorithms used for this research along with the specific toolbox used to apply the detectors to each image.

SIFT.

The Scale Invariant Feature Transform (SIFT) was developed by D. G. Lowe in 2004 [29], it has since become a widely-used feature detection and description algorithm. The SIFT detector uses an approximation of a Laplacian-of-Gaussian (LoG) operator, called a Difference-of-Gaussians (DoG) operator: Feature-points are detected by searching local maxima using DoG at various scales of the subject images. A 1616 pixel box around each detected feature is then further analyzed and broken into sub-sections or blocks, rendering a total of 128 bin values. SIFT is computationally intensive, but is highly invariant to rotations, scale, and affine changes.

SURF.

The Speeded Up Robust Features (SURF) detector was introduced in 2008 [4]. it relies on a tiered Gaussian analysis of images and the determinant of a Hessian Matrix. Feature detection speed is improved by using only using 64 bins. The main advantage of SURF over SIFT is its lower computational cost. While SURF features are also invariant to rotation and scale, they are not as invariant to affine changes. The descriptor can be extended to 128 bin values to cope with affine variance at the cost of computation time.

ORB.

ORB, was developed by E. Rublee et al. in 2011 [46]. It is a combination of two modified feature detection algorithms: Features from Accelerated Segment Test (FAST) [45] for the feature detection, and Binary Robust Independent Elementary Features (BRIEF) [8] for the feature description. FAST corners are detected at varying scales, then the detected corners are processed with a Harris Corner Detector to select the most optimal points. The original BRIEF description method is unstable with respect to rotation, so modification is required to make ORB features rotation invariant. ORB features are also invariant to scale and some affine changes.

2.6 Open Source Computer Vision Library (OpenCV)

OpenCV is an open source computer vision and machine learning software library [1]. It provides a framework for designing and testing multiple computer vision applications. OpenCV is a Berkley Software Distribution (BSD)-licensed library, which makes it popular for academic and commercial settings alike. The library has been downloaded over 14 million times, with an estimated 50 thousand users. This popularity has built a large community of users and collaborators, and grown to include over 2500 algorithms for computer vision and machine learning [7]. OpenCV Forums are a resource for tutorials, operating instructions, and instructional manuals. Along with the OpenCV Forums, [22] was used extensively to accomplish this research.

OpenCV supports Windows, Linux, Android and Mac OS operating systems. It is written natively in C++, but it also integrates with Python, Java, and MATLAB. OpenCV algorithms will be used specifically for image calibration and feature matching in this thesis.

[1] https://opencv.org/

2.7 Camera Characteristics

Camera models are used to describe the relationship between information from the picture, on the image plane, and the actual world. The models enable comparisons between different cameras. The camera model used for this research is the pinhole camera model, which is widely used.

Pinhole Camera Model.

The most simplistic model of a camera is the Pinhole Camera model. A pinhole camera has a single small aperture (a pinhole) and no lens. Light passes through the aperture and projects an inverted image on the rear of the camera, or image plane. The distance from the aperture to the rear of the camera is known as the focal length. There is also a virtual image plane, one focal length in front of the aperture, which contains the non-inverted image. Many cameras can be fully described by using a pinhole camera along with additional intrinsic and extrinsic parameters. These additional parameters, introduced in [21], vary from sensor to sensor.

Extrinsic Parameters.

Extrinsic parameters describe the position and orientation of the camera. This can either be the relative to an origin point and axis in the environment, or relative to a second camera or image. External parameters change whenever a camera is moved or re-oriented. The extrinsic parameters transform world feature points into camera coordinates, and from camera coordinates into world feature points.

Intrinsic Parameters and Calibration.

Intrinsic Parameters transform camera coordinates into the image plane. Some of the parameters included are focal length, radial lens distortion, tangential lens distor-

tion, and sensor skew. Because these parameters are different for each camera, image matching algorithms require a calibration step to account for the camera differences. Images from a camera can be rectified using the camera calibration parameters. Rectifying an image corrects for the different types of distortion. There are two prominent methods of calibration. The first, known as self-calibration or auto-calibration, takes non-calibrated images from multiple views of an object to calculate the calibration matrix [19].

The second method is a batch approach which uses at least 10 images of a planer pattern with measured feature sizes, for instance a chess board. This method is more accurate and more common than self-calibration. First, specific points or corners are identified, Then the coordinates in real-world space and the coordinates in each image are used to solve for the distortion coefficients. Because of the method's popularity there are many applications and tutorials detailing its procedures. Lambers presents a calibration approach using OpenCV [25].

2.8 Multiple-Camera Registration

Images of unique features taken from multiple cameras can be used to determine the extrinsic parameters of the cameras. That is, the rotation and translation that separate the sensor coordinate frames. This process works through the use of epipolar geometry, shown in Figure 2. The variables used, listed in Table 1, are as follows: Point q is an optical feature that appears in both cameras. O_1 and O_2 are the camera's optical centers, separated by some baseline. The epipolar plane is the plane that contains the points q, O_1, and O_2. The lines l_1 and l_2 are where the epipoplar plane intersects each image plane. Each camera also has an epipole, or the point where the baseline intersects the image plane. These epipoles are e_1 and e_2 for sensor 1 and sensor 2, respectively. The point q appears in each image plane at point p_1

and p_2. This works well for applications such as stereo-vision navigation and ranging, when the baseline is either fixed or measurable.

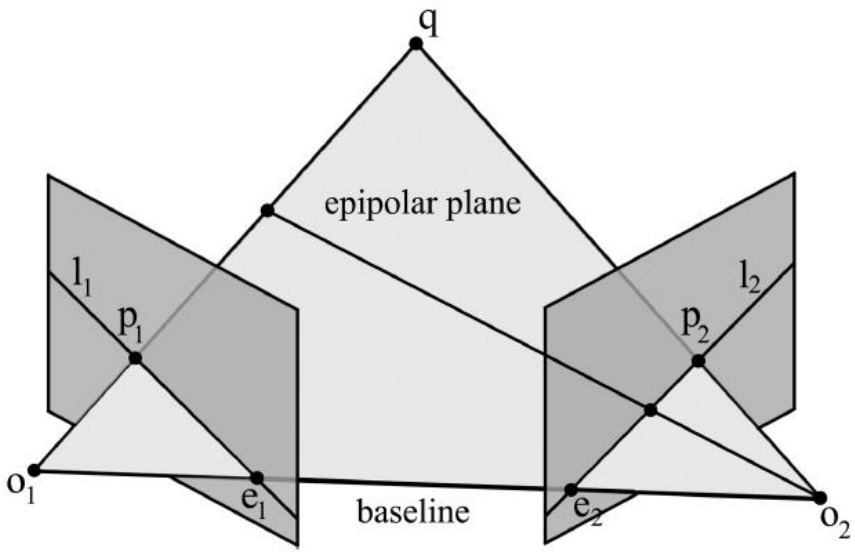

Figure 2: Epipolar geometry.

Table 1: Epipolar geometry variables.

Variable	Definition
q	optical feature point
O_1	camera 1 optical center
O_2	camera 2 optical center
l_1	intersection of epipolar plane and image plane 1
l_2	intersection of epipolar plane and image plane 2
e_1	camera 1 epipole
e_2	camera 2 epipole
p_1	q projected onto image plane 1
p_2	q projected onto image plane 2

Essential and Fundamental Matrices.

Epipolar geometry can be expressed in camera coordinates using the Essential matrix $\boldsymbol{E}$, a 3×3 matrix that only depends on the extrinsic parameters $\boldsymbol{R}$ and $\boldsymbol{T}$.

The Essential matrix consists of five parameters (three rotation parameters and two translation parameters). It has two constraints:

1. the determinant is zero.

2. the two non-zero singular values are equal.

$$\mathbf{E} = \mathbf{R} \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{2}$$

Epipolar geometry can also be can be expressed in image coordinates by using the Fundamental matrix $\boldsymbol{F}$, which depends on the extrinsic and the intrinsic parameters. The Fundamental matrix always has a rank of two, and contains seven parameters (two for each of the epipoles and three for the homography between the two epipolar lines). If the cameras are calibrated, then the point location from each sensor is given with respect to its camera's coordinate frame and the Essential matrix can be used. The Fundamental matrix can be used with uncalibrated cameras.

To recover a relative pose between two cameras, the Essential matrix must be derived from sets of corresponding points in images from two calibrated cameras. Using the Essential matrix and the pose estimation algorithm, introduced by David Nistér in 2004 [39], the relative $\mathbf{R}$ and $\mathbf{T}$ matrices can be recovered. Although Nistér's algorithm only requires five data points to be tractable and convergent, it often requires many more to account for noise and co-planar features [43].

2.9 Sensor Synchronization

When using multiple sensors, it is necessary to synchronize the data between sensors. To minimize timing errors, the signals can be triggered by a single signal, or synchronized using frames with feature common to all sensors. This thesis uses the frame synchronization method which, while less accurate than using single timing sources, is much simpler to implement.

Frame Synchronization.

When using frame synchronization, timing differences between sensors are calculated by interpolating between sets of data. This is done by identifying a temporal feature at the start and end of each data set. It can be as simple as observing the start and end of movement, or using specific movements at determined frequencies. This method is practical for short tests with a limited number of sensors and if a timing error of $\frac{1}{framerate}$ is acceptable.

External Timing.

External timing sources can trigger data collection on most sensors. They can run on their own internal frequency generator, or be synchronized to a global reference such as GPS time if required. Sensors can have very different signal requirements, ranging from a single low-voltage pulse to an Ethernet data packet, oftentimes through a dedicated synchronization port. Depending on the type of signal and the accuracy of the timing circuit, synchronization within 10 ns is possible.

III. Methodology

3.1 Objective

The purpose of this research is to test the efficacy of using LIDAR and camera sensors to generate a reference database of images (taken from SLAM-derived positions) and use that database to determine the location and orientation of a new image. The experiment execution necessitates the use of multiple sensors, so calibration and image rectification will be used to properly characterize the sensor-to-sensor relationships and sensor-to-feature relationships. The specific experiments aim to accomplish several goals:

1. Measure the accuracy of calculating range data from an image with known feature sizes.
2. Generate a set of geo-referenced images using SLAM techniques.
3. Collect sets of images using the same environment as the SLAM dataset.
4. Solve for the 3D position and orientation of an camera using the SLAM dataset.

This chapter begins with the setup and factors used by GTSAM. Then an overview of the important markers, equipment, and sensors used in the experiments. Next, there is a description of the different coordinate systems and frames, followed by the physical relationships between sensor frames. Finally, there are overviews of each experiment.

3.2 GTSAM Setup

This thesis uses a 2D landmark-based SLAM problem, with prior factors, odometry factors, and landmark measurement factors. Figure 3 is an example of a factor

graph for a landmark-based SLAM problem. The variables are listed in Table 2. There is one landmark l_1, along with three pose variables: x_1, x_2, and x_n, representing the poses of the platform over time. As shown in the example, there is a factor p on the pose x_1 that encodes prior knowledge about x_1, two factors that relate successive poses: o_1 and o_2, and two factors that relate poses to the landmark: br_1 and br_2.

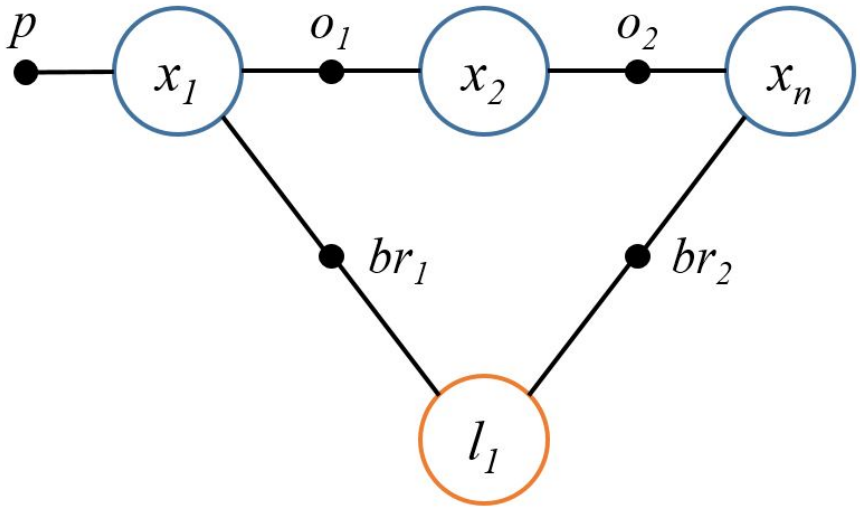

Figure 3: Example factorgraph with pose and landmark factors.

Prior Factors.

Prior factors are unary factors that encode the prior knowledge of each pose or landmark. They describe a probability of an initial condition. In this thesis, there is a prior factor based on the approximate starting location of the platform, as well as the approximate locations for each of the landmark features. These are passed to GTSAM as *PriorFactorPose2* functions. The *PriorFactorPose2* function requires certain parameters, including the pose or landmark ID and the position constraint.

Table 2: GTSAM example factorgraph variable definitions.

Variable	Factor Used	Definition
p	*PriorFactorPose2*	prior factor
x_1	-	1^{st} pose
x_2	-	2^{nd} pose
x_n	-	n^{th} pose
l_1	-	landmark
o_1	*BetweenFactorPose2*	odometry factor from x_1 to x_2
o_2	*BetweenFactorPose2*	odometry factors from x_2 to x_n
br_1	*BearingRangeFactor2D*	factor from x_1 to l_1
br_2	*BearingRangeFactor2D*	factor from x_n to l_1

Each prior factor also has an associated noise model which can account for measurement uncertainty. This is an Example of a prior factor in MATLAB that adds a prior from pose 1 to the origin:

```
PriorFactorPose2(x1, Pose2(0, 0, 0), noiseModel.Diagonal.Sigmas([1; 1; 0.1]));
```

Odometry Factors.

Odometry factors are binary factors that relate two poses. This is expressed in GTSAM by the *BetweenFactorPose2* function, which is expressed in MATLAB using the command:

```
BetweenFactorPose2(x1, x2, odometry, noise))
```

The function input parameters are described in detail in Table 3. This thesis only uses the odometry factors to relate consecutive platform poses. For example: pose 1 to pose 2, pose 2 to pose 3, pose n-1 to pose n. The parameters required for an odometry factor are shown in Table 3.

Table 3: GTSAM odometry factor parameters.

Parameter	Definition
$x1$	1^{st} pose ID
$x2$	2^{nd} pose ID
odometry	relative translation and rotation from the 1^{st} pose ID to the 2^{nd} pose ID
noise	diagonal Gaussian noise model for the x position, y-position, and orientation

Bearing-Range Factors.

Binary factors can also be used to relate a pose and a landmark. This uses a different GTSAM function, the *BearingRangeFactor2D* function, which uses the relative bearing and range from a pose to a specific landmark. Ensuring that measurements are associated with the correct landmark is crucial to SLAM. Incorrect matches between measurements and landmarks can have a detrimental impact on the SLAM result. This research simplifies the data association problem by using the ArUco marker ID to assign a unique identifier to each landmark.

This function is expressed in MATLAB using the command:

```
BearingRangeFactor2D(x1, l1, Rot2(angle), range, noise)
```

each parameter is described in detail in Table 4.

Table 4: GTSAM bearing-range factor parameters.

Parameter	Definition
$x1$	Pose ID
$l1$	Landmark ID
Rot2(angle)	Relative bearing to landmark
range	Range to landmark
noise	diagonal Gaussian noise model for the bearing and range

Optimizer.

The GTSAM factors and variables, in a factor graph structure, are compiled into a an optimizer along with any initial conditions. Because the odometry factors involve the orientation and movement of the platform, which are not linear, the optimizer also needs to be non-linear. The optimizer class linearizes the factor graph multiple times to minimize the non-linear squared error specified by the factors. The Levenberg-Marquardt algorithm, described in detail in Subsection 2.3, is a good choice because it often converges, even if it starts very far off the final minimum. The MATLAB command for the Levenberg-Marquardt optimizer is:

```
LevenbergMarquardtOptimizer(graph, initial);
```

3.3 Equipment and Sensors

ArUco Markers.

Because loop closures and revisits are such a critical part of SLAM, there is a need to create unambiguous data associations between multiple views of a single feature. The features in this case are ArUco markers, which have unique IDs that can be detected and processed using OpenCV. ArUco markers are available in a wide range of libraries, depending on how many unique features are available and how many bits each marker has. For this thesis, the 6x6 library of 250 unique markers provides enough unique IDs without having a higher bit density. Only the first 25 6x6 markers, shown in Figure 4, are used. Each marker was printed at a scale of 82 mm by 82 mm with the unique ID below each marker, as shown in Figure 5, since the markers are not human-readable.

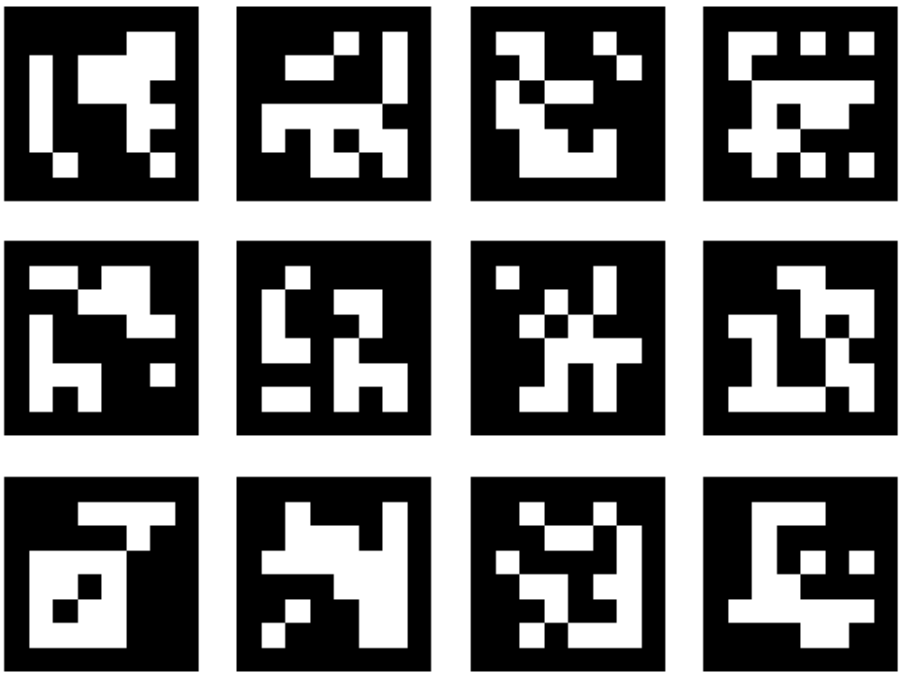

Figure 4: ArUco markers used as unambiguous features.

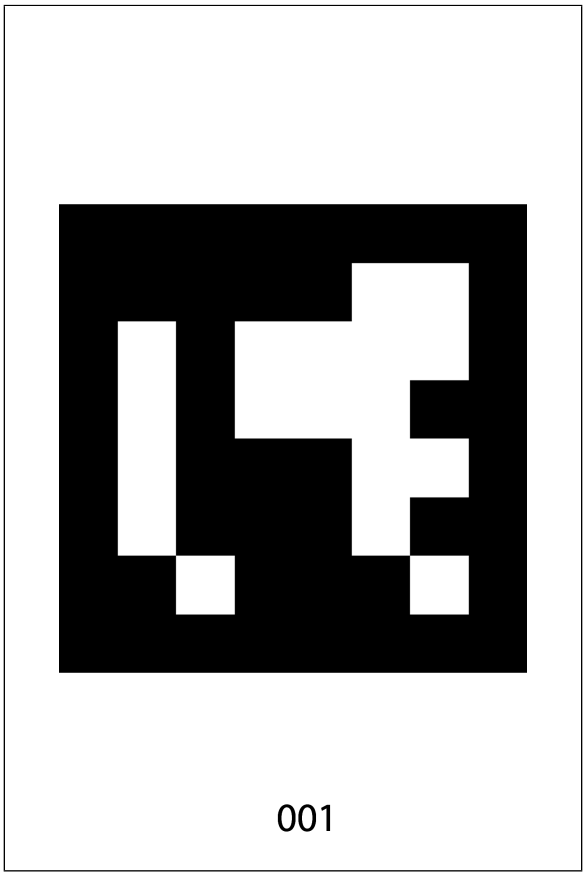

Figure 5: ArUco marker printed with marker ID.

Velodyne HDL-64E S2 LIDAR.

The LIDAR used for the experiments is the HDL-64E, version S2, from Velodyne. The HDL-64E has 64 lasers, mounted in pairs of 32 lasers on upper and lower blocks. Both blocks are mounted onto a rotating platform, spinning at 600 rotations per minute. This design provides a much richer point cloud than previous designs in which a singe laser is fired through a rotating mirror. The HDL-64E has the specifications listed in Table 5. Several methods have been studied for improving the accuracy of the HDL-64E [16]. Velodyne also provides software for viewing, calibrating, and parsing the raw LIDAR data into other formats. An example of the velodyne viewer is shown in Figure 6, compared to the MATLAB point cloud viewer in Figure 7.

Table 5: Velodyne HDL-64E S2 Specifications.

Parameter	Value	Unit
Spin Rate	300-1200	RPM
Vertical Field Of View	26.8	degrees
Horizontal Field Of View	360	degrees
Sensing Range	120	m
Range Accuracy	15	mm
Output Capture Format	PCAP	-

FLIR Ladybug3 1394b Spherical Camera.

The camera used to capture the geo-located images is the FLIR Ladybug3 camera. It is a comination of six 2 MP sensors with global shutters that enable the system to collect video from more than 80% of a full sphere. The data from each camera is collected and transmitted over an IEEE-1394b (FireWire) interface. JPEG-compressed 12MP resolution images can be captured up to 15fps. The Ladybug3 was designed for weather-resistant high-resolution, direct FireWire, and synchronized image capture. These specifications are listed in Table 6.

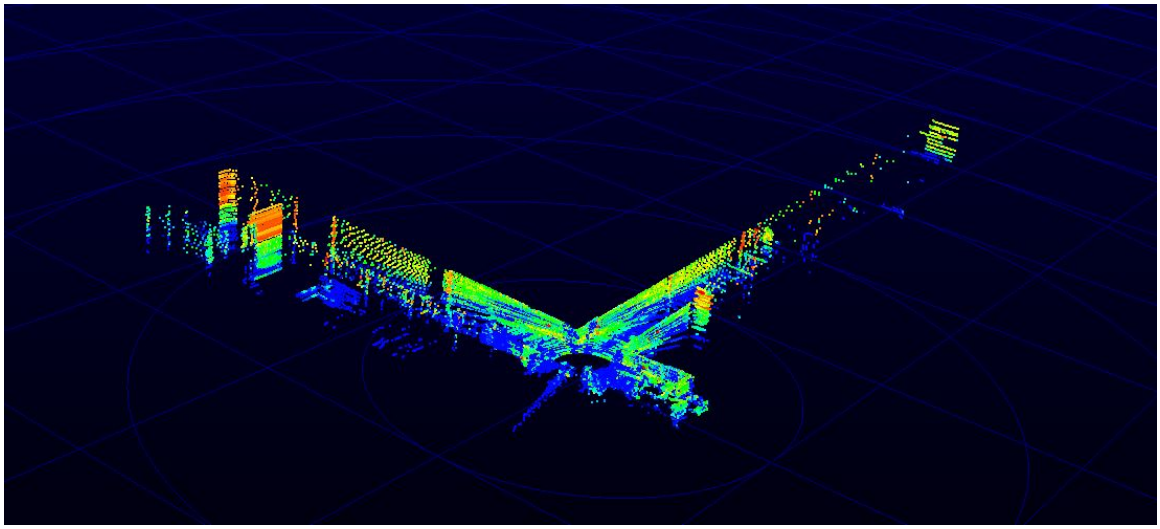

Figure 6: Velodyne point cloud viewer.

