Integration of 3D and 2D Imaging Data for Assured Navigation in Unknown
Environments

A thesis presented to
the faculty of
the Russ College of Engineering and Technology of Ohio University

In partial fulfillment
of the requirements for the degree
Master of Science

Evan T. Dill

March 2011

© 2011 Evan T. Dill. All Rights Reserved.

This thesis titled
Integration of 3D and 2D Imaging Data for Assured Navigation in Unknown
Environments

by
EVAN T. DILL

has been approved for
the School of Electrical Engineering and Computer Science
and the Russ College of Engineering and Technology by

Maarten Uijt de Haag

Professor of Electrical Engineering and Computer Science

Dennis Irwin

Dean, Russ College of Engineering and Technology

ABSTRACT

DILL, EVAN T., M.S., March 2011, Electrical Engineering

Integration of 3D and 2D Imaging Data for Assured Navigation in Unknown
Environments

Director of Thesis: Maarten Uijt de Haag

As technology advances in the area of mobile vehicles, the need for precise reliable navigation increases with it. Whether the vehicle is an unmanned aerial vehicle (UAV), a manned aerial vehicle (MAV), or an intelligent ground vehicle (IGV), there is a constant need for precise navigation capabilities. This need spurred the invention and development of many navigation systems including the most useful system to date, the Global Positioning System (GPS). GPS is a powerful tool that can reliably give meter level accuracy on a world-wide scale. With this capability, GPS is the answer to a significant number of navigation problems, but it is not the answer to them all. Since GPS relies on exterior signals from orbiting satellites, tasks such as underground navigation and navigation in dense foliage can be difficult due to signal strength attenuation as it passes through these media. GPS is also very susceptible to multipath at the receiver. If the receiver is operating in a building or in a heavy urban environment, the multipath created can degrade the received signal to the point of losing its true capabilities. Lastly, GPS capabilities are ideal for military applications. However, any system that uses exterior signals for military applications must deal with the possibility of interference, jamming, or even an attack on the system in a wartime scenario.

Although, the list of scenarios in which GPS is not a viable answer is small, it is important that those scenarios be addressed. One viable possibility is developing a new system that complements GPS by having functionality in scenarios in which GPS is a poor option or not an option at all. This thesis describes and discusses one such possibility that could complement GPS. The proposed system is a self contained system that would use multiple sensors and the environment around them for navigation. This method would integrate three-dimensional (3D) point cloud data, two-dimensional (2D) gray-level (intensity) data, 2D digital image data, and data from an Inertial Measurement Unit (IMU). Using the 2D and 3D image frames, key features can be extracted from the surroundings such as planar surfaces, lines, and corners. Positions of the features can be calculated with respect to the sensors observing them. With these features, consecutive observations in the 3D and 2D image frames can be used to compute and estimate position and orientation change of the sensors. The use of 3D image features for positioning suffers from limited feature observability, which results in deteriorated position accuracies. Also, the 2D imagery suffers from an unknown depth when estimating the position from consecutive image frames. It is expected that the integration of these two data sets will alleviate the problems with the individual methods, resulting in a position and attitude determination method with a high level of assurance. An IMU will be used to set up the tracking gates necessary to perform feature association in consecutive frames. Finally, the position and orientation change estimates can be used to correct for the IMU drift errors using an Extended Kalman Filter (EKF), and output

inertial data that drifts at a significantly slower rate than that of the IMU without this correction.

Approved: _____

Maarten Uijt de Haag

Professor of Electrical Engineering and Computer Science

ACKNOWLEDGMENTS

The completion of this thesis is a culmination of years of education and learning in which multiple people have contributed help along the way. First and foremost I would like to thank my lord and savior for blessing me with the knowledge and strength necessary to accomplish this achievement.

Second, I would like to thank the engineering professionals and colleagues that have guided me along the path. A significant amount of the research performed for this thesis was funded through a Dayton Area Graduate Studies Institute (DAGSI) fellowship, which granted me the privilege of working with Donald Venable and Dr. Jacob Campbell at Wright-Patterson Air Force Base. These two individuals were constantly willing to answer questions and provide advice despite their own very busy work schedules. My two colleagues and office mates, Mathew Cosgrove and Dustin Bates, have helped greatly as well. Through this entire process, they have provided very helpful and enlightening conversations and debates on almost a daily basis. The last of the engineers that I would like to thank are my professors Dr. Maarten Uijt de Haag, Dr. Frank Van Graas, and Dr. Wouter Pelgrum. These three professors are some of the most intelligent and practical people that I have ever encountered. It has been a true privilege to have the opportunity to work with, learn from, and study under individuals as brilliant and talented as these three. From these individuals, I would like to give an extra thanks to my advisor, Dr. Uijt de Haag, for all of the extra work he has done. He has provided countless hours of teaching and conversations as well as valuable experiences that I would not have had

without him. I couldn't have asked for a better advisor and friend though this entire process.