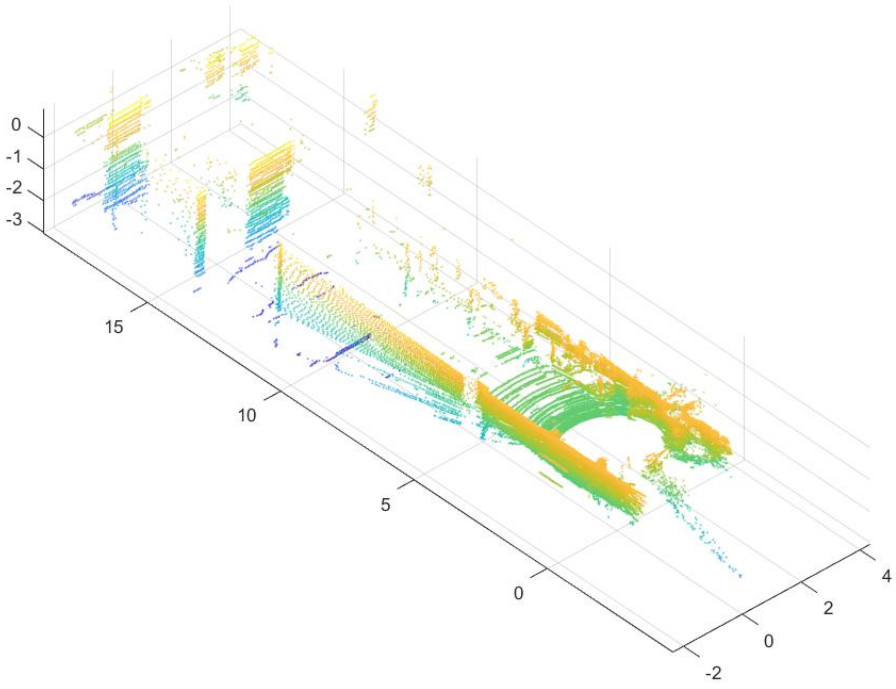

Figure 7: Matlab point cloud viewer.

Table 6: FLIR Ladybug3 1394b Specifications.

Parameter	Value	Unit
Spherical Coverage	80	percent
Resolution	2	MP
Frame Rate	15	fps
Interface	IEEE-1394b	-
Output Capture Format	JPEG	-

Mobile Cameras.

To collect the images that compare against the geo-referenced image set, three mobile cameras are used. These are a Digital Single Lens Reflex (DSLR) camera, a mobile phone camera, and a high-speed action camera. This combination represents a wide range of sensors. The DSLR used is a Nikon D7000 with a wide angle lens, collecting 2.074 MP at 30fps. The mobile phone sensor is an Apple iPhone 6s, collecting 2.074 MP at 60fps. Finally, the action camera is a GoPro Hero4 Silver collecting 2.074 MP at 60fps.

All three cameras are connected together using the camera rig shown in Figure 8. Recording is started for the cameras and the rig is pointed at the color calibration card in Figure 9, which is used as a reference for white balance and color correction. Finally, the rig is moved through the hallways, following the path travelled by the geo-referenced camera at a maintained height of approximately 1.5 meters. The rig is oriented straight ahead for one pass, 45° right for one pass, 45° left for one pass, and straight ahead for the final pass.

3.4 Coordinate Systems and Reference Frames

The experiments conducted in this research require cooperation between several different sensors and environments, which have their own unique frames of reference.

Figure 8: Camera capture rig.

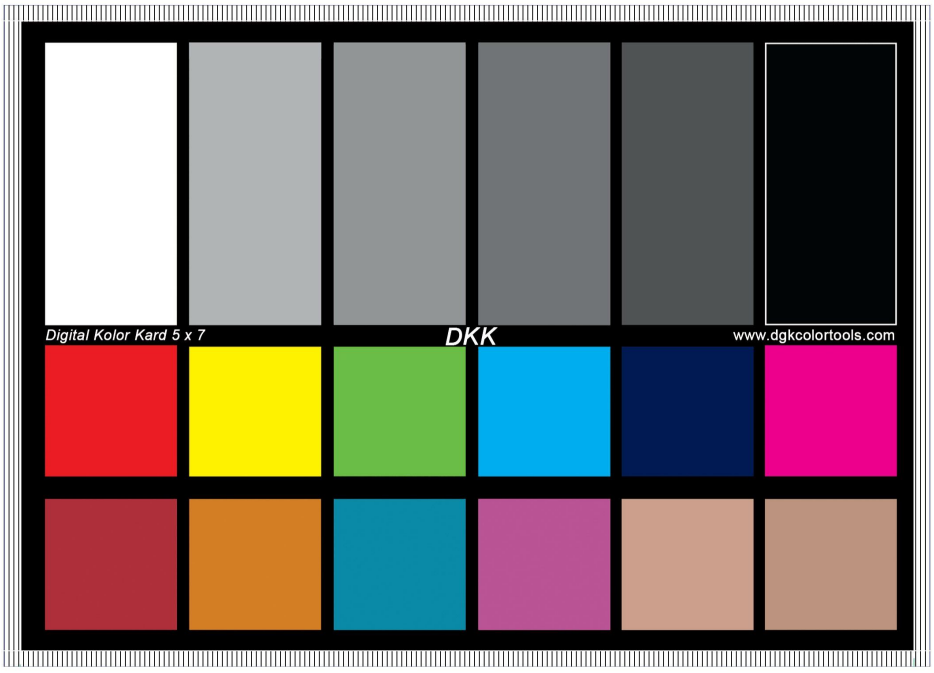

Figure 9: Camera color and white balance calibration card.

These different reference frames, such as the environment frame, the mobile camera frame, and the platform frame, encompassing the LIDAR frame and the spherical camera frame. Although there is also some global reference frame, it is unused since every sensor can be related to a common local environment. The environment covers the area of AFIT in which the experiments are conducted. The environment is mapped by overlaying a 3-dimensional right-handed Cartesian coordinate system onto the hallways where the positive x-axis is West, the positive y-axis is South, and the positive z-axis is up. This orientation is shown along with the test environment in Figure 10.

The mobile camera frame follows common convention as shown in Figure 11, where the sensor lays on the x-y plane and the z-axis is out the front of the camera. This is the same for all three cameras. The frame of the motorized platform that houses the LIDAR and the camera is simplified by sharing a common origin, namely the center

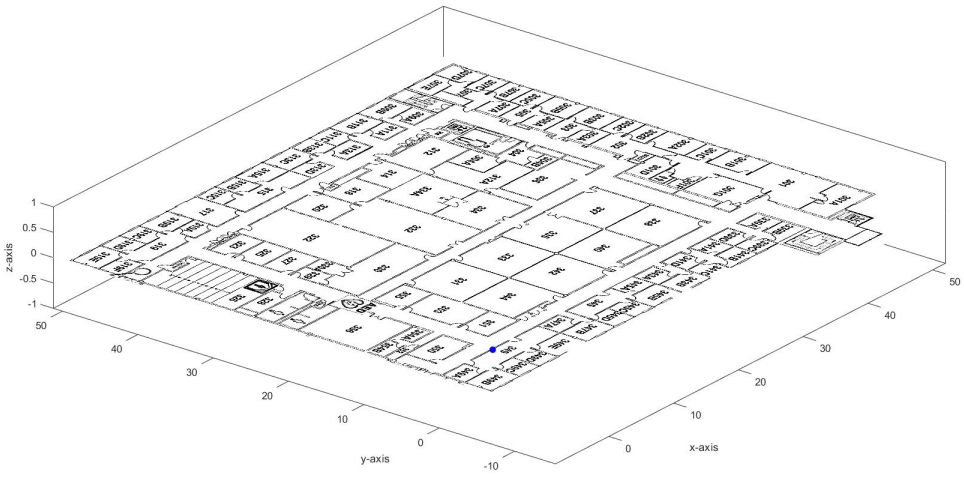

Figure 10: Test environment with origin point.

of the LIDAR sensor. The x-axis is oriented in front of the platform, the y-axis to the left, and the z-axis up. The raw LIDAR data is given with the y-axis out the front of the sensor, the x-axis to the right, and the z-axis up. The LIDAR can therefore be related to the platform by a yaw of 90°, or by the DCM:

$$R_{LIDAR}^{platform} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Applying this DCM to the LIDAR data orients both frames, making the primary motion of the platform and the LIDAR fall along the x-axis.

The spherical camera is rigidly affixed to the platform. Since the panoramic image is comprised of six individual sensors, each sensor has its own coordinate frame, following the convention used on the mobile cameras above. These sensors will be

denoted as s_0 through s_5, and their rotational offsets from the LIDAR frame will be discussed in Subsection 3.5.

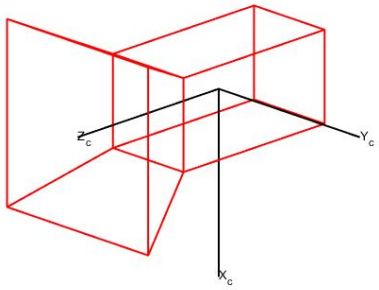

Figure 11: Camera reference frame.

The camera and LIDAR are both attached to an aluminum platform, shown in Figure 12, so the offsets remain constant throughout the experiment. To maintain a constant height, the platform traverses the environment on a motorized ground vehicle. The vehicle carries the power supply and data recorder for the sensors. This vehicle is shown with the platform attached in Figure 13.

3.5 Physical Characterization

The first step of each experiment is to determine the physical characteristics of the system, namely the translation and rotation from one frame of reference to another. To generate a set of geo-referenced images using SLAM, the corresponding relationships between LIDAR, camera, and landmark features must be well-understood. Additionally, the intrinsic parameters of each camera must be determined to account for effects such as focal length and lens distortion.

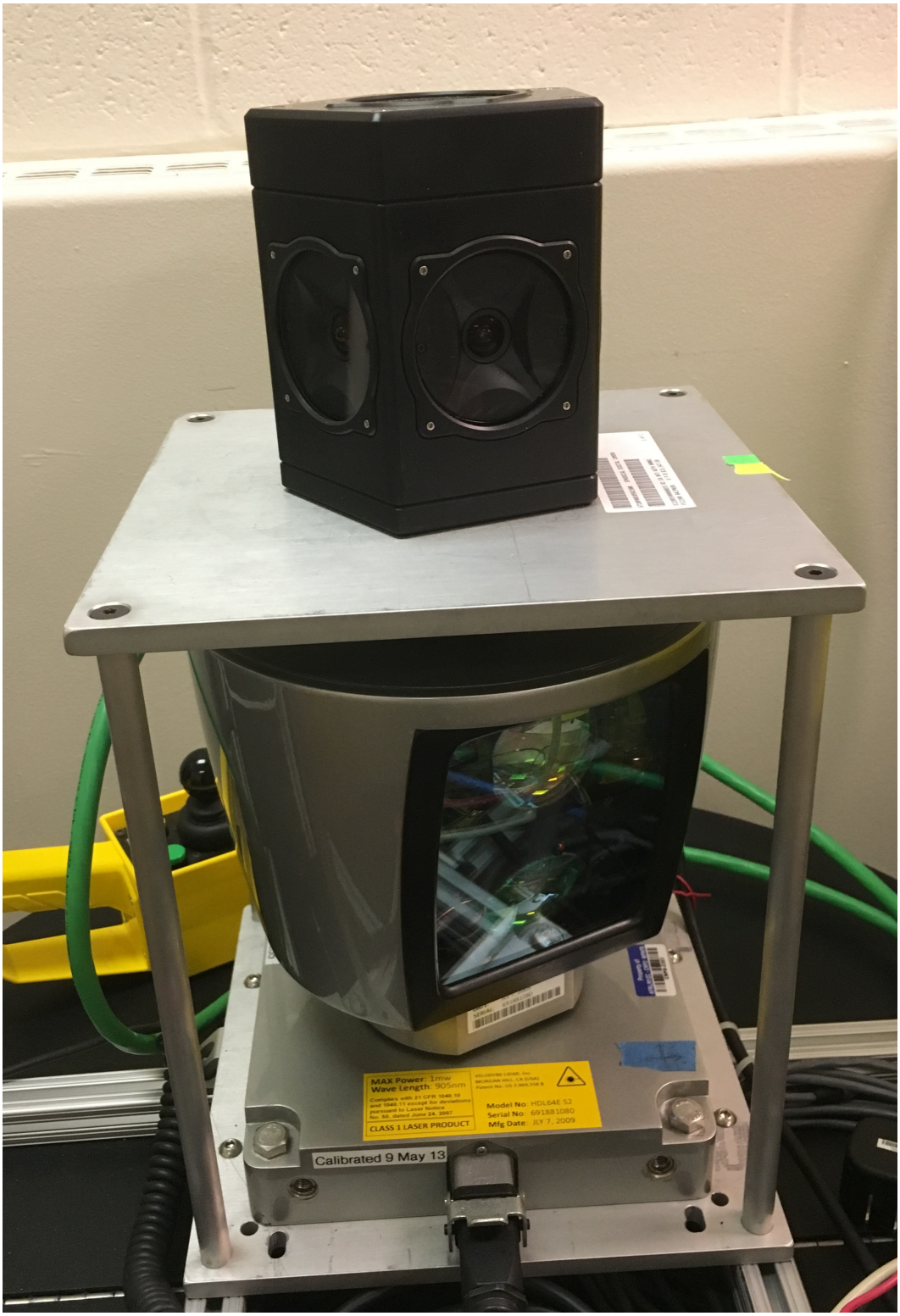

Figure 12: Camera and LIDAR mounted together.

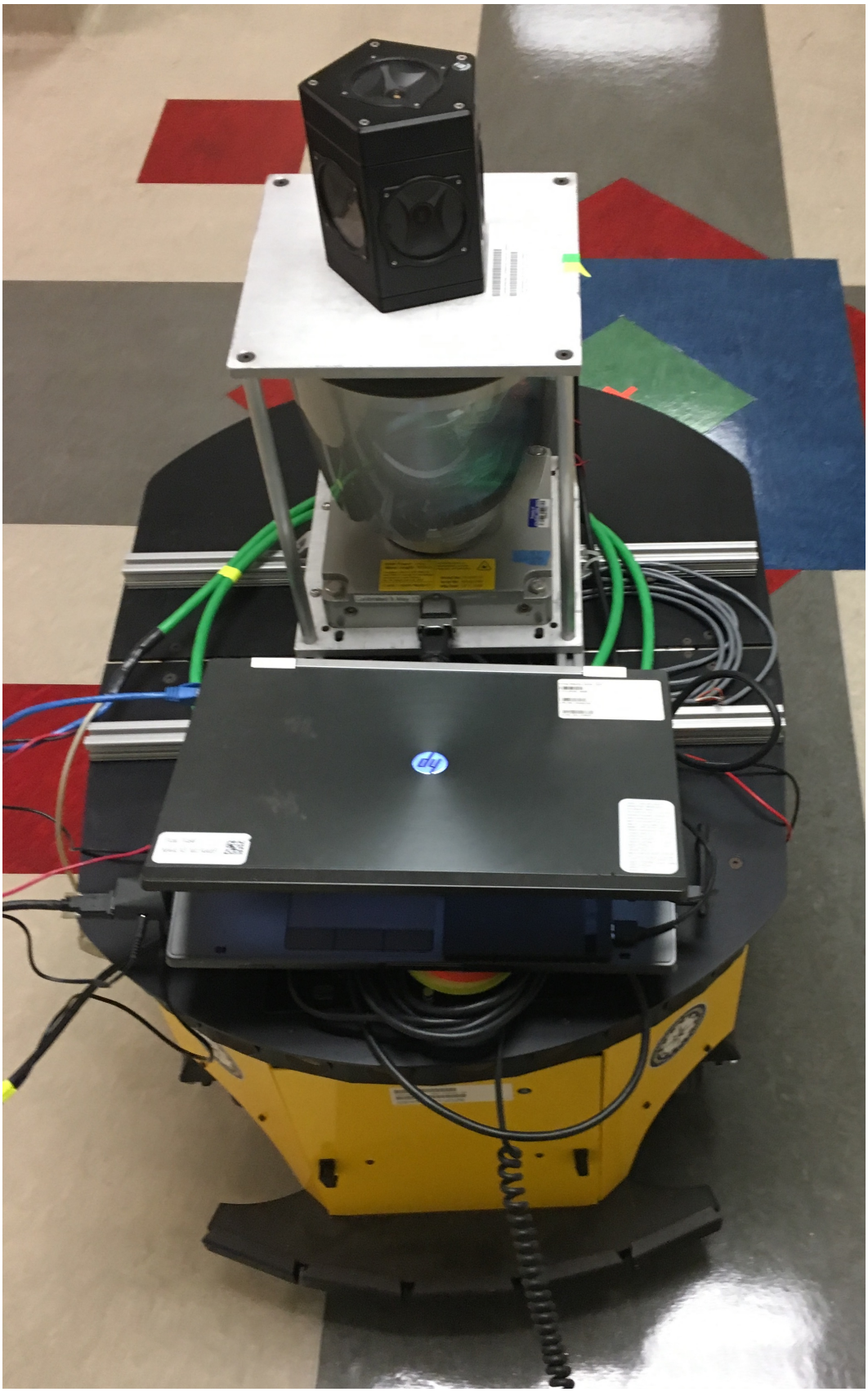

Figure 13: Camera and LIDAR platform on motorized ground vehicle.

LIDAR to Camera.

Determining the relationship between the LIDAR frame and the camera frame is a combination of physical measurements and data analysis. The relationship remains constant throughout the experiment because the LIDAR and the camera are rigidly affixed to the same structure, as shown in Figure 14. The translational offset between frame origins is calculated by the lateral, longitudinal, and vertical distance between the sensor origins.

Calculating the rotational offset between sensors requires identifying a feature in both sensor data sets. This is accomplished by finding an object in an image that is separate from any background objects.

The calibration rod can be seen in Figure 15. When the platform is rotated, it appears in each sensor frame as shown in Figure 16. The tall and thin rod has enough data to determine relative roll and yaw. Since the LIDAR and combined sensors both have 360 degree horizontal fields of view, relative pitch can be calculated by measuring the roll of the other image sensors. The resulting rotational offset for the sensor capturing landmark data is $-86.0279°$, shown in Figure 17. Table 7 shows the offsets for each individual sensor.

Table 7: Bearing offsets between the spherical camera and LIDAR.

camera sensor	cam-marker	LIDAR-marker	LIDAR-sensor
0	-0.110°	-17.040°	-16.93°
1	-3.123°	-89.151°	-86.0279°
2	-2.372°	-165.350°	-162.978°
3	-4.450°	117.432°	121.8812°
4	3.489°	59.928°	56.4392°

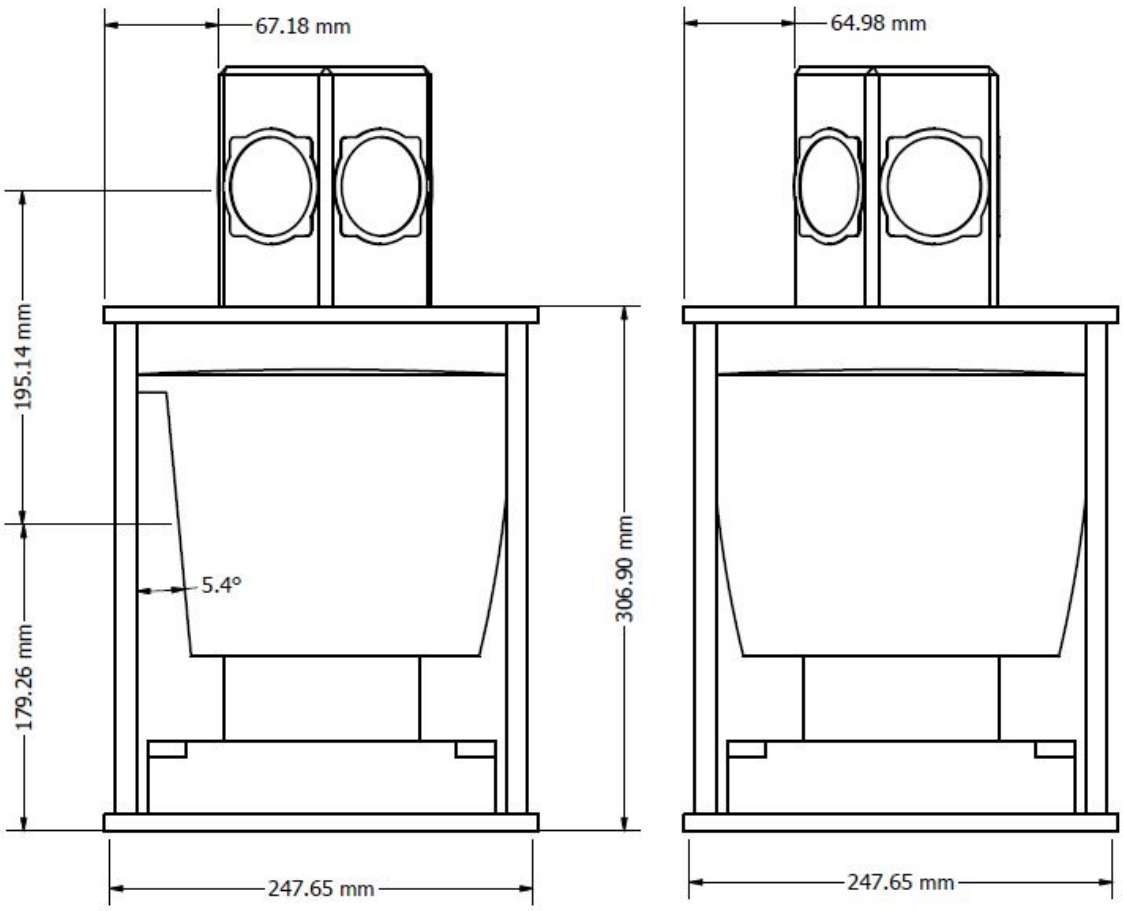

Figure 14: Translational offset between the LIDAR and the camera.