Next, I would like to thank my parents for the support they have provided my entire life, and instilling in me the principles of hard work and dedication through their actions. My mother Cindy, has provided an endless amount of encouragement from the time I was a child, to the completion of this paper. My father Jeff, an electrical engineering professor, has contributed more to my learning than any other figure in my life. Growing up he taught me how to solve problems and think critically. It truly is because of him that I am an engineer today, and I would consider myself successful if someday I am half the engineer he is.

Lastly, I would like to thank my lovely wife, Jenna, without whom I do not know where I would be today. Her love and encouragement has been a constant source of motivation that has kept me working and striving to become the best husband, man, and engineer that I can be. No words can do justice to how important she has been to me over the past few years, so a simple I love you must suffice.

Thank you to all of you.

TABLE OF CONTENTS

	Page
Abstract	3
Acknowledgments.....	6
List of Tables	10
List of Figures	11
List of Acronyms	14
Chapter 1: Introduction	15
Chapter 2: Background	21
2.1 GNSS Navigation Systems	21
2.2 Inertial Navigation Systems.....	23
2.3 Optical Navigation Systems.....	28
2.4 Navigation Techniques	33
2.4.1 SLAM	33
2.4.2 SIFT	36
2.4.3 Frame Rotation.....	39
2.4.4 Kalman Filtering	40
Chapter 3: Sensors and Data Setup	47
3.1 CSEM SwissRanger SR-3000 Camera	47
3.2 Point Grey Firefly	48
3.3 Honeywell HG1930	50
3.4 Sensor Setup	50
3.5 Camera Calibration	52
3.6 Data Collection and Matching	57
Chapter 4: Pre-Processing and Association Methodology.....	62
4.1 Wavelet De-Noising	62
4.2 Feature Extraction.....	66
4.2.1 3D Planar Extraction.....	66
4.2.2 3D and 2D Line Extraction	68
4.2.3 3D and 2D Corner Extraction	74

4.2.4 3D and 2D Point Extraction and Association	76
4.2.5 Determined 3D and 2D Extraction Thresholds.....	79
4.3 Feature Association.....	80
4.3.1 3D Feature Association.....	83
4.3.2 2D Feature Association.....	85
4.3.3 Determined 3D and 2D Association Thresholds	89
Chapter 5: Multiple Feature Type Integration Mechanization	92
5.1 Planar Surface Navigation Equations	93
5.2 3D Point and Corner Navigation Equations	96
5.3 2D Point and Corner Navigation Equations	99
5.4 2D and 3D line Navigation Equations	104
5.5 Extended Kalman Filter Equations	107
Chapter 6: Results	112
6.1 Simulation Description	112
6.2 Simulation Errors.....	115
6.2.1 Inertial Errors	115
6.2.2 Feature Errors.....	115
6.3 Simulation Observability	118
6.4 Simulation Results	119
6.4.1 Experiments 1 and 2.....	122
6.4.2 Experiments 3 through 5	123
6.4.3 Experiments 6 through 9	126
6.4.4 Experiments 10 through 13.....	129
6.4.5 Experiments 14 through 23	133
6.4.6 Experiments 24 and 25.....	141
Chapter 7: Conclusion.....	144
Chapter 8: Future work	147
References	151

LIST OF TABLES

	Page
<i>Table 1. Measurement and Features Associated with the Sensors.</i>	17
<i>Table 2. Calculated Thresholds for Feature Extraction.</i>	79
<i>Table 3. Calculated Thresholds for Feature Association.</i>	91
<i>Table 4. Inertial Error Values.</i>	115
<i>Table 5. Unaddressed Errors.....</i>	118
<i>Table 6. Description of Simulation Experiments.</i>	120

LIST OF FIGURES

	Page
<i>Figure 1. Sensor Integration Philosophy.....</i>	16
<i>Figure 2. Example Features Shown in a 2DDigital Image.</i>	17
<i>Figure 3. 3D, 2D, and IMU Integration Approach.....</i>	19
<i>Figure 4. GPS Satellite Constellation [14].....</i>	22
<i>Figure 5. The Vestibular System [15].....</i>	24
<i>Figure 6. Strapdown Inertial Mechanization [15]</i>	26
<i>Figure 7. Inertial Mechanization with Error Compensation (taken from [15]).....</i>	27
<i>Figure 8. Stereo Vision [22]</i>	30
<i>Figure 9. Position Change of a Feature</i>	32
<i>Figure 10. SLAM Representation (adapted from [24])</i>	35
<i>Figure 11. CSEM SwissRanger SR-3000 Camera.</i>	48
<i>Figure 12. Point Grey Firefly.</i>	49
<i>Figure 13. Honeywell HG1930 [36].....</i>	50
<i>Figure 14. Data Collection Platform.....</i>	51
<i>Figure 15. Data Collection Platform Connection Block Diagram.....</i>	52
<i>Figure 16. Data Collection Environment in Stocker Center(Ohio University Engineering Building)</i>	52
<i>Figure 17. Data Collection Scenario for Camera Calibration.....</i>	56
<i>Figure 18. Camera Calibration Board</i>	57
<i>Figure 19. Firefly Image Used for Camera Calibration</i>	57
<i>Figure 20. Time Tag Matching for 3D Imager and 2D Imager.....</i>	59
<i>Figure 21. Time Tag Matching for 3D Imager and IMU.</i>	61
<i>Figure 22. Block diagram of full system.</i>	62
<i>Figure 23. Original and De-noised Intensity Images.</i>	65
<i>Figure 24. Original and De-noised Digital Images.....</i>	65
<i>Figure 25. Corner Extraction on Original, LPF, and Wavelet De-Noised Image.</i>	65
<i>Figure 26. Planar Extraction Results.</i>	67
<i>Figure 27. Planar Extraction Process[10].....</i>	67
<i>Figure 28. Line Extraction Process Performed on a 2D Intensity Image</i>	70
<i>Figure 29. Different Edge Detectors Performed on a 2D Intensity Image.....</i>	70
<i>Figure 30. Line Extraction Results from a 2D Digital Image</i>	71
<i>Figure 31. Split Procedure of the Split and Merge Method.</i>	72
<i>Figure 32. Merge Procedure of the Split and Merge Method.</i>	74
<i>Figure 33. 9 Point Segment Test for FAST Corner Extraction. (adapted from [41])</i>	76
<i>Figure 34. FAST Corner Extraction Performed on 2D Digital Image.....</i>	76

<i>Figure 35. SIFT Point Extraction Performed on 2D Digital Image.</i>	78
<i>Figure 36. Chosen Associated Point Features from SIFT algorithm.</i>	78
<i>Figure 37. Image Translation and Rotation.</i>	81
<i>Figure 38. Association Between Two Frames.</i>	83
<i>Figure 39. Results of 3D Feature Association.</i>	85
<i>Figure 40. 2D Multi-View Line and Point Geometry [38]</i>	87
<i>Figure 41. Results of 2D Feature Association.</i>	89
<i>Figure 42. Probability Density Functions of Association and Miss-association</i>	90
<i>Figure 43. Extended Kalman Filter Mechanization</i>	108
<i>Figure 44. Extended Kalman Filter Mechanization Involving Constraint Equations....</i>	108
<i>Figure 45. Environment for Navigation Simulation</i>	113
<i>Figure 46. Error Free Attitude Change Data</i>	113
<i>Figure 47. Error Free Position Change Data</i>	114
<i>Figure 48. Results of Un-aided Tactical Grade INS.</i>	121
<i>Figure 49. Results of Un-aided Commercial Grade INS.</i>	121
<i>Figure 50. Results of Experiment 1.</i>	122
<i>Figure 51. Results of Experiment 2.</i>	123
<i>Figure 52. Results of Experiment 3.</i>	124
<i>Figure 53. Results of Experiment 4.</i>	125
<i>Figure 54. Results of Experiment 5.</i>	125
<i>Figure 55. Results of Experiment 6.</i>	127
<i>Figure 56. Results of Experiment 7.</i>	128
<i>Figure 57. Results of Experiment 8.</i>	128
<i>Figure 58. Results of Experiment 9.</i>	129
<i>Figure 59. Results of Experiment 10.</i>	131
<i>Figure 60. Results of Experiment 11.</i>	131
<i>Figure 61. Results of Experiment 12.</i>	132
<i>Figure 62. Results of Experiment 13.</i>	132
<i>Figure 63. Results of Experiment 14.</i>	133
<i>Figure 64. Results of Experiment 15.</i>	134
<i>Figure 65. Results of Experiment 16.</i>	135
<i>Figure 66. Results of Experiment 17.</i>	135
<i>Figure 67. Results of Experiment 18.</i>	137
<i>Figure 68. Results of Experiment 19.</i>	137
<i>Figure 69. Results of Experiment 20.</i>	138
<i>Figure 70. Results of Experiment 21.</i>	139
<i>Figure 71. Results of Experiment 22.</i>	140
<i>Figure 72. Results of Experiment 23.</i>	140
<i>Figure 73. Results of Experiment 24.</i>	142