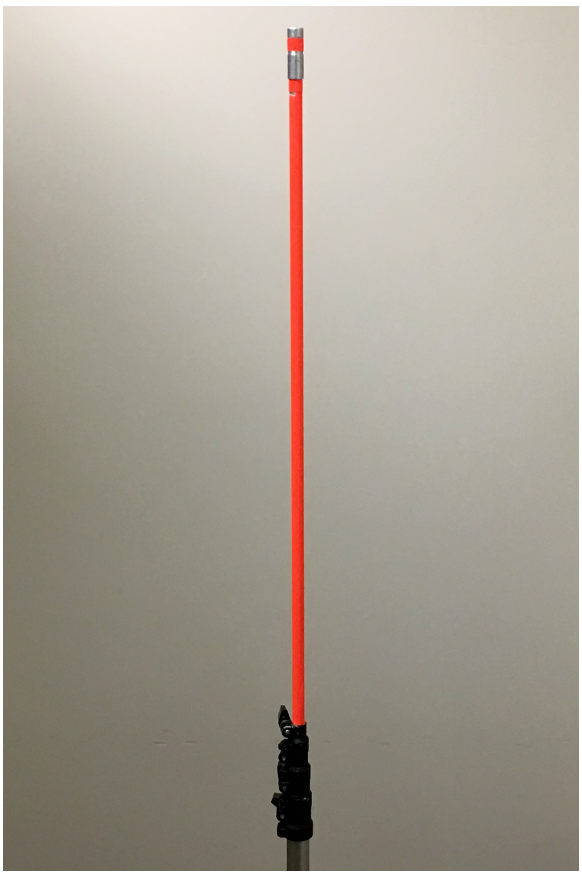

Figure 15: Calibration object used for finding relative rotation offsets.

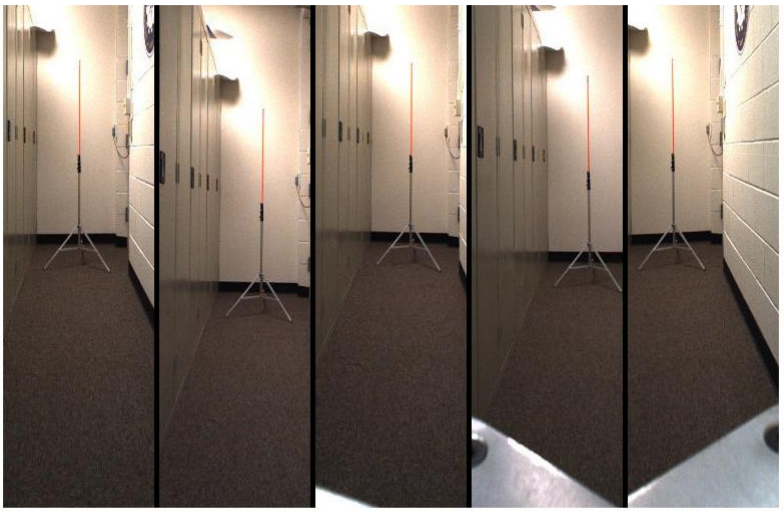

Figure 16: Calibration object in each camera sensor.

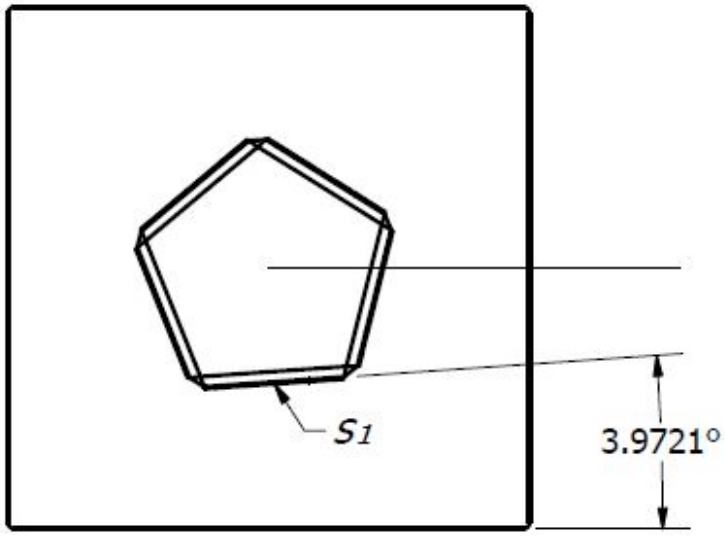

Figure 17: Rotational offset between the LIDAR frame and the s_1 frame.

Camera to Camera.

The intrinsic differences in cameras must be accounted for to properly determine the relative position of two or more cameras. The camera parameters for each camera are measured prior to each data collection using MATLAB's camera calibration tool, along with at least 20 images of the calibration checkerboard shown in Figure 18. By importing images at different skew angles and positions, as in Figure 19, MATLAB returns the calibration matrix in Equation 4, with the variables in Table 8.

$$\mathbf{C} = \begin{bmatrix} F * s_x & 0 & 0 \\ s & F * s_y & 0 \\ c_x & c_y & 1 \end{bmatrix} \tag{4}$$

Table 8: Matlab calibration matrix variable definitions.

Variable	Definition
cx	optical center x component
cy	optical center y component
s	skew parameter
F	focal length in mm
s_x	number of pixels per world unit x
s_y	number of pixels per world unit y

For instance, the calibration of the mobile phone camera, shown in Figure 20, resulted in this intrinsic matrix:

$$\mathbf{C} = \begin{bmatrix} 3486 & 0 & 0 \\ 3.3238 & 3501.7 & 0 \\ 1995.6 & 1528.2 & 1 \end{bmatrix} \tag{5}$$

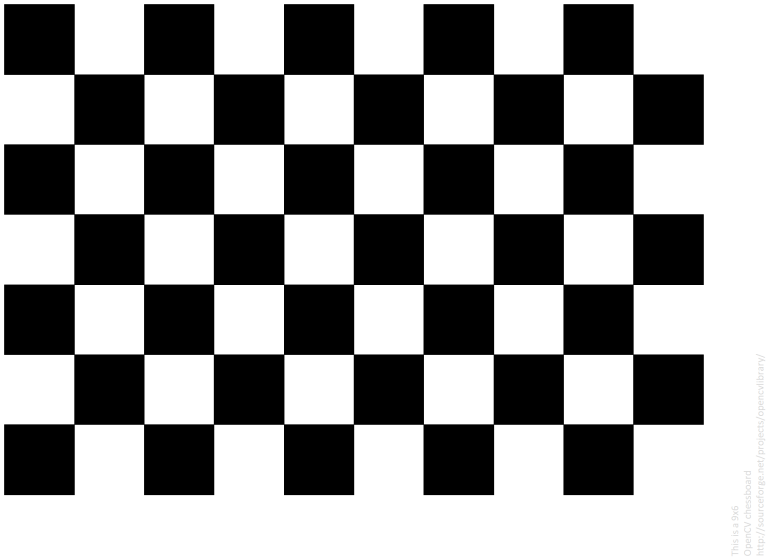

Figure 18: Checkerboard for determining intrinsic camera parameters.

This calibration method is performed before each test and for each camera.

Experiment Overview.

The following sections detail the experiments and procedures used to enable the camera pose estimation. The first experiment explores the efficacy and accuracy of using monocular images to create range and bearing measurements. The second experiment uses vision and LIDAR SLAM, along with the bearing and range data from Experiment 1, to generate geo-referenced images. The third experiment creates a collection of test images from different cameras. The final experiment uses the findings from the first 3 experiments to determine the position and orientation of the mobile cameras.

3.6 Experiment 1 - Calculate Range Data from an Image

The objective of the experiment was to demonstrate that range can be calculated to a fair degree of accuracy based only on the relationship between the number of

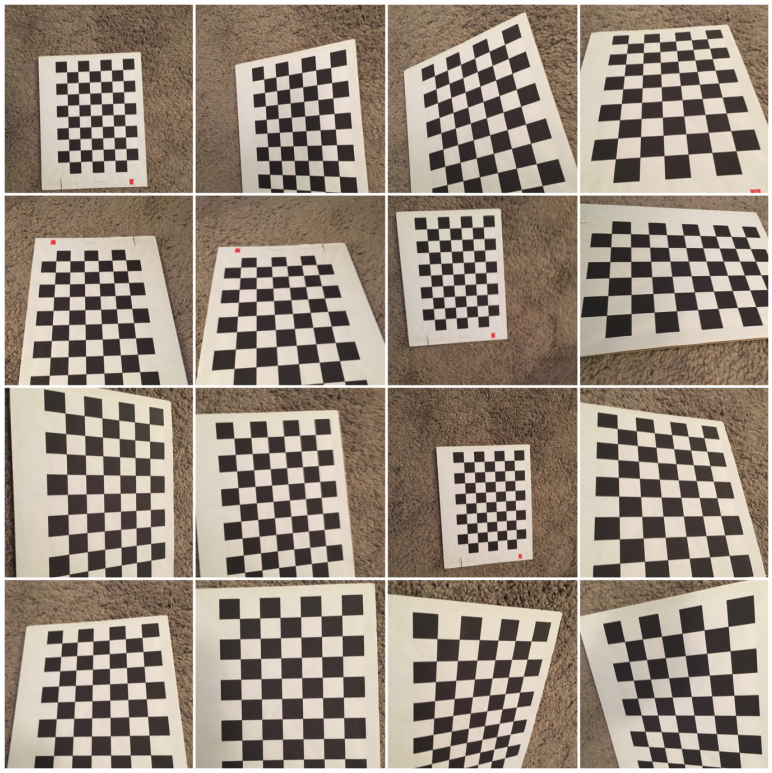

Figure 19: Subset of calibration images.

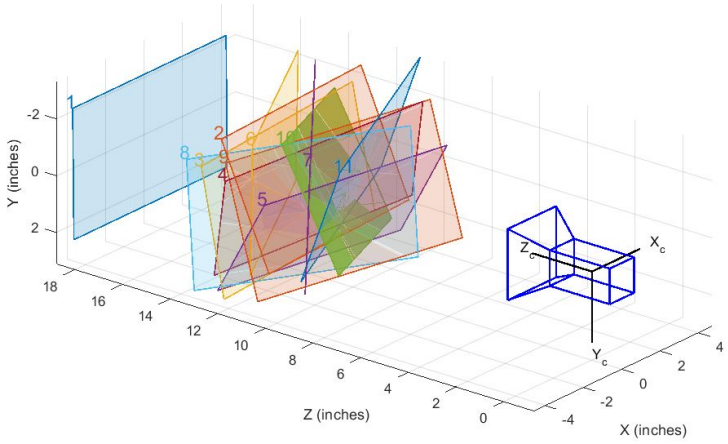

Figure 20: Camera calibration.

pixels of a feature with a known feature size, and the feature's distance from the sensor.

SLAM uses landmark measurements to improve the mapping solution. Since the spherical camera is already capturing data along with the LIDAR, it follows that images should be used to measure range and bearings to landmarks. While a single camera can easily tell the bearing to the landmark, that is insufficient for the SLAM algorithm. By using features of a known size, a range estimate can be calculated from a single camera. This experiment is designed to relate the distance of the camera to image features of known sizes, in this case ArUco markers.

The objective of the experiment is to demonstrate that range can be calculated based on the relationship between feature size in pixels and feature distance. Images of the ArUco marker shown in Figure 21 are taken at a series of known distances at the same height as the camera, as shown in Figure 22. After being rectified, the images are processed to identify the marker in each image and calculate the marker corner locations. A short MATLAB script takes the corner locations and calculates the marker area. The MATLAB Curve Fitting Toolbox is used to process the data and create a model of the pixel-distance relationship. Root mean squared error is used to evaluate the goodness-of-fit for each curve fitting model, through which the rational model in Equation 6 provides the least error.

$$y = \frac{p_1 x + p_2}{x^2 + q_1 x + q_2} \tag{6}$$

3.7 Experiment 2 - Generate a Set of Geo-Referenced Images

The first step for this experiment is to set up ArUco markers at known locations so that s_1 of the spherical camera will pass directly in front of them. The bearing and

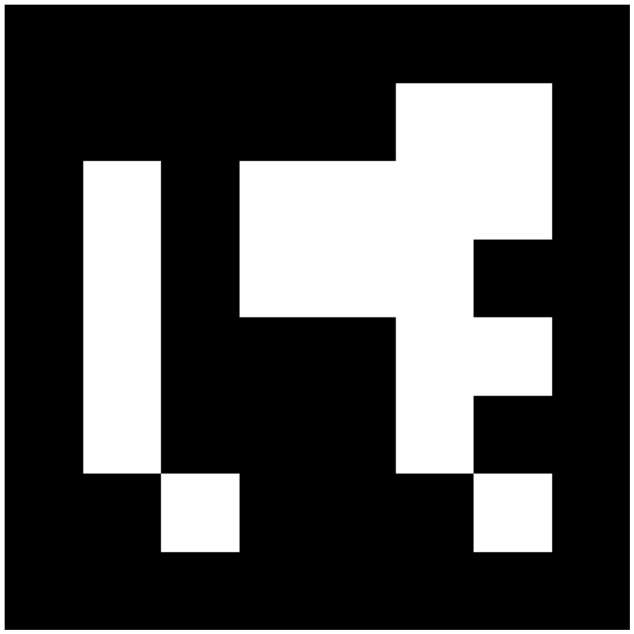

Figure 21: ArUco marker number 1, 6x6.

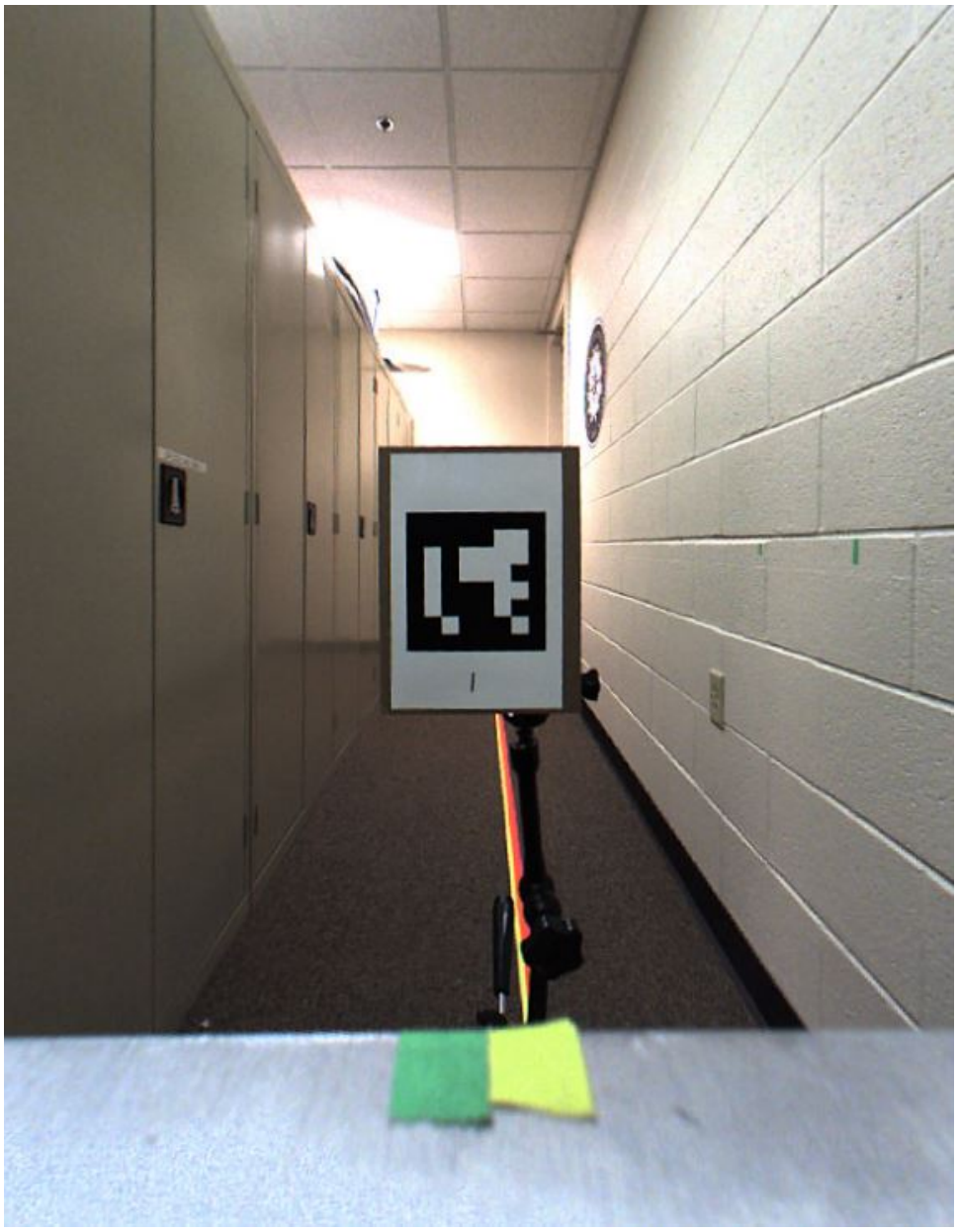

Figure 22: Test setup for capturing images of ArUco markers at different distances.

range factors to these landmarks will improve the end result of the factor graph optimization method. To simplify the bearing and elevation measurements, each feature is placed at a height of 0.934 m, the height of the camera, as shown in Figure 23. Additionally, each range measurement is only taken when the camera passes directly in front of the marker, limiting the bearing measurements to a 5° window. This method of sorting markers and identifying the most suitable measurements to use as bearing and range factors is listed in detail in Appendix A.

Once the markers are placed throughout the environment, their position must be recorded. First, the marker position in the environment frame is measured using a surveying tape. Next, the measurements are converted from the tape, with graduations in $\frac{1}{10}$ of a foot, into meters to correspond with the other distance measurements throughout the thesis. Finally, the marker positions are added to the environment map, as shown in Figure 24.

After the measurements are made, the LIDAR and camera are activated and the data capture programs begin recording. When the individual programs are operating correctly, several correlation movements are performed by moving the platform slightly. This makes it possible to temporally align the camera and LIDAR to within one frame. Then the platform is moved around the environment, moving camera s_1 past the landmark features. This loop is traversed three times, stopping after a final pass of landmark 1 which acts as a loop closure for the final pass. Again, several movements are made with the platform to account for drift between the sensors. The data from both sensors is synchronized as described in subsection 2.9. The rectified images from s_1 are passed through an ArUco marker detection algorithm which returns the frame number, marker ID, and marker corner location in pixels. The algorithm detects many different images for each pass. However, only the picture when the marker is closest to the image center is used. This recreates the conditions

of Experiment 3.6, where the marker was centered in the image and only the range was changed, maximizing the range measurement accuracy.

The following process is used to find the optimal frame for each landmark pass, while still creating landmark measurements for successive passes. First, the center of the marker in each image is calculated using the corner locations, and the absolute distance from image center to marker center is passed into the function that determines the key frames for each marker ID. Then the pixel area of each marker in a key frame is passed into the distance estimation model from subsection 3.6 to calculate the estimated range. Finally, the relevant information for creating bearing and range factors is compiled and stored.

The original LIDAR data is processed in several steps. First, running the proprietary calibration program corrects for distortion and other factors that are inherent in the data. Then the points for each LIDAR frame are exported into MATLAB, where the frames are stored as point cloud objects. Performing ICP-based point cloud registration on consecutive frames determines a relative change in position and orientation which is used to generate odometry factors.

GTSAM is used to process all of the information into an optimal trajectory estimate. The platform starting location and heading, along with all of the landmark positions, are added to the factor graph as constraints. These constraints have some prior noise model associated with them to allow for some adjustment during optimization. Next the odometry factors are added, consisting of the change in rotation and translation from ICP, a pose ID, and a noise model of the measurements. Then the bearing-range factors from the ArUco marker images are added, using the odometry pose that is closest to the image capture time. The noise models for the odometry and bearing-range factors are tunable parameters. In this case, the noise models are adjusted so that the position covariance for each pose and landmark encompasses the

true positions.

After GTSAM compiles the noise models and factors into a factor graph, it is optimized using a non-linear Levenberg-Marquardt optimization. Once the optimal pose locations and bearings are found, the extracted measurements are linked to each camera frame. Since the camera frame is not directly synchronized with odometry poses, the measurement is an interpolation of the two nearest pose measurements.

Figure 23: ArUco marker placed at camera height.

3.8 Experiment 3 - Collect Imagery in the Same Environment

To prove the concepts put forward in Section 1.2, there must be images to test. Three different cameras were used to take images to ensure that different cameras can still be localized. The image sets from the three cameras are collected simultaneously. First, all three cameras are calibrated using the method outlined in Subsection 3.5.

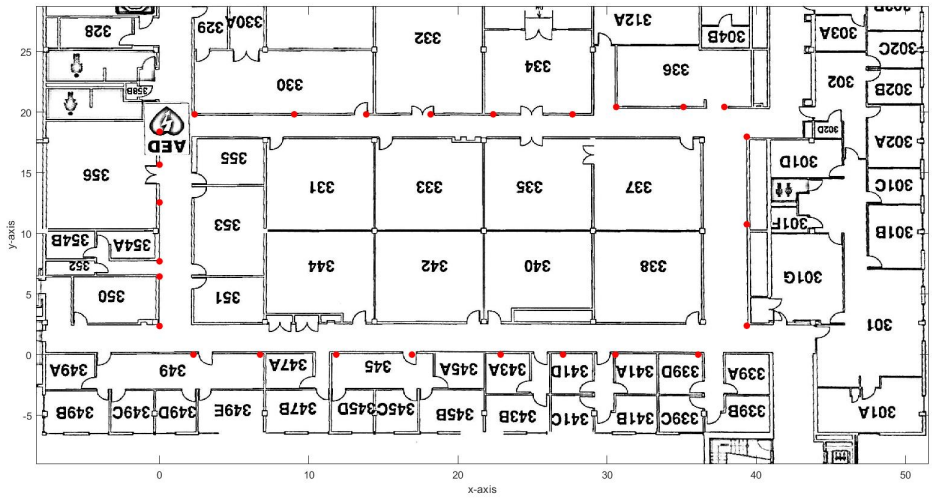

Figure 24: Test environment, with landmark feature locations.

3.9 Experiment 4 - Extract Relative Pose from Matched Images

The first part for this experiment is to build a database of image feature descriptors from the geo-referenced camera data. For this, the rectified images from s_0 to s_5 are passed through an OpenCV image processing algorithm to find all of the SIFT features. The SIFT feature descriptors for the different sensors are then compiled into a single list. This is repeated for each geo-referenced image frame. The feature lists, organized by image frame, are stored in a database.

Test images are selected from the mobile camera data, and processed using the same image feature algorithms. Then, the image features are compared to the feature database using RANSAC to determine if the mobile image has some overlap with a geo-referenced image. To reduce the possibility of having a false match, the three most likely matches are displayed to the user who can select a match that has two overlapping images.