<i>Figure 74. Results of Experiment 25</i>	143
--	-----

LIST OF ACRONYMS

UAV	Unmanned Aerial Vehicle
MAV	Manned Aerial Vehicle
IGV	Intelligent Ground Vehicle
GPS	Global Positioning System
IMU	Inertial Measurement Unit
EKF	Extended Kalman Filter
INS	Inertial Navigation System
PNAT	Precision Navigation, Attitude, and Time
GNSS	Global Navigation Satellite Systems
RF	Radio Frequency
DFD	Depth From Defocus
SFM	Structure From Motion
MFS	Motion From Structure
SLAM	Simultaneous Localization and Mapping
SIFT	Scale Invariant Feature Transforms
DCM	Direction Cosine Matrix
TOF	Time of Flight
MEMS	Micro Electromechanical System
FPGA	Field Programmable Gate Array
DWT	Discrete Wavelet Transform
SWT	Stationary Wavelet Transform

CHAPTER 1: INTRODUCTION

To enable operation of intelligent unmanned and manned vehicles such as UAV's, and IGV's a precision navigation, and attitude capability is required that can operate anytime and in any environment. A popular, reliable and accurate Precision Navigation, Attitude, and Time (PNAT) method is the integration of GPS data with IMU measurements. This integration is a viable solution for most operational scenarios, but is highly dependent on the reception of good GPS signals, which may not be attainable in certain navigation scenarios, such as navigating in heavy urban environments, underground, or indoors. Also, a system that is heavily dependent on exterior signals, may be susceptible to possible jamming or denial [1]. Since GPS is dependent on RF signals transmitted by the GPS satellites, the jamming and denial scenario is real, and there may be situations in which it can't provide the navigation solution required for on-going operation. Therefore, another system is needed that is not dependent on GPS or any external signals to have good reliability anytime, anywhere.

This thesis presents a complementary navigation technique to GPS/INS specifically for indoor and urban environments. The basic philosophy behind this approach is illustrated in Figure 1. It involves integrating range information from a 3D imager, intensity information from a 2D image, 2D digital camera data, and raw 'delta-v' and 'delta- θ ' measurements from an IMU. The IMU is chosen as the core sensor since it is self-contained, and therefore, does not rely on an external RF signal infrastructure. However, position and attitude estimates derived from the IMU measurements (Inertial Navigation System, INS) will drift over time. Integration with secondary sensors is,

therefore, required to mitigate this by periodically “resetting” the inertial errors. The navigation method proposed in this thesis, extends the solution discussed in [2], by adding the data from the 2D images into the solution. Combining the data of multiple types of sensors as well as considering multiple feature types should not only improve the accuracy of the position and attitude estimates, but also add integrity, continuity, and availability to the solution.

Figure 1. Sensor Integration Philosophy.

In accordance with the solution discussed in [2], it is crucial to be able to detect and track features from one time epoch to the next. In this thesis the sensor data used in the feature tracking process is presented in the form of 3D point cloud data, 2D intensity data, and 2D digital imaging data. An example of features being extracted from the 2D digital camera data can be seen in Figure 2. Also, Table 1 lists the measurements available for this work, the sensors from which the measurements originated, and the features that can be extracted from the sensor data.

Figure 2. Example Features Shown in a 2DDigital Image.

Table 1. Measurement and Features Associated with the Sensors.

Measurements		Features
2D Vision Camera	Intensity	Edges[3],[4], corners[5], lines[6], planar surfaces, SIFT features[7], SURF features[8], etc.
2D Laser Scanner	Point cloud (2D)	Lines[9], corners, point features
3D Imaging Camera	Point cloud (3D), Intensity	Planar surfaces[10], curved surfaces, point features

The integration method discussed in this thesis is described by the block diagram in Figure 3. First, 2D vision camera images(referred to as the 2D digital image) are captured and the desired features are extracted. Simultaneously, the 3D imager collects range and intensity measurements which are converted to a point cloud (referred to as the 3D point cloud) representing the 3D environment and a 2D monochromatic image (referred to as the 2D intensity image). Feature extraction is then performed on the 3D point cloud, the 2D intensity image, and the 2D digital image. After the feature extraction process is completed, the association stage begins. During this stage features are tracked from one frame to the next. Each extracted feature (i.e. 2D point, 3D point, 2D corner, 3D corner, 2D line, 3D line and 3D planar surface) is represented by a feature vector. In

the association stage the feature vector from the previous time epoch is propagated to the current time epoch and compared to the current feature vector. Successful association is based on the “distance” between the feature vectors in N-dimensional feature space as long as this distance is smaller than a predetermined threshold (“gate”). For feature propagation, the algorithm uses either the sensor data itself (in case of limited motion) or the outputs of the INS mechanization at both time epochs. In other words, the algorithm takes into consideration the estimate of the sensor motion since the last frame, which comes from the INS data. If an extracted feature corresponds to where this calculation estimates it should be, then it is stored along with all of the other features from the current frame that associated with previously seen features.

The next step in the algorithm is to transform all associated 2D features into 3D body-frame coordinates. Once the current associated feature data is represented in 3D coordinates, it is possible to compare it to the corresponding previously observed features, and used to estimate the drift errors in the inertial sensor. The tightly coupled integration is performed using a complementary extended Kalman filter. The outputs of this Kalman filter are estimates of the errors contained in the INS. When these errors are removed by feeding them back to the inertial mechanization [11], a position, velocity, and attitude estimate is found that drifts at a significantly slower rate than the INS alone.

Figure 3. 3D, 2D, and IMU Integration Approach

The following chapters can be divided into four main sections. The first section, consisting of Chapters 2 and 3, discusses important background information regarding the research of others relating to this topic, as well as important information on the sensors that have been used in the data collections. The second section, Chapter 4, describes implemented image processing techniques and feature association algorithms. The results of these processes performed on real data are displayed and discussed in this chapter. Chapters 5 and 6 are encompassed in the third section of this thesis. Chapter 5, derives all of the necessary navigation equations to implement the navigation portion of the process shown in Figure 3. To theoretically show that the equations established in chapter 5 have the capability of yielding the desired results discussed above, a simple simulation was created in which all of the inputs to the system are known and controllable. This simulation provides the ability to theoretically prove and test the concepts discussed in this thesis without addressing the complexities of dealing with real data. Chapter 6 describes this simulation, as well as multiple experiments executed within it to theoretically show the performance of the discussed system under given

circumstances. The fourth and final section of this thesis includes chapters 7, and 8. These last two chapters discuss the conclusions of this work and suggested future work, respectively.

CHAPTER 2: BACKGROUND

This chapter begins by addressing multiple different navigation systems. With regards to achieving the ultimate goal of having an accurate navigation solution anytime and anywhere, systems are discussed to show their strengths and weakness. By comparing the weaknesses of all of the discussed navigation systems, the need for a system like the one discussed in this thesis becomes apparent. Multiple navigation techniques are also addressed in this chapter. Each of these techniques is mentioned based on either, their direct use in, or similarity to, one or more of the processes used in the navigation system described in this thesis.

2.1 GNSS Navigation Systems

Global Navigation Satellite Systems (GNSS) use radio frequency (RF) signals from orbiting satellites to give receivers the capability of calculating their position globally. The only fully operational GNSS today is the Global Positioning System. This system was developed in 1973, to meet a need for an accurate, reliable system of navigation for the United States military[12]. At that time, two separate projects were started, the “TIMATION” project by the Navy, and “Program 621B” by the Air Force. These two projects, each having different strengths and weaknesses, were combined, utilizing the strongest elements of each. This new system would be known as GPS, and it has revolutionized the field of navigation [12].

The position accuracy that can be achieved with GPS ranges from a couple of meters for the standalone user, centimeters or even millimeters in post-processing mode. The main factor in this variation of accuracy is the resources (architecture) and algorithms used on the receiving end of the system. Examples include receiver capability, antenna capability, and the availability of signals [13]. The GPS constellation, shown in Figure 4, currently gives global coverage with its satellites transmitting signals at two different frequencies: L1 at 1575.42 MHz and L2 at 1227.6 MHz, and new signals such as L5 at 1176.45 MHz are being introduced as we speak. The L1 carrier for each satellite is modulated by two pseudorandom noise (PRN) codes. One of these codes repeats every millisecond and gives each satellite a unique identification [13]. A navigation message is also superimposed on the code, containing the clock parameters and ephemeris data for each individual satellite [13]. The satellite ID with the navigation message makes it possible to decode the GPS signals and acquire a position from them.

Figure 4. GPS Satellite Constellation [14]

GPS has proven that it can reliably yield accurate positioning nearly anywhere in the world, but there exist environments that are challenging for GPS. The signals broadcast by GPS satellites have a specific signal strength, and are attenuated during

propagation through air and other media. Media such as dense foliage, buildings, or the surface of the earth, can cause significant attenuation of the signal or even completely block the signals to the point where it is no longer usable. This can cause navigation in these environments to be impossible due to lack of sufficient signals. The problem of receiving multipath signals is another challenge for GPS receivers. In many cases, this problem can be solved using antennas designed to reduce multipath, or using specifically designed receiver algorithms to reduce the problem, but these can't always be relied upon to fix the problem in all scenarios. An urban canyon is an example of this, in which the multipath from buildings and other objects can be so great that there is no way of truly eliminating it.