After a matching image set is found, specific image features in both images are found and ranked based on likelihood. The essential matrix is calculated for both

images using the principles in Section 2.8. Next, the essential matrices are used to recover the relative **R** and **T** matrices, which describe the orientation of the mobile camera relative to spherical camera and the location of the mobile camera relative to spherical camera, respectively.

The pose recovery algorithm returns a translation in the form of a unit vector. To determine the actual translation, additional steps are required. First, the pose recovery calculation is repeated for the poses 15 frames before and after the selected pose. Next, the intersection of the three translation vectors is calculated. Each vector v has an origin point a and the intersection point c is the closest point of intersection for all three vectors. The distance of this point to the line is:

$$H = \frac{\|(c-a) \times d\|}{\|d\|} \tag{7}$$

With the identity $(a \times b) \cdot (a \times b) = \|a\|^2\|b\|^2 - (a \cdot b)^2$ the distance of this point to the line becomes:

$$H^2 = \frac{\|c-a\|^2\|d\|^2 - \|(c-a) \cdot d\|^2}{\|d\|^2} \tag{8}$$

$$H^2 = \|c-a\|^2 - \frac{\|(c-a) \cdot d\|^2}{\|d\|^2} \tag{9}$$

To find a c such that the sum is minimized, the derivative with regard to c should be zero:

$$0 = \sum_{i=1}^{3} c - a_{(i)} - d_{(i)} \frac{(c-a_{(i)}) \cdot d_{(i)}}{\|d_{(i)}\|^2} \tag{10}$$

The final stage is to convert the mobile camera pose from the camera coordinate frame back to the local environment frame using the relative rotations and translations described in Section 3.4. The first step is to calculate the Direction Cosine Matrix

(DCM) $\mathbf{C}$ from the mobile camera to the environment using Equation 11. $\mathbf{R_1}$ is the DCM from the spherical camera to the environment, and $\mathbf{R_2}$ is the DCM from the mobile camera to the corresponding spherical camera sensor. The camera position is calculated using Equation 12, where $\mathbf{V}$ is the position of the mobile camera in the environment frame, and $\mathbf{V_1}$ is the position of the spherical camera in the environment frame.

$$\mathbf{C} = \mathbf{R_1}\mathbf{R_2} \tag{11}$$

$$\mathbf{V} = \mathbf{V_1} + \mathbf{CT} \tag{12}$$

IV. Results and Analysis

This Chapter reviews and analyzes the results from the four experiments from Chapter 3. First is the efficacy and accuracy of calculating range data from monocular image data. Second is generating geo-referenced images using vision-based and LIDAR-based SLAM using the range data from Experiment 1. The third experiment is the collection of mobile image sets using different cameras. The final experiment combines the images from Experiment 3 with the geo-referenced images from Experiment 1 and Experiment 2 to determine the position and orientation of the non-localized sensor.

4.1 Experiment 1 - Calculate Range Data from an Image

Initial Test.

The objective of the experiment was to demonstrate that range can be calculated to a fair degree of accuracy based only on the relationship between the number of pixels of a feature with a known feature size, and the feature's distance from the sensor. The marker area and distance from each image are tabulated in Table 9. When the marker distance is plotted as a function of pixel area, as in Figure 25, a correlation becomes apparent.

The MATLAB Curve Fitting Toolbox has many different models that can be used to describe variable relationships. When each curve model was tuned to fit the training data, the Exponential 2, Power, and Rational models all appeared to fit the data correctly. To quantitatively determine the optimal model, the RMSE was calculated. The equations for each fit model, as well as the RMSE when using that model, are listed in Table 10. This approach to calculate goodness of fit provided a definitive answer: the 1-2 rational fit model had the lowest RMSE error of 0.0200.

Table 9: Marker distance and area used to create a distance estimation model for a 82mm×82mm marker.

Marker Distance (m)	Marker Area (pixels)
0.4159	8145.5
0.5683	5587.5
0.6445	4128
0.7207	3136
0.7969	2450.5
0.8731	1958
0.9493	1601
1.0255	1350.5
1.1017	1139
1.1779	977
1.2541	841
1.3303	729
1.4065	663
1.4827	564.5
1.5589	517.5
1.6351	462
1.7113	410
1.7875	370.5
1.8637	351.5
1.9399	315
2.0161	289
2.0923	256
2.1685	256
2.2447	232.5
2.3209	217.5
2.3971	203
2.4733	182
2.7019	144
2.7781	144

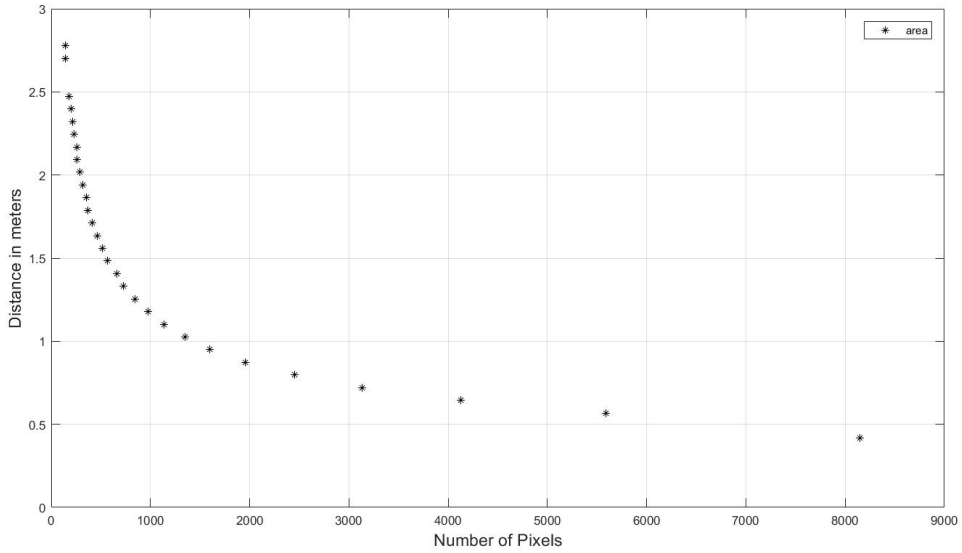

Figure 25: Training data used to model the area-distance relationship for an 82mm×82mm marker.

This was less than half of the error produced by using the Exponential model, and slightly better than the Power model. Figure 26 shows the tuned fit model overlaid on the training data set.

The accuracy of using a model with these tuning parameters was unknown. Additional tests using images taken with known marker distances were required to determine the accuracy of the method. More images of the marker were taken using two different distance intervals. The true distances with respect to pixel area from the new images were compared to the estimated distance calculated by using the model. As hypothesized, the new data follows the same relationship as the training data and the rational fit model. These three data sets are shown in Figure 27. The measurement error, shown in Figure 28, is the estimated distance from the model subtracted from the measured distance. All of the distances calculated when using this method were within 0.055 m of the measured value.

The error from using this method has to be described to account for it when

Table 10: Goodness of fit for different curve models based on RMSE of the 82mm×82mm marker data.

Fit Model	Fit Equation	RMSE
Polynomial 1	$y = p_1x + p_2$	0.4518
Polynomial 2	$y = p_1x^2 + p_2x + p_3$	0.3235
Polynomial 3	$y = p_1x^3 + p_2x^2 + \ldots + p_4$	0.2232
Weibull	$y = abx^{b-1} \times e^{-ax^b}$	1.7612
Exponential 1	$y = ae^{bx}$	0.2661
Exponential 2	$y = ae^{bx} + ce^{dx}$	0.0427
Fourier 1	$y = a_0 + a_1cos(xp) + b_1sin(xp)$	0.4481
Gaussian 1	$y = a_1e^{-(\frac{x-b_1}{c_1})^2}$	N/A
Power 1	$y = ax^b$	0.0226
Power 2	$y = ax^b + c$	0.0230
rat12	$y = \frac{p_1*x+p_2}{x^2+q_1*x+q_2}$	0.0200

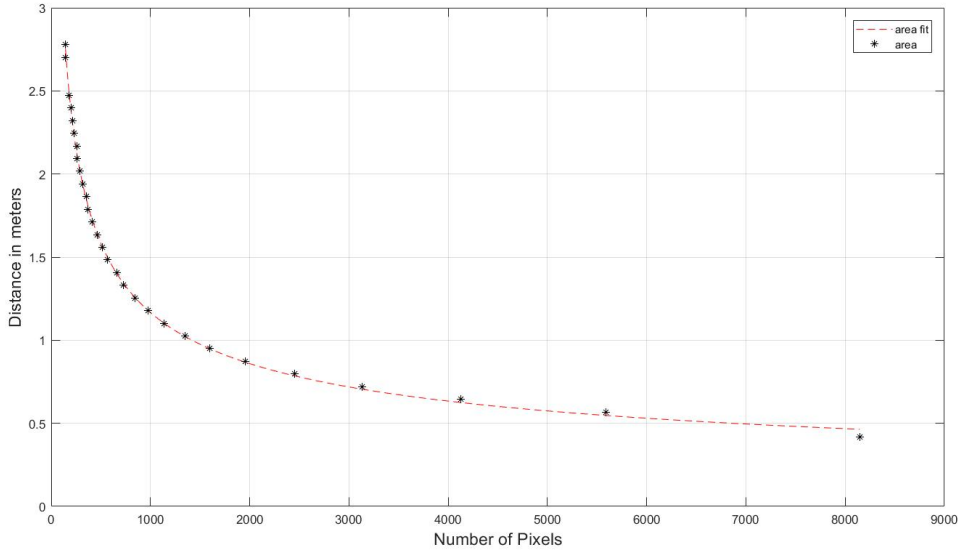

Figure 26: Curve fitting results with rational model for an 82mm×82mm marker.

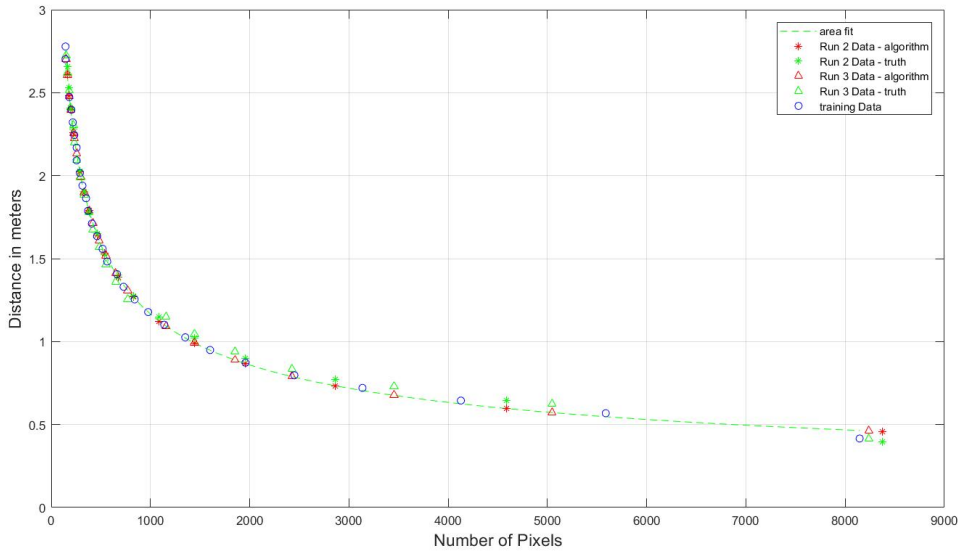

Figure 27: Marker distance as a function of pixel area for training data and verification data using an 82mm×82mm marker.

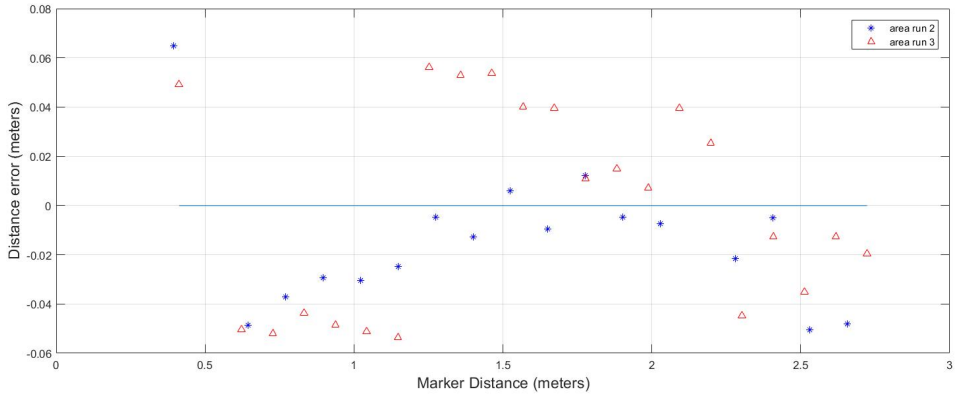

Figure 28: Distance error as a function of marker distance for 82mm×82mm marker.

using graph-based SLAM. The Gaussian noise model that describes this error is also determined using MATLAB. First, a histogram of the distance error is generated. Then a Gaussian probability density function (PDF) is fit to the histogram. Figure 29 shows the histogram and resulting PDF for the distance estimation function. The PDF is described in Equation 13, with the tunable parameters of μ and σ being the mean and standard deviation in meters, respectively. Equation 14 describes the specific PDF for the test.

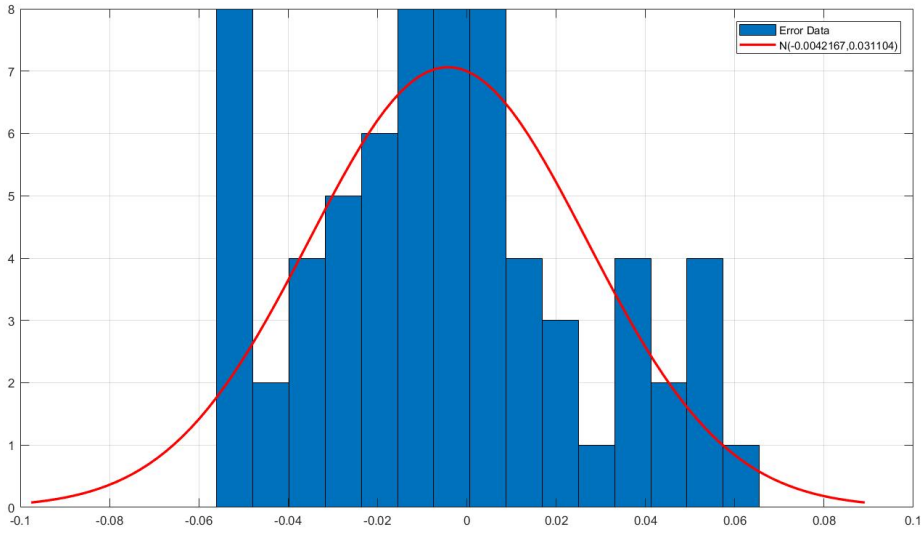

Figure 29: Histogram and corresponding PDF when using the 82mm×82mm marker estimation function.

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{13}$$

$$\mathcal{N}(-0.0042167, 0.031104) \tag{14}$$

Verification Test.

The entire experiment was repeated a second time to ensure that the principles were still valid when using markers with a different feature size. To reduce feature size without changing bit size, the 6x6 ArUco marker was substituted for a 4x4 marker with the same bit dimensions. This created a 55mm×55mm marker. The steps of collecting, plotting, fitting, and verifying the data are all the same as the initial test.

The data collection resulted in the distances and areas in Table 11. When graphed, as in Figure 30, the shape of the distribution remained the same as when using the larger marker. As expected, the marker distance was closer for a given area. This resulted in different values for the curve fitting models, along with the different RMSE values shown in Table 12.

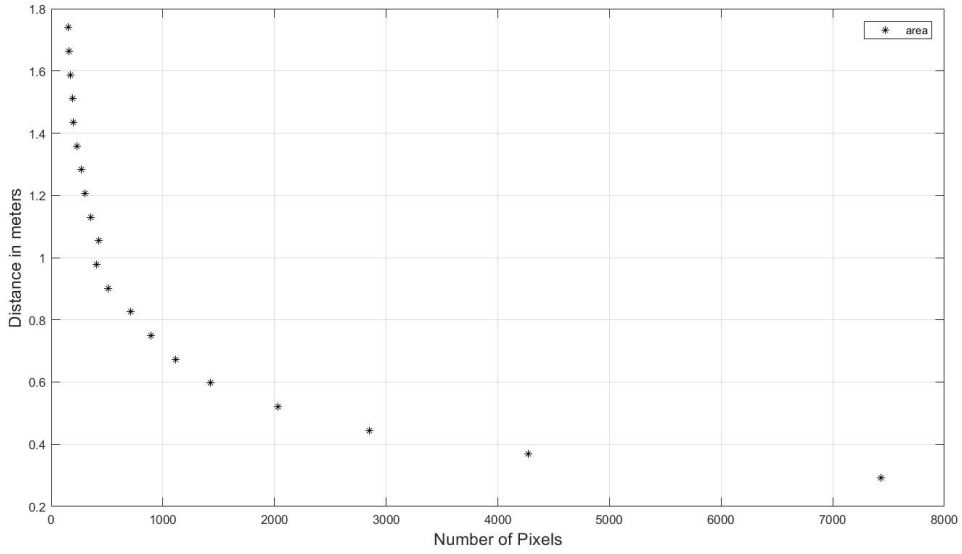

Figure 30: Training data used to model the area-distance relationship for an 55mm×55mm marker.

As with the larger marker the Rational 1-2 model had the lowest RMSE, with the Power model and Exponential model also fitting the data well but with slightly more

Table 11: Marker distance and area used to create a distance estimation model for a 55mm×55mm marker.

Marker Distance (m)	Marker Area (pixels)
0.2921	7432
0.3683	4275.5
0.4445	2849.5
0.5207	2034
0.5969	1427
0.6731	1116
0.7493	895.5
0.8255	708.5
0.9017	509
0.9779	409
1.0541	424
1.1303	356.5
1.2065	305
1.2827	267
1.3589	229
1.4351	200
1.5113	190.5
1.5875	170
1.6637	157
1.7399	153

Table 12: Goodness of fit for different curve models based on RMSE of the 55mm×55mm marker data.

Fit Model	Fit Equation	RMSE
Polynomial 1	$y = p_1x + p_2$	0.3124
Polynomial 2	$y = p_1x^2 + p_2x + p_3$	0.2197
Polynomial 3	$y = p_1x^3 + p_2x^2 + \ldots + p_4$	0.1547
Weibull	$y = abx^{b-1} \times e^{-ax^b}$	1.1656
Exponential 1	$y = ae^{bx}$	0.1808
Exponential 2	$y = ae^{bx} + ce^{dx}$	0.0425
Fourier 1	$y = a_0 + a_1cos(xp) + b_1sin(xp)$	0.2706
Gaussian 1	$y = a_1e^{-(\frac{x-b_1}{c_1})^2}$	N/A
Power 1	$y = ax^b$	0.0317
Power 2	$y = ax^b + c$	0.0287
rat12	$y = \frac{p_1*x+p_2}{x^2+q_1*x+q_2}$	0.0274

error. The Rational model had an error of 0.0274. Figure 31 shows the fit model overlaid on the training data set for the smaller marker.

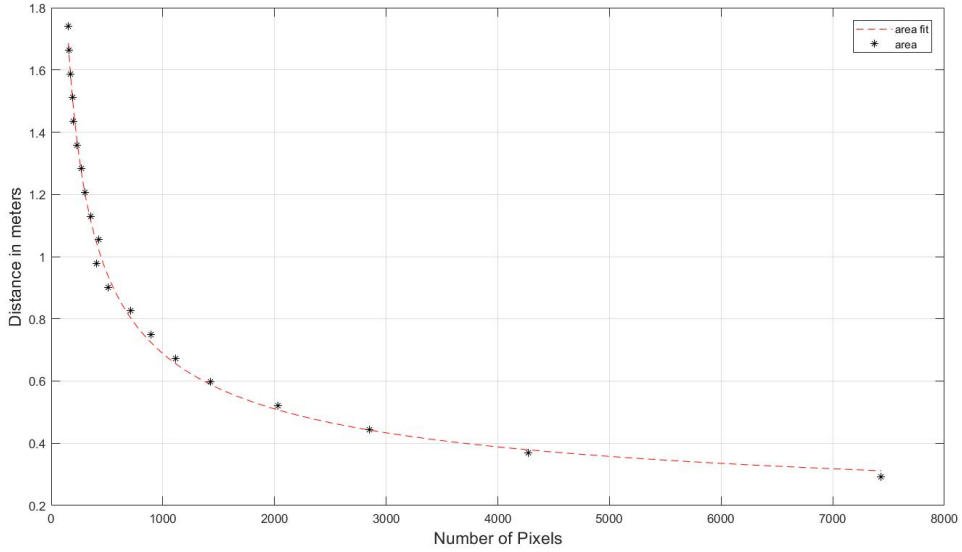

Figure 31: Curve fitting results with rational model for an 55mm×55mm marker.

Another two data sets with the 4x4 marker were collected to validate the model. As with the larger marker, the actual and estimated distances are plotted in Figure 32 with an overlay of the Rational model. The error, shown in Figure 33, is calculated as before: the distance estimate from the model is subtracted from the measured distance.

The Gaussian noise model that describes this error is also determined using MATLAB. First, a histogram of the distance error is generated. Then a Gaussian PDF is fit to the histogram. Figure 34 shows the histogram and resulting PDF for the distance estimation function. The PDF is described in Equation 15, with the specific characteristics in Equation 16. Table 13 shows the noise model parameters for both marker tests.

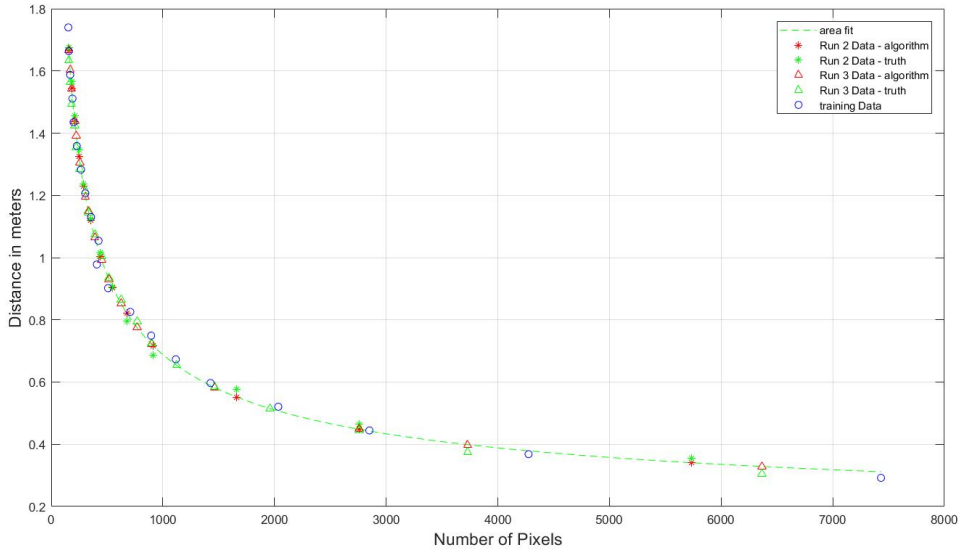

Figure 32: Marker distance as a function of pixel area for training data and verification data using an 55mm×55mm marker.