2.2 Inertial Navigation Systems

Inertial navigation systems are often referred to as dead reckoning systems. Dead-reckoning systems measure the change in position or the velocity and integrate it to obtain an estimate of the new position. The main disadvantage of this method, however, is that the position uncertainty grows over time as the error is accumulated. This type of navigation is one of the oldest forms used, but it has been only in the last century that there has been a large scientific push to develop and perfect it. Humans have a very crude, and inaccurate inertial system called the Vestibular System, seen below in Figure 5. It consists of the Otolith organs which sense the direction and speed of linear accelerations as well as the current tilt of the head, and the semicircular canals, which sense the direction and speed of angular accelerations which yields pitch, roll, and yaw of

the head [15]. For a human, the information from the Vestibular system yields a rough estimate of how far they have moved and in what direction. The accuracy of the Vestibular system varies from person to person, but in general, it is highly inaccurate, easily confused, and becomes more erroneous the longer it is used by itself.

Figure 5. The Vestibular System [15]

A typical INS has three orthogonal gyroscopes as well as three orthogonal accelerometers. Also, according to [16] there are five functions that an INS must be able to accomplish. The first function is that it must be able to determine the angular motion of a platform using gyroscopic sensors, from which the attitude of the platform can be derived. Next, the system must be able to measure the specific forces being exerted on the sensor using accelerometers. The system must also be able to resolve the specific force measurements into a reference frame using the knowledge of attitude derived from the gyroscopes. The fourth function that an INS must accomplish is that it must evaluate the force resulting from the gravitational field of the mass on which it is operating. Lastly, it is necessary for an INS to integrate the resolved specific force measurements to obtain estimates of the velocity and position of the platform. A navigation system must be capable of accomplishing all of these tasks to be considered an INS.

The evolution of INS into what is currently being used would not have been possible without a few key scientific breakthroughs. In the seventeenth century Sir Isaac Newton defined the laws of mechanics and gravitation. These laws are the foundation on which inertial navigation is based [16]. In 1852, Leon Foucault discovered the gyroscopic effect, and in 1890, Professor G.H. Bryan discovered the principles that would later be used in the creation of solid-state gyroscopes. In the beginning of the twentieth century the development of the gyrocompass occurred, which provided an indication of true north for a directional reference. Also, during this time frame, Professor Max Schuler, derived the technique known as Schuler tuning which gave the ability to create a directional instrument that is unaffected by accelerations. In the 1950s, Professor Charles Draper made multiple contributions to inertial technology, the largest of which was the invention of the floated rate-integrating gyro, which drifted at a significantly lower rate than any gyroscope produced up to that point. Later in the twentieth century the advancements of gyroscope technology and the microcomputer made it possible to create a strapdown inertial system, which is commonly used today for UAV, IGV, and aircraft navigation.

Strapdown inertial systems are rigidly attached to the frame of the platform which is using the INS for navigation. Instead of being gimbaled to always have the same orientation, they provide outputs with respect to a chosen reference frame, which can be mathematically transformed into other reference frames. Since strapdown systems are mechanically simpler than gimbaled systems, they are, currently, the most commonly used type of INS. A block diagram of an example strapdown INS mechanization is shown in Figure 6.

Figure 6. Strapdown Inertial Mechanization [15]

Although inertial systems are very useful, they contain quite a few errors. The biggest of these problems is that since this system has feedback from previous measurements, and these measurements have error, the error accumulates and continuously gets worse if no other sensors are integrated with the INS. Due to this, it is important to remove any errors that can be estimated. Some of these errors include position displacement error, $\delta \mathbf{r}$, velocity error, $\delta \mathbf{v}$, tilt or miss-orientation, Ψ , gyro bias, $\delta \omega$, and accelerometer bias, $\delta \mathbf{f}$. These errors can be related to the input variables of the INS through equations (2.1), (2.2), and (2.3). By implementing these equations, the errors discussed can be taken out through the mechanization shown in Figure 7 if their values can be estimated.

$$\delta \dot{\mathbf{v}}_e^n = [\mathbf{f}^n \times] \Psi + \mathbf{C}_b^n \delta \mathbf{f}^b \quad (2.1)$$

$$\dot{\Psi} = -\omega_{in}^n \times \Psi - \mathbf{C}_b^n \delta \omega_{ib}^b \quad (2.2)$$

$$\hat{\mathbf{C}}_b^n = (\mathbf{I} + [\Psi \times]) \mathbf{C}_b^n \Rightarrow$$

$$\mathbf{C}_b^n = (\mathbf{I} + [\Psi \times])^T \hat{\mathbf{C}}_b^n \Rightarrow$$

$$\mathbf{C}_b^n = (\mathbf{I} - [\Psi \times]) \hat{\mathbf{C}}_b^n \quad (2.3)$$

Figure 7. Inertial Mechanization with Error Compensation (taken from [15])