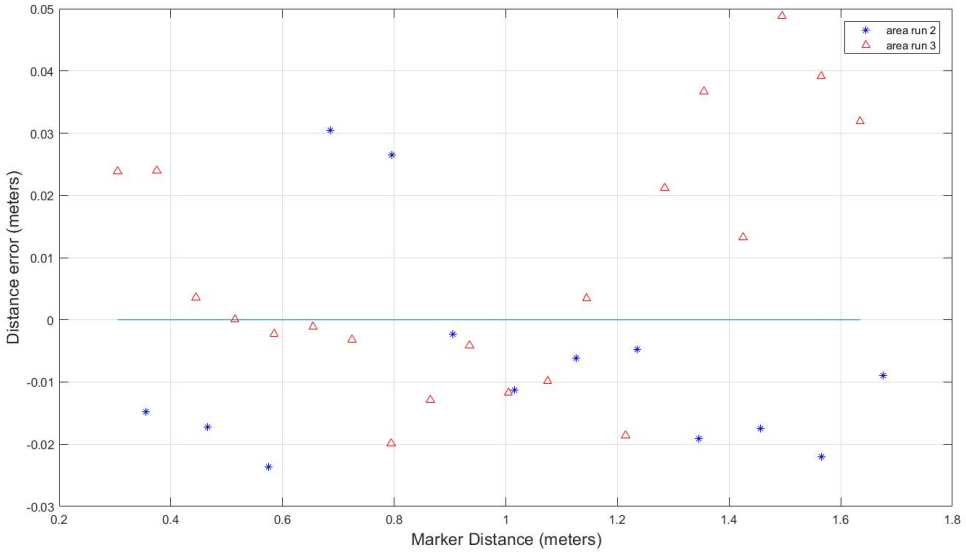

Figure 33: Distance error as a function of marker distance for 55mm×55mm marker.

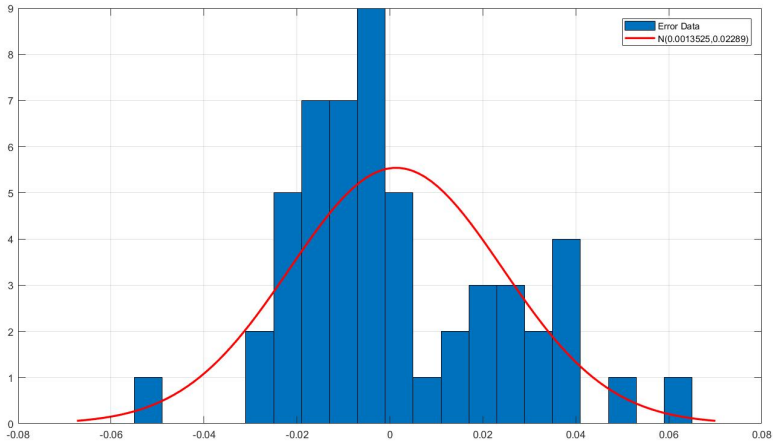

Figure 34: Histogram and corresponding PDF when using the 55mm×55mm marker estimation function.

$$\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{15}$$

$$\mathcal{N}(0.0013525, 0.02289) \tag{16}$$

Table 13: Noise models for both marker sizes.

Marker Size	μ	σ
82mm	-0.0042167	0.031104
55mm	0.0013525	0.02289

4.2 Experiment 2 - Generate a Set of Geo-Referenced Images

This experiment combines the range estimation method from Experiment 1, ICP-based odometry from LIDAR data, and the factor graph optimization of GTSAM to generate position and orientation measurements for a set of images. Instead of the

single marker from Experiment 1, markers with IDs ranging from 1-24 were captured in the camera footage. Figure 35 shows how markers are placed, and how they appear in the panoramic camera.

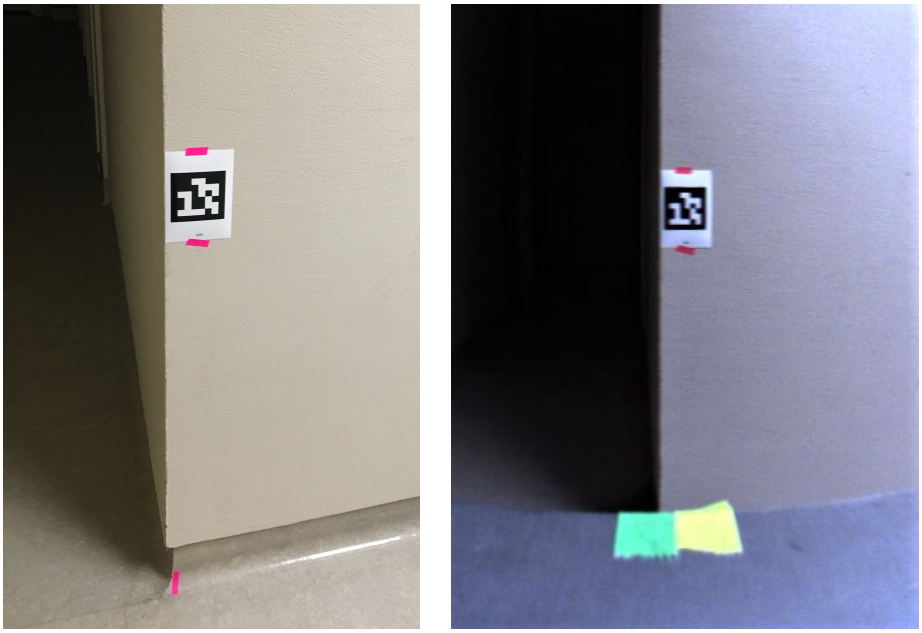

Figure 35: ArUco marker placement in the environment (left), and marker as viewed from s_1 (right).

Data Collection.

After the measurements are made, the LIDAR and camera are activated and the data capture programs begin recording. When the individual programs are operating correctly, several correlation movements are performed by moving the platform slightly. This makes it possible to temporally align the camera and LIDAR to within one frame. The running frequency of the camera was slightly higher than the LIDAR, although they were both initialized to record at 10 FPS. Following the assumptions laid out in Chapter 1, where the oscillator drift in the sensors is approximately linear, Figure 36 shows the timing error induced by this interpolation method.

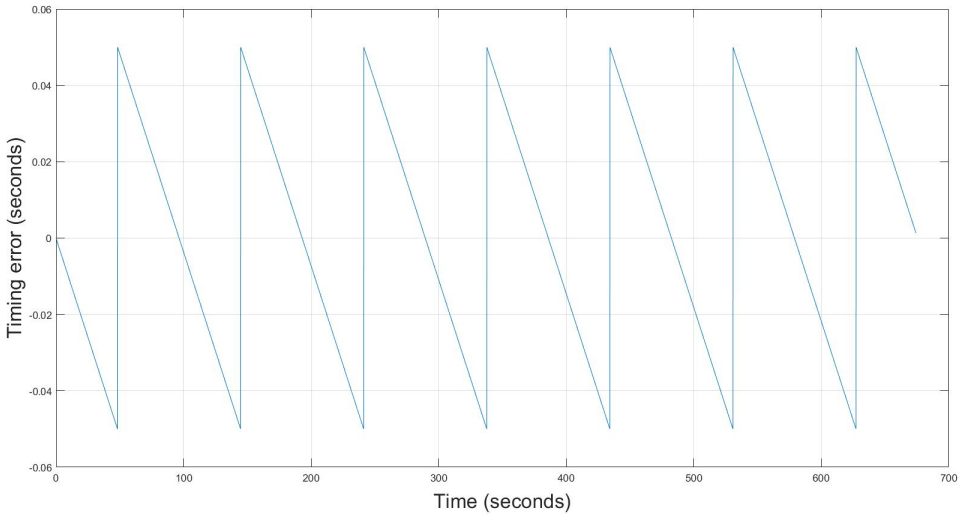

Figure 36: LIDAR and camera frame synchronization.

After the initial calibration movements, the platform is moved through the environment in a counter-clockwise path so each marker appears in s_1. After three passes the calibration movements are repeated before the sensors are safely disconnected.

Image Processing.

The rectified images from s_1 are passed through an ArUco marker detection algorithm which returns the frame number, marker ID, and marker corner location in pixels. The center of the marker in each image is calculated using the corner locations. Since the range estimation function was created using centered markers parallel with the image plane, a minimizing function is used to determine the optimal frames for each marker ID. This function is best demonstrated by Figure 37, where the marker center and image center are shown for three consecutive frames. In this example the center frame is chosen as a keyframe.

The results of this minimizing function are shown in Figure 38, where the distance from marker center to image center first decreases and then increases after

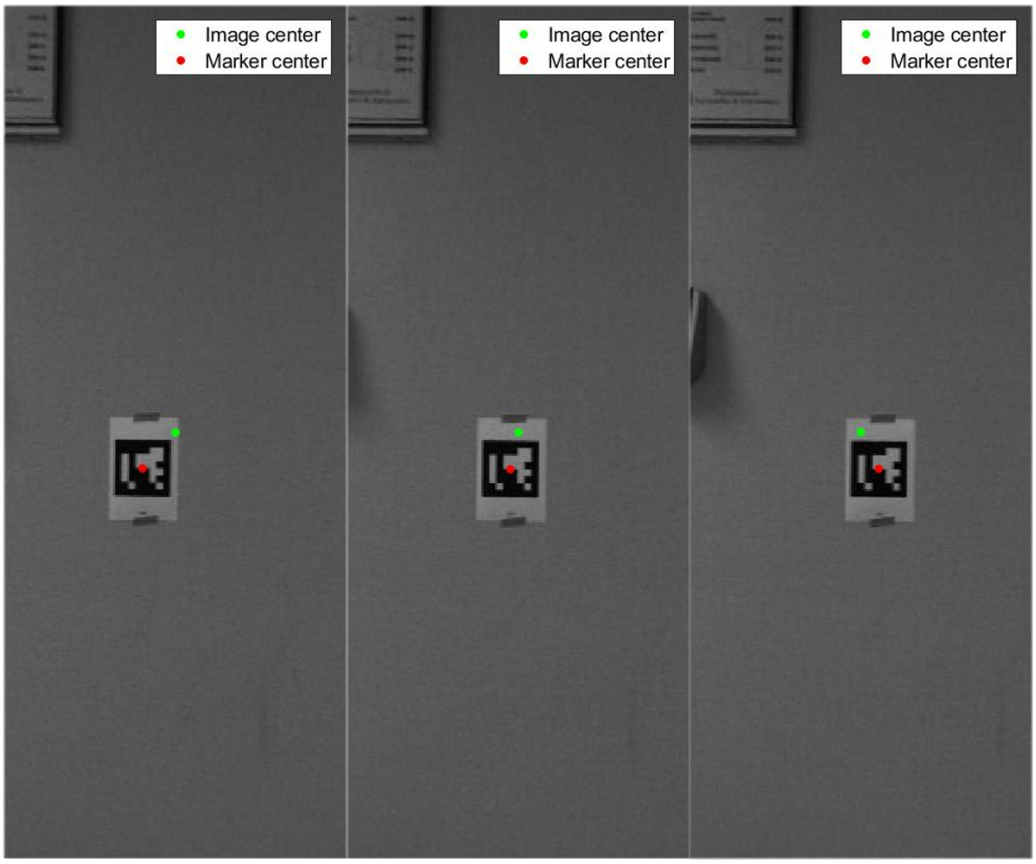

Figure 37: ArUco marker images, showing image center in green and marker center in red. The center image is selected as a keyframe.

some minimum value. Note the threshold in center-to-center distance that separates each marker pass. This threshold value, which is dependent on image resolution, ensures that new keyframes are created for new passes, without creating multiple measurements in a single pass.

The loop-closure effect of having one measurement per pass is better understood when looking at a single marker ID, as in Figure 39. Each time the marker is passed, a keyframe is created (in red). This unambiguous feature association results in strongly defined loop closures for the pose constraints in GTSAM.

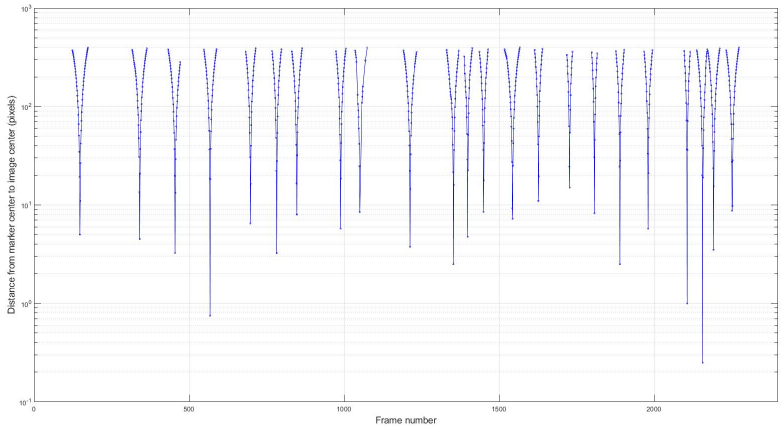

Figure 38: Output of the keyframe calculation function for all markers in the first pass.

The pixel area of each marker for each keyframe is passed into the distance estimation model from subsection 3.6 to calculate the estimated range. This range data is shown in Figure 40. The relevant information for creating bearing and range factors is then compiled and stored.

LIDAR Processing.

The original LIDAR data is processed in several steps. First, running the proprietary calibration program corrects for distortion and other factors that are inherent

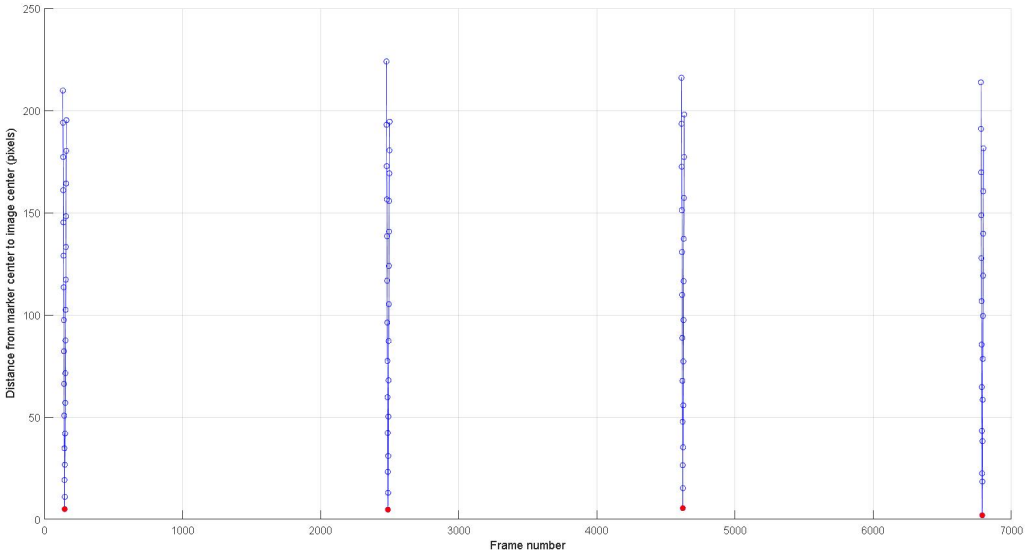

Figure 39: Output of the keyframe calculation function for marker 1.

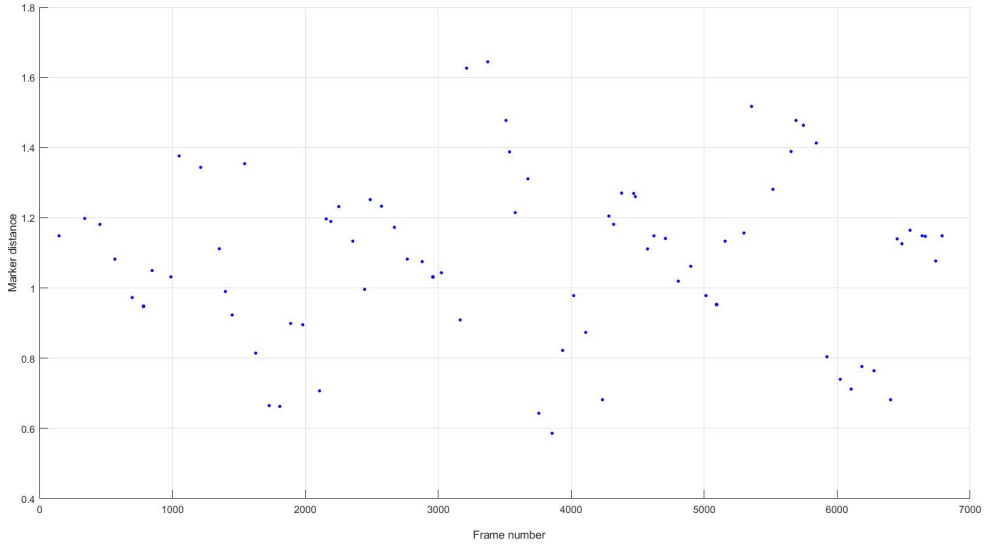

Figure 40: Estimated distance to each marker for each keyframe.

in the data. Then the points for each LIDAR frame are exported into MATLAB, where the frames are stored as point cloud objects. Performing ICP-based point cloud registration on consecutive framesdetermines a relative change in position and orientation which is used to generate odometry factors.

Optimization and Pose Extraction.

GTSAM is used to process all of the information from ICP and the keyframe function, creating an optimal trajectory estimate and covariance. The platform starting location and heading, along with all of the landmark positions, are added to the factorgraph as constraints. Since these positions are measured in the physical environment, they help the optimization function to converge. These constraints have some prior noise model associated with them to allow for some adjustment during optimization. The noise on the prior constraints is estimated at less than 12mm, since there is very little variance in physical measurements.

Once the prior constraints are added, each odometry factor from the ICP function is added. These factors include pose deltas, but they can be integrated without any additional factors to view the initial odometry path. This initial odometry is shown in Figure 41. Although it starts in the correct location with a proper heading, the accumulation of error becomes apparent. After roughly 1100 frames the position estimate is not even bounded by the hallway.

To account for the accumulating error in the odometry, the noise model must be altered. Since the trajectory is known, while the noise of ICP is unknown, the noise model is determined empirically. Starting with a very small noise value, the noise is slowly increased until the covariance of each pose encompasses the true position. Figure 42 shows the covariance ellipses and odometry poses after the noise model is properly adjusted.

Figure 41: Plot of platform odometry in GTSAM without covariance.

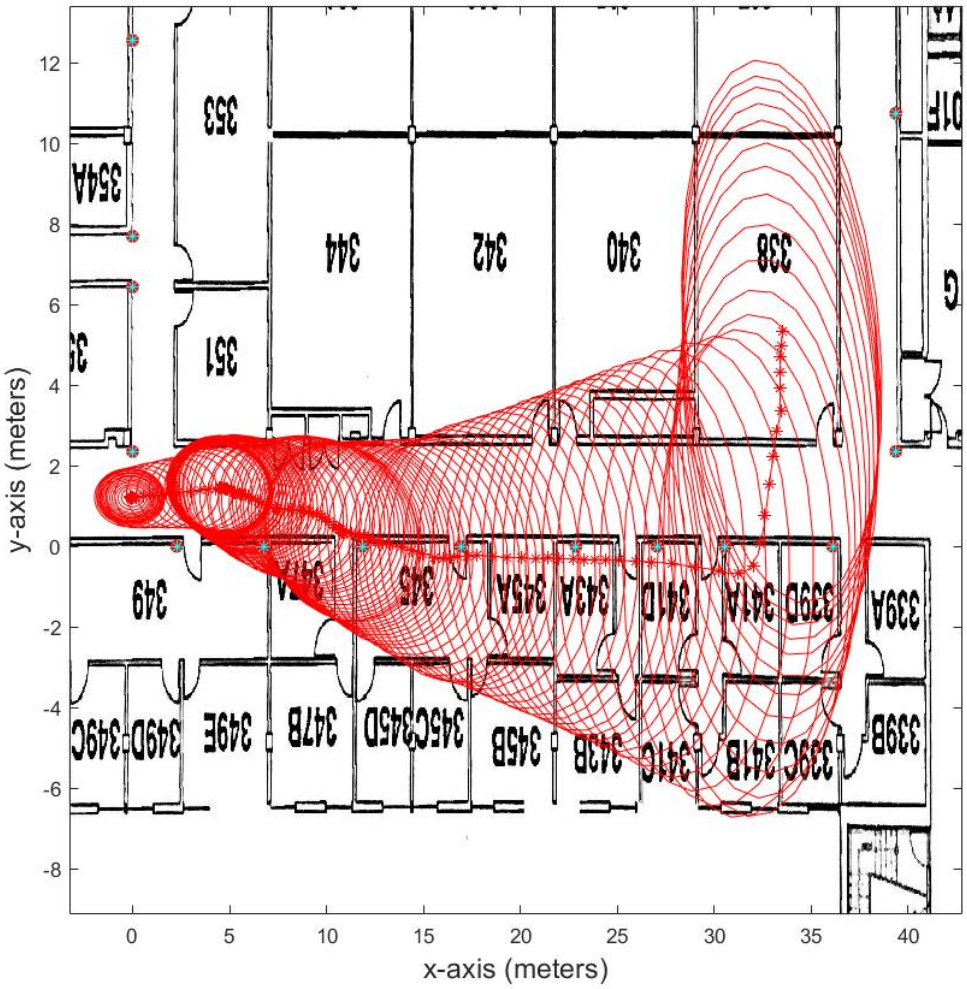

Figure 42: Plot of platform odometry in GTSAM with noise model properly tuned.

With properly tuned odometry, the landmarks and bearing-range factors can be added. Landmarks are initialized to the locations measured, with a small amount of noise to account for some measurement error during optimization. The initialization of landmarks is shown in Figure 43. Each bearing and range constraint from the keyframe function is attached to the pose of the LIDAR frame that is closest to the keyframe. The Gaussian noise model from Experiment 1 for the 6x6 ArUco marker is used as the noise model for all of the bearing-range factors.

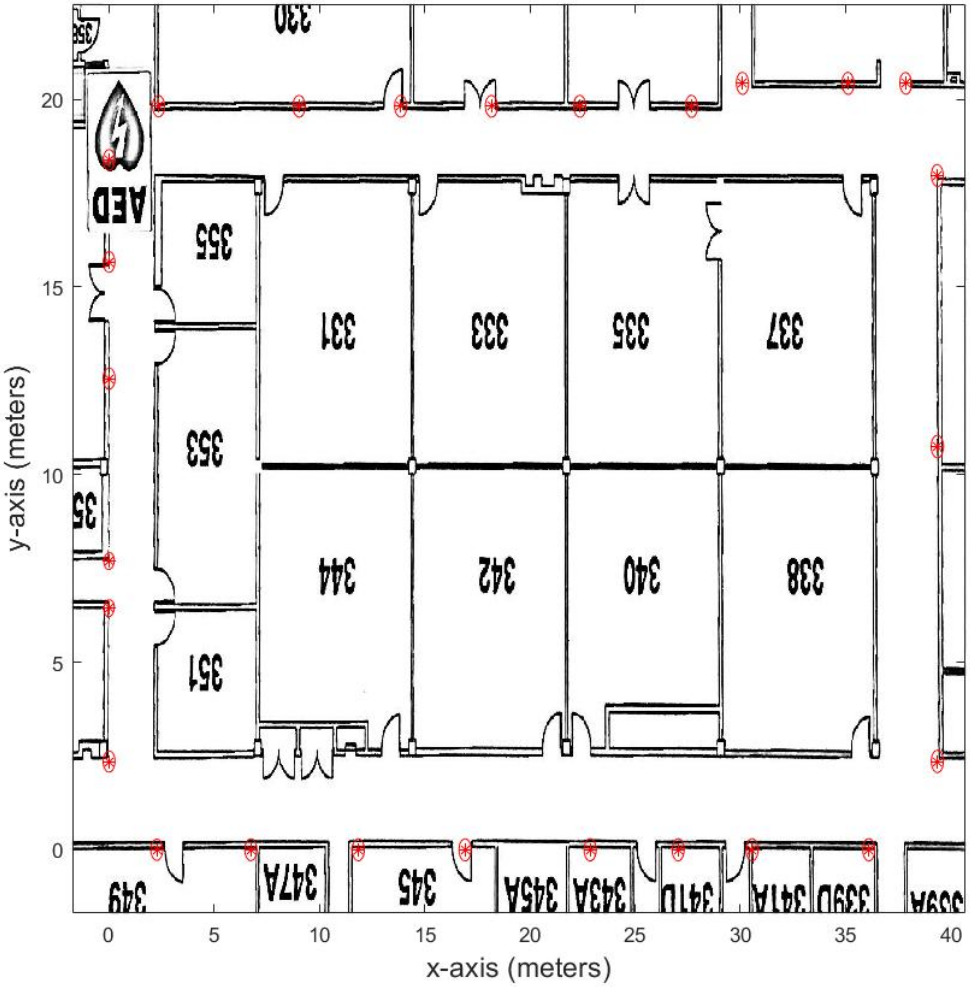

Figure 43: Initialization of landmark factors in GTSAM.

When all of the factors are added, GTSAM can find the optimal pose solution using a range of optimization algorithms. This research uses the Levenberg-Marquardt algorithm which works well for handling non-linear optimization. The optimized trajectory, shown in Figure 44, is much more accurate than the integrated odometry solution.

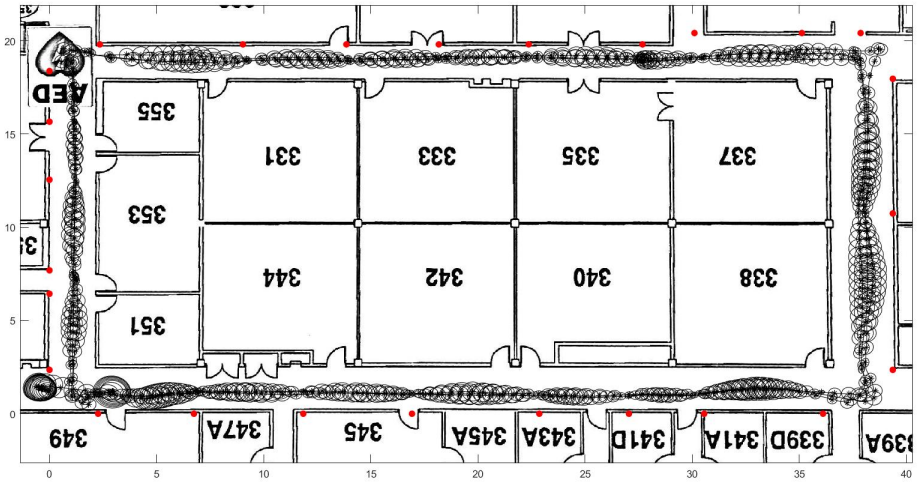

Figure 44: GTSAM factorgraph after Levenberg-Marquardt optimization.

Figure 45 shows the initial landmark location estimate and location covariance along with the optimized landmark location and covariance. Because of the loop-closure constraints detailed in Subsection 4.2, the odometry uncertainty no longer grows unchecked. Instead, the highest uncertainty occurs when the platform is equidistant from two landmarks. This phenomenon, shown in Figure 46 demonstrates the importance of having unambiguous landmarks.

Once the position and orientation for each pose have been translated into the environment frame, the information can be used to determine the location of the panoramic camera for each image. The pose for each image is calculated using the sensor synchronization data. The pose is determined by interpolating between the

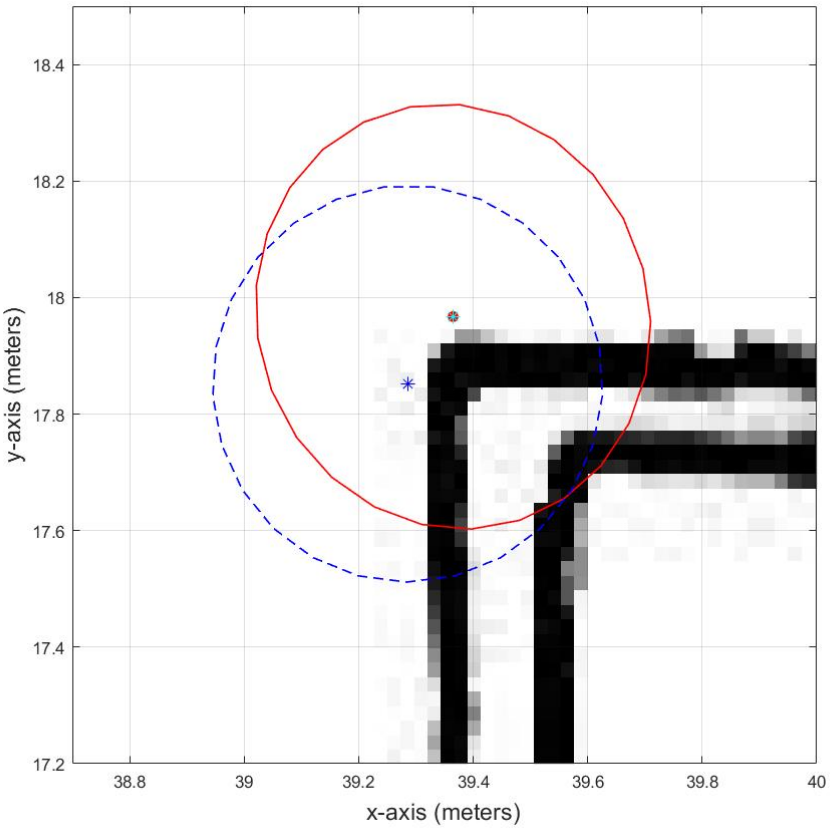

Figure 45: Landmark location and marginal before (solid line) and after optimization (dashed line).

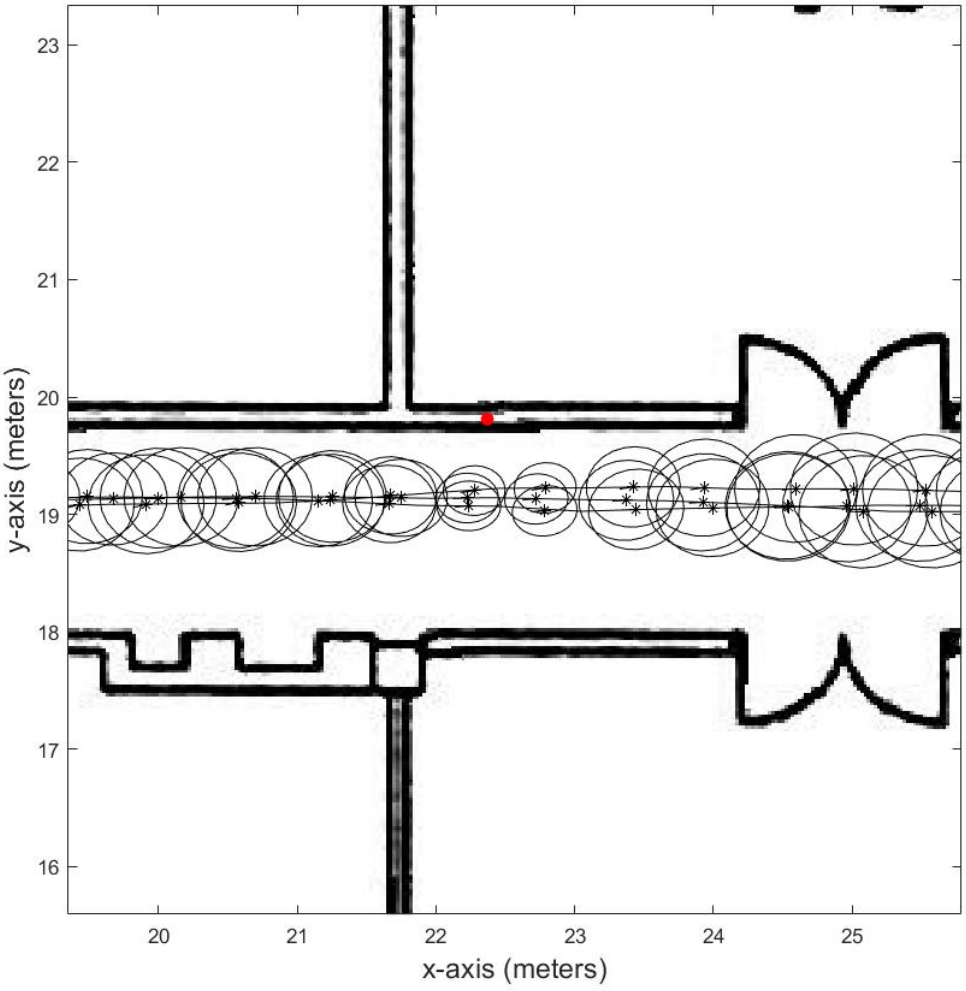

Figure 46: Pose uncertainty decreases when passing a landmark.

nearest two poses. Figure 47 shows an example image from s_1 and its corresponding position and orientation.

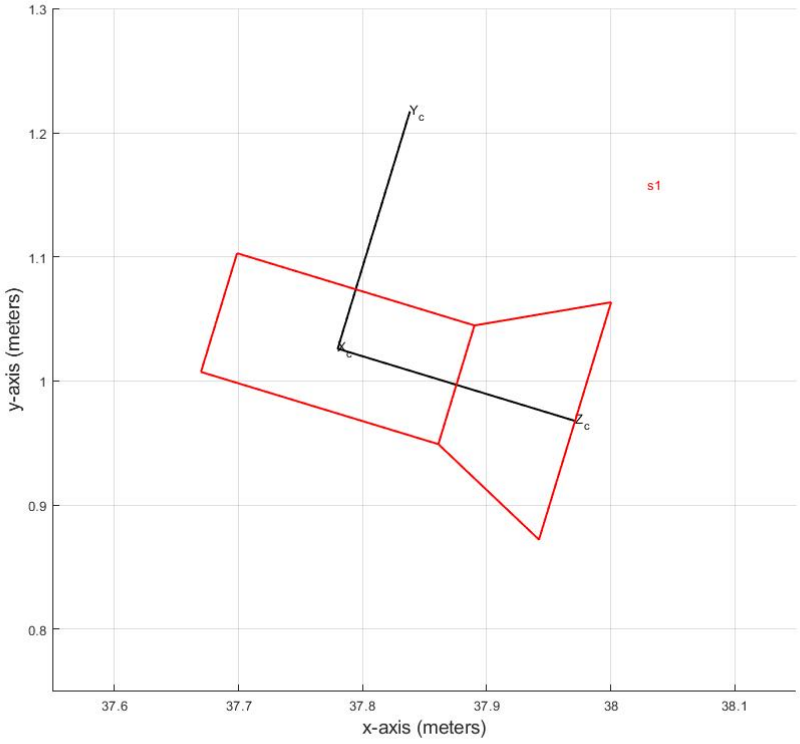

Figure 47: Position and orientation of s_1 for frame 3189.

4.3 Experiment 3 - Collect Imagery in the Same Environment

The mobile camera imagery is collected using the sensors described in Chapter 3. The path follows the same counter-clockwise trajectory as the platform in Section 4.2, as shown in Figure 48. Since the camera footage from each sensor was collected simultaneously, image frame interpolation can be performed to account for camera-to-camera timing. This correlation is possible using the same method as the LIDAR-

camera correlation in Section 4.2 since there are changes in motion at the beginning and end of the collection runs that appear in all three sensors.

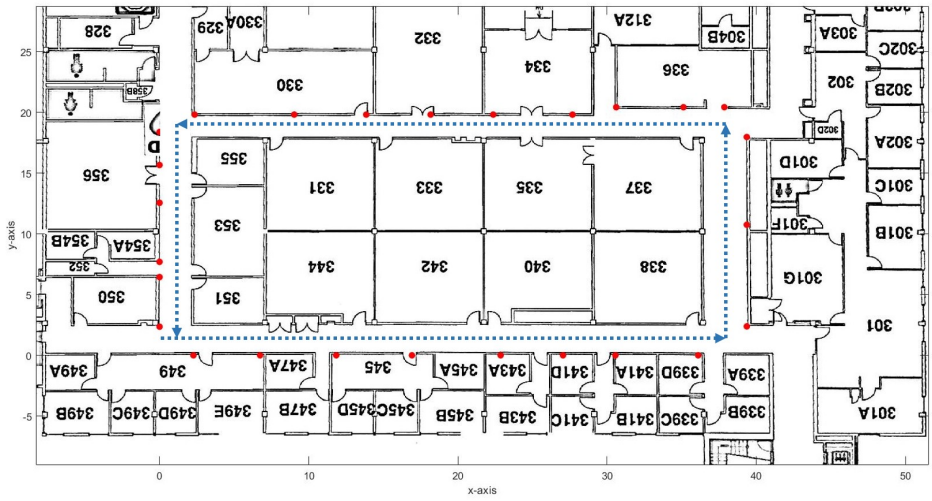

Figure 48: General path taken to capture mobile camera imagery.

Figure 49 shows an image from each sensor. There are significant variations in focal length, white balance, and image clarity. Although all of the sensors have rolling shutters, the increased frame-rate of the GoPro and iPhone resulted in much sharper images when compared to the DSLR. The DSLR was initialized using standard video capture settings, instead of capturing high-resolution, high-sensitivity still images at a lower framerate.

4.4 Experiment 4 - Extract Relative Pose from Matched Images

Linked Image Database.

The first part for this experiment is to build a database of image feature descriptors from the geo-referenced camera data. For this, the rectified images from s_0 to s_5 are passed through an OpenCV image processing algorithm to find all of the SIFT, SURF, and ORB features. Figure 50 shows the SIFT features detected in an example image,

Figure 49: Images captured from the DSLR (top), iPhone (middle), and GoPro (bottom).

while Figure 51 and Figure 52 show SURF and ORB features, respectively. Note that there are several features detected on the edges of the images, along with some on the actual platform. These features are compiled into the reference list as a matter of convenience. While they will at best not be used at all, there is a possibility that they could cause a mismatch later. The feature descriptors for the different sensors are then compiled into a single list. This is repeated for each geo-referenced image frames. The feature lists, organized by image frame, are stored in a database.

Matching a Test Image.

A test image is selected from each of the mobile camera images to test the feature matching and pose recovery functions of this research. The test image selected, shown in Figure 53, contains some unique hallway features that would be apparent to a human observer. First, it is one of only two intersections where one hallway ends at a 'T' intersection with another hallway. Second, it is the only location in the test area that has certain safety equipment, including an Automated External Defibrillator (AED) station, a fire extinguisher, and a fire alarm. These provided enough unique image features to provide a good match to the geo-referenced image database.

The test image, after being rectified, is processed using the same three image feature algorithms. Figure 54 shows the SIFT features in the test image, while Figure 55 shows the SURF features and Figure 56 shows the detected ORB features.

Once the test image has been processed, the image features are compared to the feature database using RANdom SAmple Consensus (RANSAC) to determine if the mobile image has some overlap with a geo-referenced image. The fit statistics from the RANSAC algorithm are used to determine the likelihood of a match and amount of image overlap. To reduce the possibility of having a false match, the three most likely matches are displayed to the user. The user can then select a match between

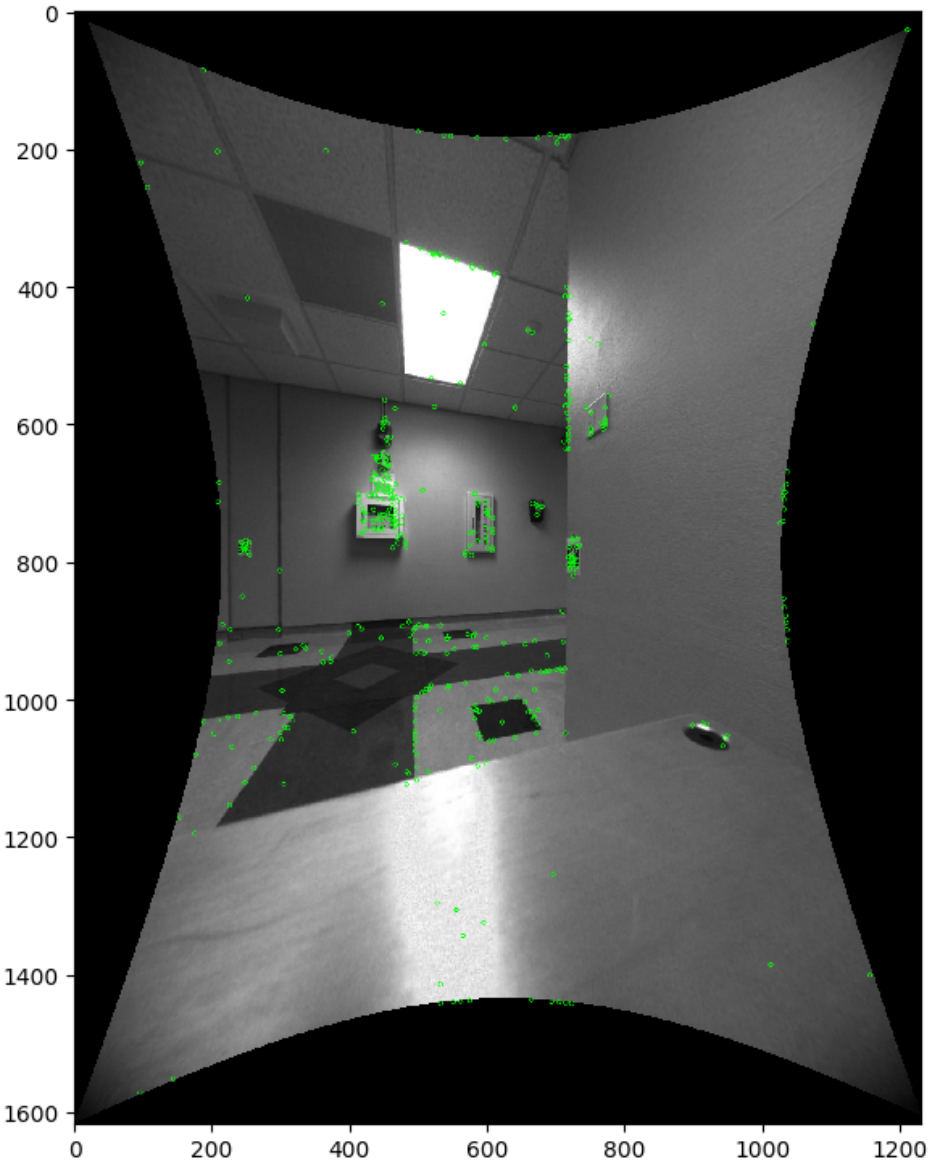

Figure 50: Camera s_0 image with SIFT algorithm features shown.

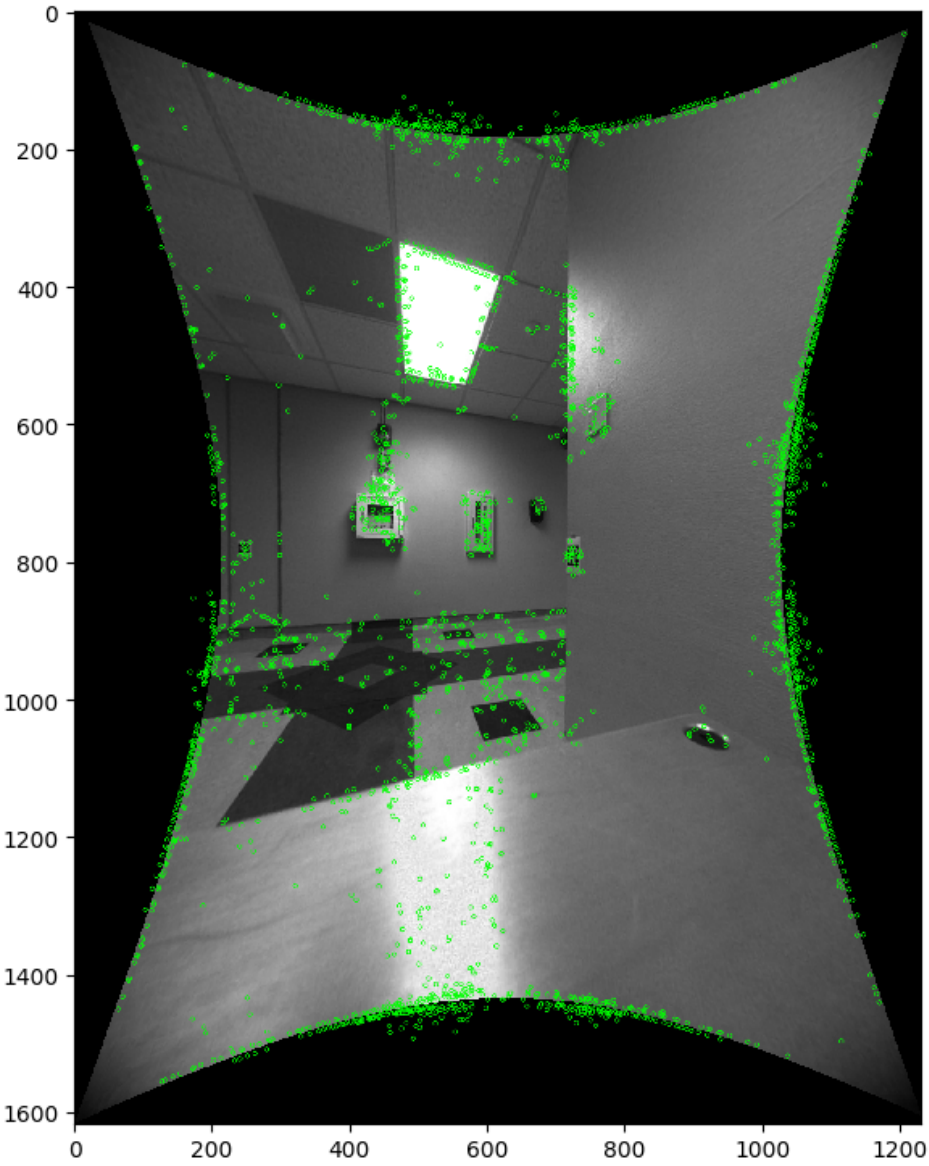

Figure 51: Camera s_0 image with SIFT algorithm features shown.