The problem with any of the errors that cannot be accurately estimated and taken out is compounded when the commonly desired outputs of position and attitude are examined through the mechanization shown in Figure 7. The inputs to this mechanization from a rate-integrating IMU are $\Delta \mathbf{v}$'s, giving the change in velocities from the accelerometers, and the $\Delta \theta$'s, providing the change in attitude angles from the gyroscopes. When inspecting the system inputs, the $\Delta \mathbf{v}$'s seen in equation (2.4) and the $\Delta \theta$'s in equation (2.5), it can be seen that any errors within the original measurements have already been integrated once, adding a linearly growing 't' component to the errors, with 't' representing time. When the position calculation is performed, the $\Delta \theta$'s are integrated twice. Therefore, the errors in the $\Delta \theta$'s are integrated a total of three times, creating a 't³' error component, and dominating the position error. A similar process occurs during the attitude calculation. When this is computed the inputs to the system are

integrated once more, creating a ‘t²’ error component in the output of the attitude. These errors, the position deviating at a rate of ‘t³’ and the attitude at a rate of ‘t²’, are known as inertial drift errors and are typically the largest source of error from inertial systems. It is because of the nature of these drift errors that INSs are considered to be great at short term navigation, but poor for long term applications.

$$\Delta v = \int_{t-\Delta t}^t \mathbf{f}^b dt \quad (2.4)$$

$$\Delta \boldsymbol{\theta} = \int_{t-\Delta t}^t \boldsymbol{\omega}_{ib}^b dt \quad (2.5)$$

2.3 Optical Navigation Systems

The two navigation methods discussed above are used in many navigation applications, however, there are situations where neither are a viable standalone option. As discussed above, there are multiple scenarios where GNSS may not be an acceptable standalone solution due to a challenging operating environment. There are also situations in which INS by itself is not an acceptable solution due to the drift errors which are excessive for long term applications. These two sets of scenarios where either GNSS or INS is not a viable option are not mutually exclusive. Therefore, there are situations in which an alternative navigation solution must be sought if accurate positioning for that environment is desired. Many different solutions have been proposed to alleviate this problem. One of the most promising is optical navigation.

Optical Navigation uses sensors to capture images of the surroundings of the navigation platform [17]. Consecutive image frames are taken and content changes are identified. These changes in image content, in combination with a priori information about the optical sensor used to acquire them, can be used to derive estimates of the user position and attitude changes. The two main methods used to evaluate the image content are via feature tracking and optical flow. Optical flow based techniques have been proven to yield a basic estimation of user motion[18], but feature tracking based methods have shown to provide more accuracy and are the more common choice for navigational purposes and depth resolution[19].

In order to use feature tracking for navigational purposes it is necessary to accurately estimate the depth of each feature. There are different techniques that can be used to estimate the depth of a feature. One technique discussed in [20] is depth from defocus (DFD). This process utilizes the time at which an object comes into focus on the focal plane, in combination with knowledge of the cameras optics and focal length, to estimate the distance to a feature [20]. DFD can yield good depth measurements but, due to the effects of motion blur, it can't be used to accurately detect depth while the camera is moving. Therefore this procedure should not be used for navigation on a moving platform.

Another methodology that is commonly used for feature depth estimation is stereo vision shown in Figure 8. This process takes observations from two separate cameras and compares the pixel locations of common features [21]. Using this information, the disparity in pixel location 'd' can be calculated using equation (2.6). The disparity, in

combination with a known baseline between the cameras 'b', and the focal length 'f', can be used to calculate an estimate of feature depth using equation (2.7). This method provides an accurate estimation of depth, but two cameras must be used that are rigidly attached to one another.

$$d = x_1 - x_2 \quad (2.6) [22]$$

$$\frac{d}{b} = \frac{f}{z} \quad (2.7) [22]$$

Figure 8. Stereo Vision [22]

The process of stereo vision shows that it is not possible to estimate depth with only a single camera, but there are other processes such as structure from motion (SFM) or motion from structure (MFS) that make use of a single camera and another sensor [23]. These processes use consecutive image frames while the camera is moving to create two different images taken at different locations. These images are then treated as if they

were from two different cameras and the difference of pixel locations from common features can be determined, similar to the process used in stereo vision. With the difference of pixel locations of features, and the focal length known, the only information necessary to estimate the depth of a feature is the baseline between where the two frames were observed. In order to estimate this baseline a sensor must be used to measure the movement of the camera between the observations [10]. An IMU is the most common choice for this sensor [10]. The integration of a camera with an IMU gives the possibility of using one camera to estimate depth of features accurately while the platform is moving.

If depth of features can be resolved during the process of feature tracking, it can be a useful navigation technique. Feature tracking takes images from an optical sensor, locates features, and tracks them. These features include planes, lines, points, and corners. This process begins with a method called feature extraction. When an image frame is taken, feature extraction is used to analyze the data and locate any desired features contained in the sensors surroundings. Using specifications of the camera used, in combination with the data from it and feature depth information, it is possible to calculate a position for features with respect to the camera. Feature extraction is performed on each consecutive frame as data is collected. This information about the features is stored. Using the information from the feature extraction process, feature association is executed. This process compares the feature information from the current frame to the previous frame and searches for common features. When a feature is seen in

a frame and can be matched to the same feature in a previous frame, the change in position of that feature can be calculated, illustrated in Figure 9.