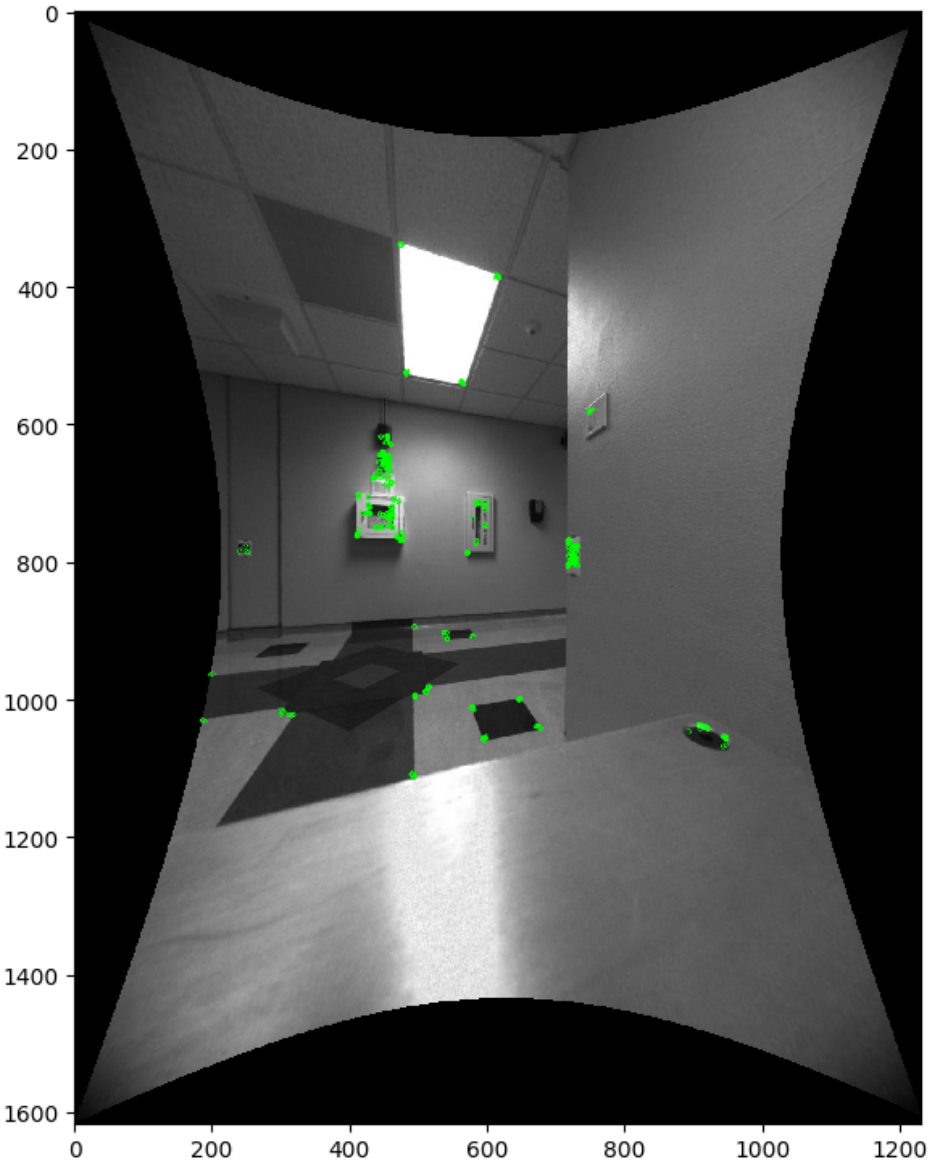

Figure 52: Camera s_0 image with ORB algorithm features shown.

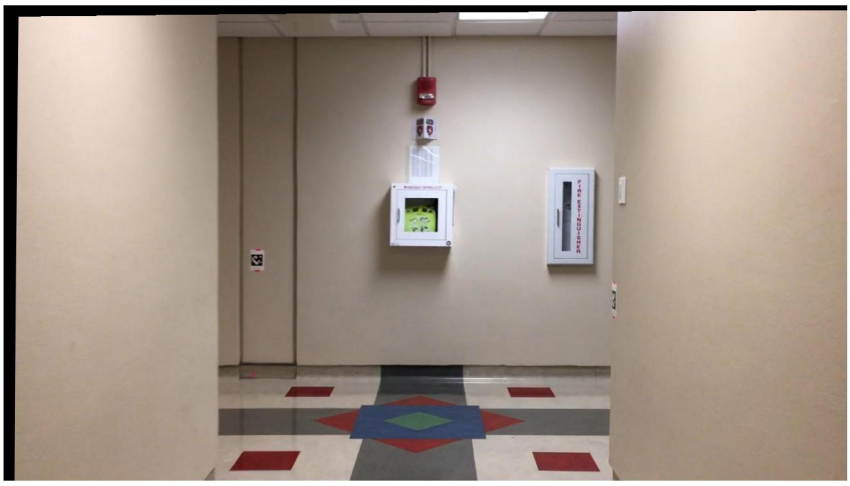

Figure 53: Rectified image from mobile camera, used to test image matching and pose recovery.

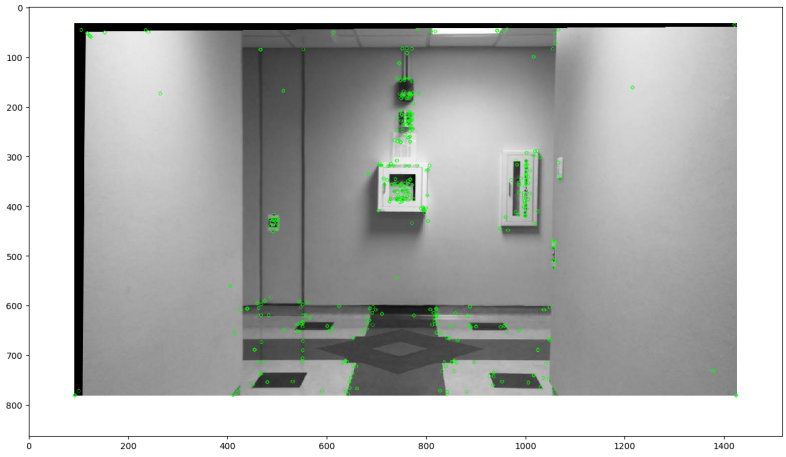

Figure 54: Detected SIFT features in the test image.

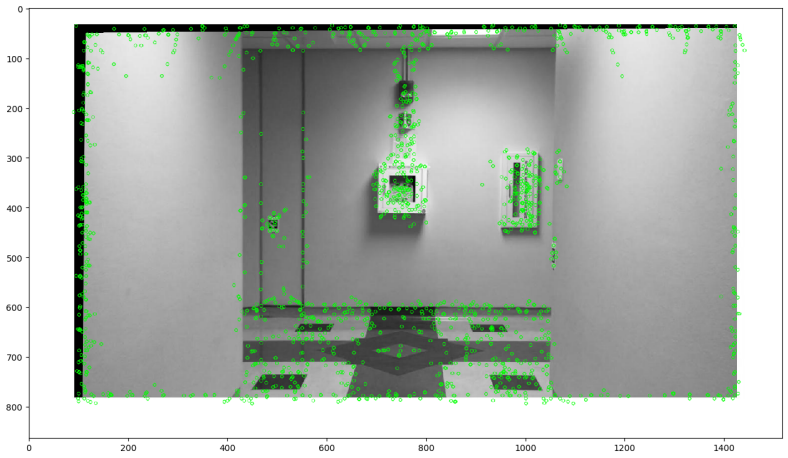

Figure 55: Detected SURF features in the test image.

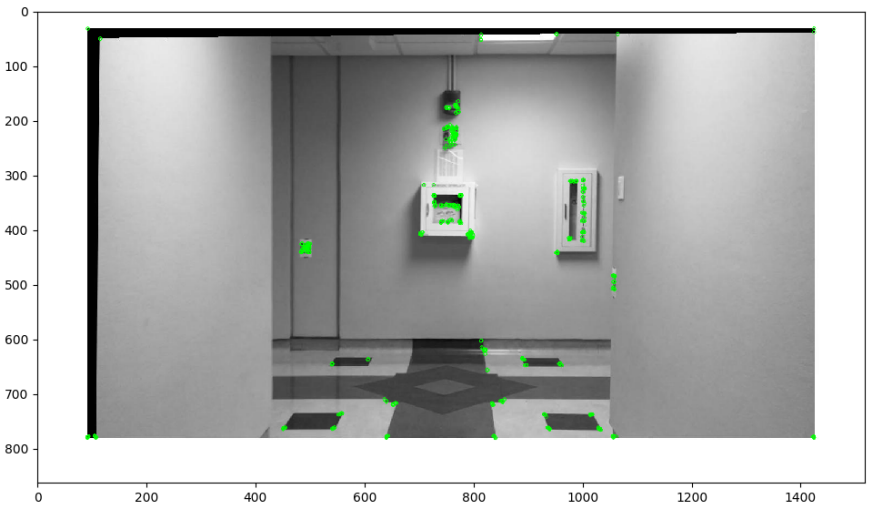

Figure 56: Detected ORB features in the test image.

two overlapping images. In this case, as pictured in Figure 57, the top three matches are all from the correct area, and appear very similar.

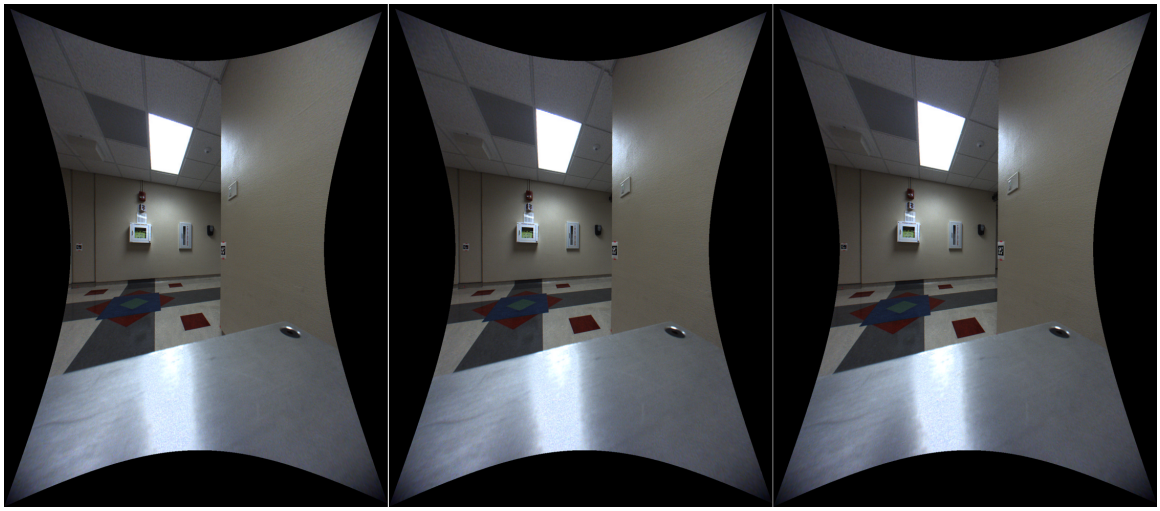

Figure 57: Top three image matches returned by computer.

The features between the geo-referenced image and the test image were compared for all three feature descriptors. The SIFT and SURF algorithms provided plenty of comparable features, but the ORB descriptor did not have enough matched features to recover a relative pose. Figure 58 shows the matched SIFT features in both images. The SURF feature matches, shown in Figure 59, had very similar results. As pictured in Figure 60, there are not enough ORB features to recover a pose.

Pose Recovery.

After a matching image set is found, the essential matrix is calculated for both images. Next, the essential matrices are used to recover the relative $\mathbf{R}$ and $\mathbf{T}$ matrices shown in Equation 17 and Equation 18, respectively.

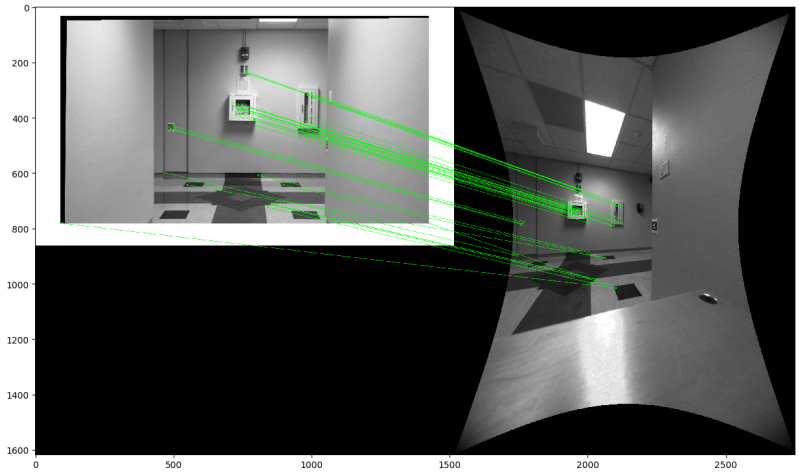

Figure 58: SIFT features matched between images.

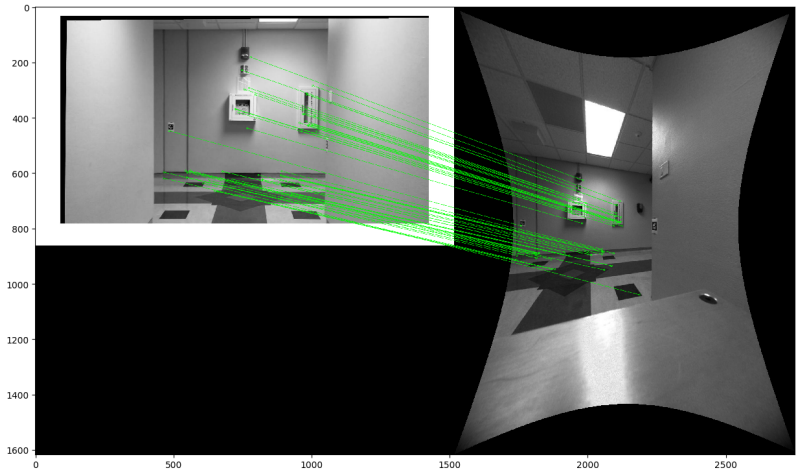

Figure 59: SURF features matched between images.

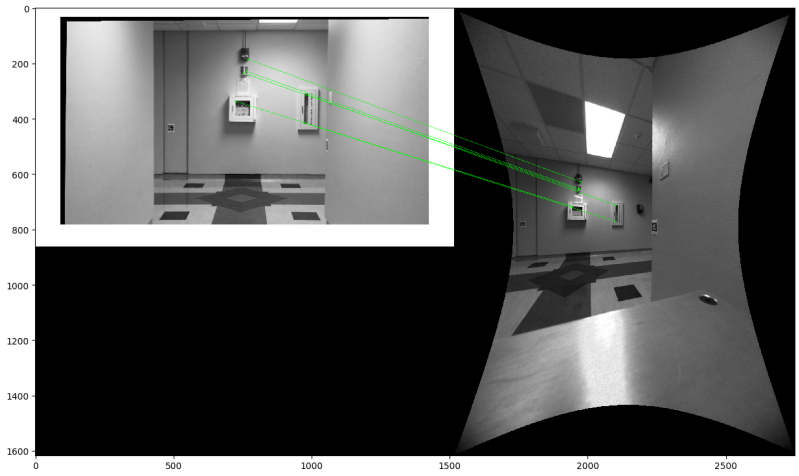

Figure 60: ORB features matched between images.

$$\mathbf{R} = \begin{bmatrix} 0.9647697 & -0.2630994 & -0.0006935 \\ 0.2630952 & 0.9647983 & 0.0002675 \\ 0.0005987 & 0 - 0.0004458 & 0.9999997 \end{bmatrix} \tag{17}$$

$$\mathbf{T} = \begin{bmatrix} 0.7370602 \\ 0.6758260 \\ 0.0011784 \end{bmatrix} \tag{18}$$

As discussed in Subsection 2.8, **R** and **T** describe the rotation and translation that would move the mobile camera into the spherical camera frame. Following the procedure in Section 3.9, the translation and orientation were calculated as shown in Figure 61. The final step was to convert the mobile camera pose from the camera coordinate frame back to the local environment frame using the relative rotations and translations.

For the test image, the first matching image was frame 2086 from the SLAM

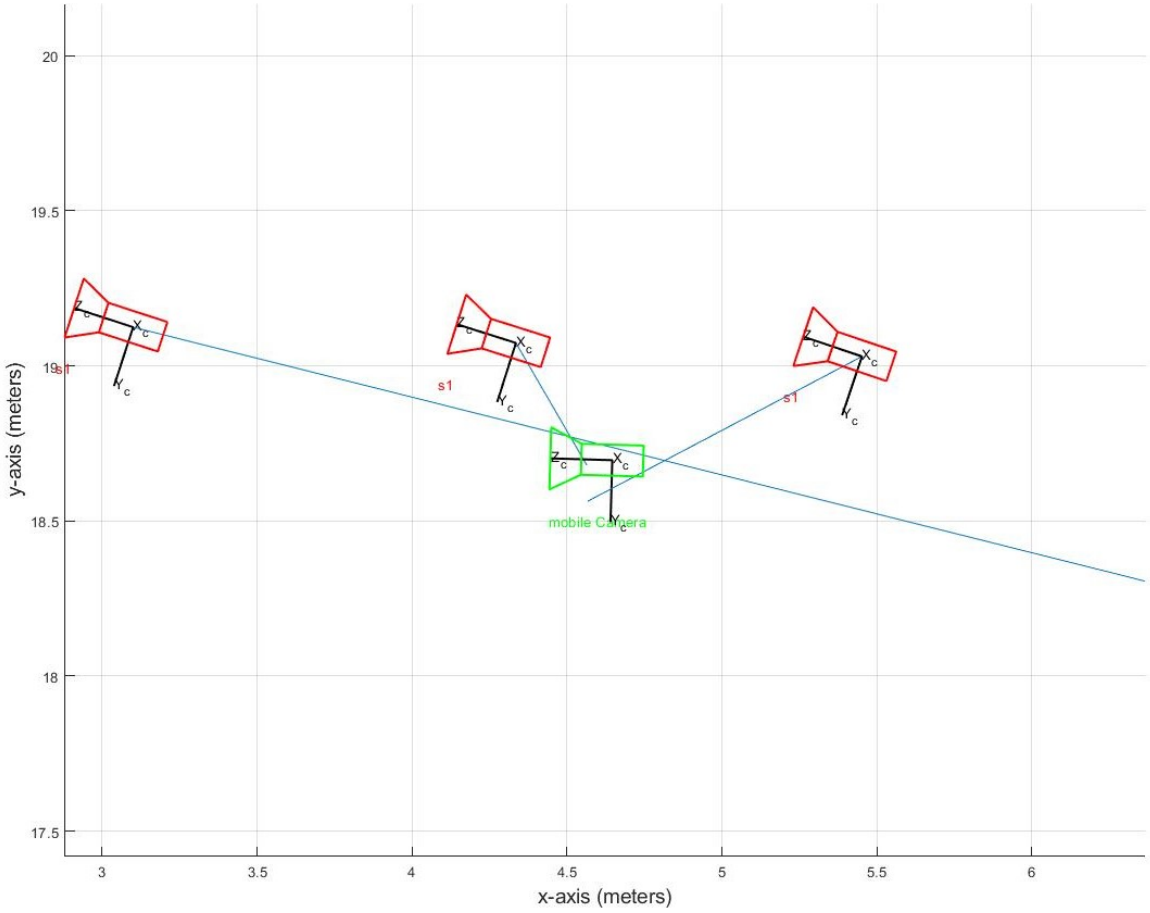

Figure 61: Recovery of pose translation using three intersecting vectors.

dataset. The geo-located image has a location of $x = 2.874$ m, $y = 19.090$ m, and $z = 0.975$ m. After calculating the transformations for frames 2071, 2086, and 2101, and applying the transformation in Equation 12 and Equation 11, the final position of the mobile camera is given as $x = 4.645$ m, $y = 18.6957$ m, and $z = 0.989$ m.

V. Conclusion

5.1 Overview

This chapter completes the thesis by drawing conclusions, and outlining the research significance and areas for further study. Section 5.2 details notable conclusions drawn from the results and analysis. Section 5.3 emphasizes the impact that the research has on the fields of mapping, navigation, and computer vision. It also describes the research impact to the USAF, AFIT, and the ANT Center. Finally, Section 5.4 outlines recommended experiment changes and future work that can be made to improve the results.

5.2 Conclusions

As hypothesized, the results demonstrated that a high SWAP-C SLAM system can be used to determine the position and orientation of low SWAP-C cameras. Additionally, images taken at different times from different sensors can be located in the environment without geo-referencing individual features.

Each component of the thesis was successful. First, the accuracy of range estimation based on an image of known feature sizes is sufficient for improving the results factor graph SLAM. Next, sets of images were matched to location and orientation data created using SLAM. Bearing and range factors derived from imagery dramatically improved the optimized pose solution. Imagery, from multiple mobile systems, was taken and calibrated using the same environment. Finally, the data from the first three experiments was successfully combined. The camera positions relative to the geo-located sensor were used to find the camera position in the overall environment.

5.3 Research Significance

This two-step process of building a map of geo-referenced images, and then matching mobile images to the map, combines the best practices of fixed infrastructure tracking and SLAM. This results in an improvement on the currently available solutions. The SLAM solution becomes much more accurate by using a high SWAP-C system instead of trying to do optical SLAM with every sensor. Additionally, the SLAM method of building a map enables truthing in a much larger environment than localization methods with like Vicon systems.

By demonstrating that common image sensors can be localized in an environment based on relative positioning of other image sensors, this research had a significant impact to the indoor navigation community. The methods, processes, and data can be used as a foundation for more focused research projects involving SLAM, sensor positioning, or autonomous navigation. These methods can be used in combination with other methods and sensors to increase accuracy and enable projects that aren't currently feasible. For instance, if current low-cost cameras can be localized in large indoor environments without setting up permanent infrastructure, then these methods could be used for navigation of robotic swarms or arrays of sensors.

The independence of the mobile data collection and the geo-referencing stages support a noteworthy application of this research. Namely, this method can be applied to determine camera position in environments that have never been properly mapped or associated with calibrated imagery. If there is video or imagery data of a specific event, along with some knowledge of the camera's intrinsic parameters, the geo-referencing and pose recovery can be performed afterward. This has resounding implications in the Department of Defense as well as in public environments for events that require physical reconstruction. For instance, imagery from bystanders and first responders to fires, earthquakes, active shooter situations, and other emergencies can

be used to determine locations of personnel and other points of interest.

5.4 Future Work

Equipment Changes.

There were several obstacles and issues with the equipment and its implementation that presented themselves while executing the experiments. The most apparent was the way that the odometry factor data was collected, but issues with the camera and platform also deserve mentioning.

Gathering each odometry pose with the current ICP technique is far too noisy, varying in accuracy by several meters just by varying the ICP parameters slightly. This could be an artifact of the specific LIDAR sensor, or just a result of data disassociation when the ICP algorithm associates different places on a flat plane as being the same point. The variance may be mitigated by making one of more of the following changes:

1. Substitute ICP odometry for wheel movement measurements.
2. Add an inertial sensor to provide an initial estimate to help alleviate local minimums.
3. Add side-facing ultrasonic range sensors to improve the side-to-side positioning data.
4. increase LIDAR resolution or frame rate to negate data disassociation.

The panoramic camera used to create the geo-referenced imagery has a very low sensor resolution, having an individual sensor resolution of 2MP. Additionally, the legacy IEEE-1394b interface severely limits the bandwidth and therefore frame rate. Switching to a different global-shutter camera that has a much higher resolution would

improve the image feature matches in Experiment 4. A higher-resolution camera would also improve the range and bearing estimation because the angular resolution will also improve. Geo-located imagery could also be collected with non-spherical or non-panoramic cameras if multiple cameras are synchronized or the hallways are traversed in multiple passes with a single camera pointed at different angles for each pass.

The LIDAR sensor, while adequate, was designed as an outdoor LIDAR for autonomous vehicle navigation. Switching to a sensor designed specifically for indoor use, with a shorter minimum range, would likely have a significantly different SLAM result. One alternative that should be explored is the switch to a 2D LIDAR that has either a much faster scan rate or a higher angular resolution. Finally, adding additional compensation to the LIDAR data to correct for the distance traveled by the sensor during each revolution may also improve the accuracy of the ICP algorithm. One method for testing this improvement would be to perform each LIDAR scan while the platform is stationary, only moving between frame measurements.

The platform design and its use likely introduced a significant amount of error. First, the operator stands behind the platform to guide it, interfering with both the LIDAR data and the camera images. Mounting the platform above the operator would negate a majority of the interference, without impacting sensor height if the operator is seated. This mobile platform design has appeared in other LIDAR navigation projects, including Schwesinger's warehouse navigation project [47].

Since the camera and LIDAR were not designed to be compatible, several hardware trade-offs were made. Several of the aluminum pieces supporting the camera obstruct some of the LIDAR signals. Also, the plate that the camera is attached to is very large to avoid touching the LIDAR, but it blocks a portion of the camera images. Switching to a camera and LIDAR combination that was specifically designed to

work together would remove most of the obstructions.

The number of ArUco markers used was not analyzed. There may be some benefit to accuracy by either increasing or decreasing the number of landmarks used in the SLAM process. Since the ArUco marker dictionaries can cover thousands of unique identifiers, many more features could be labeled with markers. In any case, their ease of application and removal, coupled with their ability to identify non-ambiguous loop closures, makes them a good choice.

The mobile camera images were collected using standard settings on sensors that are several years old. Testing different combinations of framerate, resolution, and sensor sensitivity may improve the image results. Although the cameras used represent a wide range of commercially available sensors, testing additional sensors with different features may be warranted.

Code Optimization.

This thesis used different programming languages for subsections of each problem. For instance, Python 3.6 was used to run openCV, while running ICP and plotting data was done in MATLAB 18b. Additionally, the LIDAR and camera data was initially processed with proprietary software. Although this was faster to develop, it is slower to execute and more confusing. Ideally, all the program functions would be merged into a common programming language. This language would most likely be C++, since openCV, GTSAM, and ICP are already supoorted. The end result would be a unified code structure where images, LIDAR data, and noise characteristics are passed as inputs, and images with position and orientation are passed as outputs.

The image-based range estimation method may be too slow if the SLAM solution needs to operate in real time. Although using the Rational 1-2 model has the lowest RMSE, the Power 2 or Power 1 models would result in a more computationally efficient

range estimation function. The 5% to 15% increase in RMSE may be worth the faster speed.

Another improvement would be to make the localization stage easier to use. The process of matching images to the high SWAP-C SLAM results should be optimized and compiled into a stand-alone phone or computer application. This would allow for localization of a wider range of cameras and be available to a larger set of users.

Machine Learning Approach.

This research used a RANSAC approach to find geo-referenced images with the most data in common with the test data as possible. Another approach to matching would be to use machine learning to detect and match unique frames. Machine learning may find patterns and matches faster or better than the RANSAC approuch.

UcoSLAM.

UcoSLAM is a library for Simultaneous Localization and Mapping using matched keypoints between frames. This is an extension of SPM-SLAM [38], with the addition of optical odometry using KeyPoints. It is able to operate with monocular cameras, stereo cameras, or RGB-D cameras. It uses frame-to-frame keypoint matching to determine odometry, combined with the ArUco library for detecting ArUco markers, which can be placed in the environment to improve tracking. It has many features that make it an interesting option for future work:

1. SLAM solutions can be processed in real-time.
2. Includes a graphical user interface for processing videos, visualizing maps, and calibrating cameras.
3. Save and load the generated maps.

4. Using ArUco markers enhances the initial trajectory estimates and provide loop-closures.

5. Using ArUco markers allows for estimating the map scale from monocular cameras.

6. A arallelized KeyPoint detector improves tracking speed.

7. Operates on Windows, Linux and Android.

Appendix A. ArUco Marker To Bearing and Range Factor Script

```
function keyframes = calcKeyframes(centerMin,frames,threshold,plotEnable,marker)
% This function returns the key frames, images likely to have the optimal
% landmark data, from a list of the frames, the distance from the center of
% the marker to the center of the image, and the threshold value to allow
% for landmark re-visits
    % Find out where the distance is within threshold.
    nonZeroElements = centerMin < threshold; % Logical vector of 0's and 1's
    % Find out indexes where it goes from 0 to 1 or 1 to 0.
    diffPattern = diff(nonZeroElements);
    % Find out where every subset starts and stops.
    subsetStarts = strfind(diffPattern, 1);
    subsetEnds = strfind(diffPattern, -1);
    % Loop through all humps extracting each segment into a cell
    for subset = 1 : length(subsetEnds)
      indexRange = subsetStarts(subset) : subsetEnds(subset);
      data(subset).subset = [indexRange;centerMin(indexRange)];
      [M(subset),I(subset)] = min(centerMin(indexRange));
      %data(subset).minFrame = square_data(frames(I(subset))).framenumber;
    end
    % return
    for k = 1:size(data,2)
        keyframes(k) = frames(data(k).subset(1,I(k)));
    end
    % Should this function return plots?
```

```
    if plotEnable == 1
    % Plot frame centers, with keyframes
        figure()
        %plot(frames,centerMin, 'bo-')
        hold on
        for j = 1:size(data,2)
            plot(frames(data(j).subset(1,:)),data(j).subset(2,:),'bo-')
        end
        grid on
        % Give a name to the title bar.
        set(gcf, 'Name', num2str(marker), 'NumberTitle', 'Off')
        % Show selected keyframes:
        for k = 1:size(keyframes,2)
            scatter(keyframes(k),data(k).subset(2,I(k)),40,'r','filled')
        end
    end
end
```

Appendix B. GTSAM Script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          SLAM USING GTSAM
%
% AIR FORCE INSTITUTE OF TECHNOLOGY
% ---------------------------------
%
% Author: Capt David Beargie
%
% ---------------------------------
%
% GTSAM Copyright 2010, Georgia Tech Research Corporation,
% Atlanta, Georgia 30332-0415
% All Rights Reserved
% GTSAM Authors: Frank Dellaert, et al.
% See LICENSE for the license information
%
% Factors used:
% -- Bearing and Range Factors
% -- BetweenFactorPose2 Odometry Factors
% -- Priors
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%% MATLAB Setup
clc
close all
```

```
clear

import gtsam.*

%% Load Data
load('C:\Users\User\odometryData.mat');
load('C:\Users\User\LandmarkData.mat');
load('C:\Users\User\bearingRangeFactors.mat');

% make Deltas
[DeltaDcm,DeltaP] = RetrieveDeltas(dcmTotal,S);
% rotate odometry to global reference:
[absoluteDcm_test,absoluteP_test] = IntegrateDeltas([1,0,0;0,-1,0;0,0,-1],S(:,1),Delt

for make_pose=1:size(DeltaP,2)
   delta_rpy = DcmToRpy(DeltaDcm(:,:,make_pose));
   delta_poses(make_pose,:) = [DeltaP(1,make_pose),DeltaP(2,make_pose),delta_rpy(3)];
end

for make_pose=1:size(dcmTotal,3)
   absolute_rpy(:,make_pose) = DcmToRpy(dcmTotal(:,:,make_pose));
end
% Initial starting location and orientation
initial_pose = [0.0,1.2,0.0];

% create path from absolute poses
```

```
absolute_poses = [S(1:2,:)',absolute_rpy(3,:)'];

%% Create keys for variables
for p_keys = 1:size(delta_poses,1)
i(p_keys) = symbol('x',p_keys);
end
for lm_keys = 1:size(Landmarks_m,1)
j(lm_keys) = symbol('l',lm_keys);
end

%% Create graph container and add factors to it
fg = NonlinearFactorGraph;

%% Add starting prior
priorMean = Pose2(0.0,1.2,0.0); % prior at origin
priorNoise = noiseModel.Diagonal.Sigmas([0.025; 0.025; deg2rad(1)]);
fg.add(PriorFactorPose2(i(1), priorMean, priorNoise));

%% Add odometry factors
odometryNoise = noiseModel.Diagonal.Sigmas([0.15; 0.15; deg2rad(0.5)]);
for odometry = 1:(size(delta_poses,1)-1)
    fg.add(BetweenFactorPose2(i(odometry), i(odometry+1), ...
      Pose2(delta_poses(odometry,1)+0.0*randn(),delta_poses(odometry,2)+0.0*randn(),d
end

%% Add bearing/range measurement factors
```

```
brNoise = noiseModel.Diagonal.Sigmas([deg2rad(0.005); 0.2]);
for markers = 1:size(outputFactor,2)
    for add_BR = 1:size(outputFactor(markers).pose,2)
        fg.add(BearingRangeFactor2D(i(outputFactor(markers).pose(add_BR)),...
            j(markers), ...
            Rot2(outputFactor(markers).bearing(add_BR)),...
            outputFactor(markers).range(add_BR), ...
            brNoise));
    end
end

%% Add Landmark Priors
LmPriorNoise = noiseModel.Diagonal.Sigmas([0.25; 0.25]);
for lm = 1:size(Landmarks_m,1)
    fg.add(PriorFactorPoint2(j(lm), Point2(Landmarks_m(lm,1),Landmarks_m(lm,2)), LmPr
end

%% Create Initial Guesses
initialEstimate = Values;

% initialize poses with some noise
init_noise = 0;
for init_pose=1:size(delta_poses,1)
    initialEstimate.insert(i(init_pose),Pose2(absolute_poses(init_pose,1)+0.00*randn(
end
```

```
%initialize landmarks
for init_lm=1:size(Landmarks_m,1)
   initialEstimate.insert(j(init_lm),Point2(...
       Landmarks_m(init_lm,1), ...
       Landmarks_m(init_lm,2)));
end

% print initial estimate
initialEstimate.print(sprintf('\nInitial estimate:\n'));

%% Optimize using Levenberg-Marquardt optimization
optimizer = LevenbergMarquardtOptimizer(fg, initialEstimate);
result = optimizer.optimizeSafely();
% print result
result.print(sprintf('\nFinal result:\n'));

%% Plot Covariance Ellipses
% Calculate marginals
InitialMarginals = Marginals(fg, initialEstimate);
marginals = Marginals(fg, result);

figure()
% Plot initial estimate trajectory
plot2DTrajectory(initialEstimate, 'k', InitialMarginals);
% Plot initial estimate of landmarks
```

```
plot2DPoints(initialEstimate, 'c', InitialMarginals);


figure()
% Plot result trajectory
plot2DTrajectory(result, 'k', marginals);
% Plot result landmarks
plot2DPoints(result, 'c', marginals);

%% END
```

Bibliography

[1] Hatem Alismail, L Douglas Baker, and Brett Browning. "Continuous trajectory estimation for 3D SLAM from actuated lidar". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 6096–6101.

[2] Tim Bailey and Hugh Durrant-Whyte. "Simultaneous localization and mapping (SLAM): Part II". In: *IEEE Robotics & Automation Magazine* 13.3 (2006), pp. 108–117.

[3] Tim Bailey et al. "Consistency of the EKF-SLAM algorithm". In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE. 2006, pp. 3562–3568.

[4] Herbert Bay et al. "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (2008). Similarity Matching in Computer Vision and Multimedia, pp. 346 –359. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. URL: http://www.sciencedirect.com/science/article/pii/S1077314207001555.

[5] P. J. Besl and N. D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791.

[6] I. Bogoslavskyi and C. Stachniss. "Analyzing the Quality of Matched 3D Point Clouds of Objects". In: *IEEE IROS*. 2017. URL: http://www.ipb.uni-bonn.de/pdfs/bogoslavskyi17iros.pdf.

[7] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[8] Michael Calonder et al. “BRIEF: Binary Robust Independent Elementary Features”. In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792. ISBN: 978-3-642-15561-1.

[9] Jinwoo Choi, Sunghwan Ahn, and Wan Kyun Chung. “Robust sonar feature detection for the SLAM of mobile robot”. In: *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE. 2005, pp. 3415–3420.

[10] Alejo Concha et al. “Visual-inertial direct SLAM”. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1331–1338.

[11] Michael Csorba. “Simultaneous localisation and map building”. PhD thesis. University of Oxford, 1998.

[12] Michael Csorba, Jeffrey K. Uhlmann, and Hugh F. Durrant-Whyte. “New approach to simultaneous localization and dynamic map building”. In: *Proceedings of SPIE - The International Society for Optical Engineering* (May 1996). DOI: 10.1117/12.241084.

[13] B. Della Corte et al. “A General Framework for Flexible Multi-Cue Photometric Point Cloud Registration”. In: *Robotics and Automation (ICRA), 2018 IEEE International Conference on*. 2018. URL: http://www.ipb.uni-bonn.de/wp-content/papercite-data/pdf/della-corte2018icra.pdf.

[14] Frank Dellaert. *Factor graphs and GTSAM: A hands-on introduction*. Tech. rep. Georgia Institute of Technology, 2012.

[15] Hugh Durrant-Whyte, D. Rye, and E. Nebot. “Localisation of automatic guided vehicles”. In: *Robotics Research: The 7th International Symposium (ISRR95)* (1996), 613625.

[16] Craig Glennie and Derek Lichti. "Static Calibration and Analysis of the Velodyne HDL-64E S2 for High Accuracy Mobile Scanning". In: *Remote Sensing* 2 (June 2010). DOI: 10.3390/rs2061610.

[17] Giorgio Grisetti et al. "A tutorial on graph-based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.

[18] Giorgio Grisettiyz, Cyrill Stachniss, and Wolfram Burgard. "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling". In: *Proceedings of the 2005 IEEE international conference on robotics and automation*. IEEE. 2005, pp. 2432–2437.

[19] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[20] Ying He et al. "An Iterative Closest Points Algorithm for Registration of 3D Laser Scanner Point Clouds with Geometric Features". In: *Sensors* 17 (Aug. 2017), p. 1862. DOI: 10.3390/s17081862.

[21] Janne Heikkila and Olli Silven. "A four-step camera calibration procedure with implicit image correction". In: *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE. 1997, pp. 1106–1112.

[22] Adrian Kaehler and Gary Bradski. *Learning OpenCV 3: computer vision in C++ with the OpenCV library.* " O'Reilly Media, Inc.", 2016.

[23] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. "iSAM: Incremental smoothing and mapping". In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378.

[24] Daphne Koller, Nir Friedman, and Francis Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[25] Martin Lambers, Hendrik Sommerhoff, and Andreas Kolb. “Realistic Lens Distortion Rendering”. In: ().

[26] Henning Lategahn, Andreas Geiger, and Bernd Kitt. “Visual SLAM for autonomous ground vehicles”. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1732–1737.

[27] J. J. Leonard and H. F. Durrant-Whyte. “Mobile robot localization by tracking geometric beacons”. In: *IEEE Transactions on Robotics and Automation* 7.3 (1991), pp. 376–382. ISSN: 1042-296X. DOI: 10.1109/70.88147.

[28] Kenneth Levenberg. “A method for the solution of certain non-linear problems in least squares”. In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168.

[29] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[30] Donald W Marquardt. “An algorithm for least-squares estimation of nonlinear parameters”. In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[31] Adam Milstein. “Occupancy grid maps for localization and mapping”. In: *Motion Planning*. InTech, 2008.

[32] Michael Montemerlo and Sebastian Thrun. “FastSLAM 2.0”. In: *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics* (2007), pp. 63–90.

[33] Michael Montemerlo et al. “FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem”. In: *national conference on artificial intelligence* 1.1 (2002), pp. 1–6.

[34] Michael Montemerlo et al. “FastSLAM: A factored solution to the simultaneous localization and mapping problem”. In: *Aaai/iaai* 593598 (2002).

[35] Hans P Moravec and Alberto Elfes. “High resolution maps from wide angle sonar”. In: *Proceedings of the IEEE Conference on Robotics and Automation*. 1985, pp. 19–24.

[36] Jorge J Moré. “The Levenberg-Marquardt algorithm: implementation and theory”. In: *Numerical analysis*. Springer, 1978, pp. 105–116.

[37] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[38] Rafael Muoz-Salinas, Manuel Marn-Jimenez, and R. Medina-Carnicer. “SPM-SLAM: Simultaneous localization and mapping with squared planar markers”. In: *Pattern Recognition* Volume 86 (2019), pp. 156–171. ISSN: 21530866. DOI: 10.1109/ROBIO.2015.7418951.

[39] David Nistér. “An efficient solution to the five-point relative pose problem”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (2004), pp. 756–770.

[40] Hans P. Moravec and Alberto Elfes. “High resolution maps from wide angle sonar”. In: vol. 2. Apr. 1985, pp. 116 –121. DOI: 10.1109/ROBOT.1985.1087316.

[41] E. Palazzolo and C. Stachniss. “Fast Image-Based Geometric Change Detection Given a 3D Model”. In: 2018. URL: http://www.ipb.uni-bonn.de/pdfs/palazzolo2018icra.pdf.

[42] Lina M Paz, Juan D Tardós, and José Neira. “Divide and conquer: EKF SLAM in $O(n)$”. In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 1107–1120.

[43] Volker Rodehorst, Matthias Heinrichs, and Olaf Hellwich. "Evaluation of relative pose estimation methods for multi-camera setups". In: *International Archives of Photogrammetry and Remote Sensing (ISPRS08)* (2008), pp. 135–140.

[44] John Rogers et al. "Effects of sensory precision on mobile robot localization and mapping". In: *Experimental Robotics*. Springer. 2014, pp. 433–446.

[45] Edward Rosten and Tom Drummond. "Machine Learning for High-Speed Corner Detection". In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443. ISBN: 978-3-540-33833-8.

[46] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[47] Dylan Schwesinger and John Spletzer. "A 3D approach to infrastructure-free localization in large scale warehouse environments". In: *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. 2016, pp. 274–279. DOI: 10.1109/COASE.2016.7743418.

[48] Erik Smistad et al. "Medical image segmentation on GPUs - A comprehensive review". In: *Medical Image Analysis* 20 (Feb. 2015), pp. 1–18. DOI: 10.1016/j.media.2014.10.012.

[49] Randall Smith, Matthew Self, and Peter Cheeseman. "Estimating uncertain spatial relationships in robotics". In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. 1987, pp. 850–850. DOI: 10.1109/ROBOT.1987.1087846.

[50] Joan Solà. "Simultaneous localization and mapping with the extended kalman filter". In: *A very quick guide with MATLAB code* (2013).

[51] Z. Su et al. "Global localization of a mobile robot using lidar and visual features". In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2017, pp. 2377–2383. DOI: 10.1109/ROBIO.2017.8324775.

[52] A. J. B. Trevor, J. G. Rogers, and H. I. Christensen. "Planar surface SLAM with 3D and 2D sensors". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 3041–3048. DOI: 10.1109/ICRA.2012.6225287.

[53] Matthew R Walter, Ryan M Eustice, and John J Leonard. "Exactly sparse extended information filters for feature-based SLAM". In: *The International Journal of Robotics Research* 26.4 (2007), pp. 335–359.

[54] H Durrant Whyte. "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms". In: *Robotics and Automation Magazine* (2006).

[55] Oliver Wulf et al. "Ground truth evaluation of large urban 6D SLAM". In: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE. 2007, pp. 650–657.

[56] Mingchuan Zhang, Kai-Bor Yu, and Robert M Haralick. "Fast correlation registration method using singular value decomposition". In: *International Journal of Intelligent Systems* 1.3 (1986), pp. 181–194.

[57] Zhengyou Zhang. "Iterative point matching for registration of free-form curves and surfaces". In: *International Journal of Computer Vision* 13.2 (1994), pp. 119–152. ISSN: 1573-1405. DOI: 10.1007/BF01427149. URL: https://doi.org/10.1007/BF01427149.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
10–02–2013	Master's Thesis	Sept 2017 — Mar 2019

4. TITLE AND SUBTITLE

ASSESSMENT OF CAMERA POSE ESTIMATION USING GEO-LOCATED IMAGES FROM SIMULTANEOUS LOCALIZATION AND MAPPING

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Beargie, David W., Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-19-M-009

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

intentionally left blank

10. SPONSOR/MONITOR'S ACRONYM(S)

intentionally left blank

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research proposes a method for enabling low-cost camera localization using geo-located images generated with factorgraph-based Simultaneous Localization And Mapping (SLAM). The SLAM results are paired with panoramic image data to generate geo-located images, which can be used to locate and orient low-cost cameras. This study determines the efficacy of using a spherical camera and LIDAR sensor to enable localization for a wide range of cameras with low size, weight, power, and cost. This includes determining the accuracy of SLAM when geo-referencing images, along with introducing a promising method for extracting range measurements from monocular images of known features.

15. SUBJECT TERMS

Navigation; Localization; Simultaneous Localization and Mapping; LIDAR; Mapping;

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Capt Aaron Canciani, PhD, AFIT/ENG
U	U	U	U	125	19b. TELEPHONE NUMBER *(include area code)* 937-255-3636 x4618; Aaron.Canciani@afit.edu