Figure 9. Position Change of a Feature

The knowledge of position change of a tracked feature, combined with the amount of time between frames from which the motion was observed, can be used to solve for motion of the sensor platform. If the position change of two non-parallel features is known for a specific frame then the attitude change equation, equation (2.8), can be performed. The position change equation, equation (2.9), can be performed if there are three non-parallel features available. If there are enough features to execute both of these equations, then an accurate estimate of the sensors change in position and change in attitude can be determined.

$$\hat{\mathbf{C}}_{b_k}^{b_{k+1}} = \arg \max_{\mathbf{C}} \left\{ \sum_{i=1}^N \mathbf{C}_{b_k}^{b_{k+1}} \mathbf{n}_i(t_k) \mathbf{n}_i(t_{k+1}) \right\} \quad (2.8)$$

$$\Delta \hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \Delta \mathbf{p} \quad (2.9)$$

Optical Navigation system have been proven to work, but just as the other navigation systems discussed, they have multiple error sources that can cause tremendous difficulties to the system. One major source of error is sensor error. As it is for almost all navigation systems, the accuracy of the sensors being used is directly correlated with the accuracy of the navigational outputs. Another large source of error can occur during the feature tracking algorithm. If the feature extraction or the feature association algorithms do not perform perfectly, they will inject error into the system. When two features are incorrectly associated or a feature that is not actually a feature is extracted, these will likely cause errors in motion estimation that will never be removed unless a relative position is acquired. Associating features incorrectly will cause an optical navigation systems errors to grow rapidly.

2.4 Navigation Techniques

2.4.1 SLAM

Simultaneous localization and mapping (SLAM) is an algorithm commonly used in unknown environments where exterior signals can't be used [24]. SLAM is used by a navigational platform to simultaneously build a map of an unknown area, while calculating its position within the map. This technique uses sensors such as cameras and

laser scanners to give information about the platforms surroundings, in combination with a sensor, such as an IMU, to estimate the motion of the platform.

In order for SLAM to have its desired results, a simultaneous estimate of platform location and feature location is required. This action is demonstrated through Figure 10, showing a mobile platform observing its location relative to the estimated locations of multiple observed features. When the SLAM process begins, the sensors on the platform begin taking observations of the area around them. These observations are analyzed to produce the data needed for this process. For each data frame that is captured, feature extraction, feature association, mapping, and localization, must all be done. The feature extraction process is completed by analyzing each image that is collected to find all usable features in it. When a feature is found, feature association is performed on it. This process compares the currently observed feature to all previously observed features to determine whether that feature had been seen before. After feature extraction and feature association are completed, the observed data is added to the continuously growing map. Concurrently, localization of the observed features and the sensor platform are done within the produced map. This procedure is completed on every set of data that is captured by the sensors.

Figure 10. SLAM Representation (adapted from [24])

In Figure 10, it can be observed that there are errors between the estimated and true locations of observed features. A large part of these errors are common among all feature locations. From this it can be deduced that these estimates are highly correlated [24]. This means that the locations of these features relative to each other are much more accurate than their locations within the map[24]. More importantly, these correlations increase as more features are found [24]. Therefore the relative locations of features always improves as more features are observed. With an increase in relative accuracy coming directly from increasing the number of features observed, this process can also benefit greatly from executing feature association. If a feature is seen two or more times, the location of that feature becomes more accurately known relative to other features as well as within the map. Obtaining multiple measurements of a single feature increases knowledge of that feature, and therefore increases accuracy and decreases uncertainty. With this information it can be stated that as time proceeds and more measurements are taken, overall accuracy will only increase. This is due to a reduction in the errors affected

by this algorithm, encompassing all errors except measurement error. Therefore, it can be stated that assuming adequate amounts of run time and computational power, the accuracy of SLAM is mainly dependent on the quality of the sensors being used.

2.4.2 *SIFT*

Many different methods have been developed for the processing of feature extraction and feature association. One of the most common is the use of Scale Invariant Feature Transforms (SIFT). This process was derived by David Lowe with the intent using local features that are invariant to translation, rotation, and image scaling [25]. The use of this class of features makes SIFT a very powerful and robust way of optically tracking features.

To identify desired features, SIFT uses a multiple staged approach. The first stage is to identify key locations in an images scale space. Key locations are chosen by attempting to identify places in an image that are invariant to translation, rotation, and scaling. This is done in [25] by computing a difference of Gaussian function in scale space on an image. The results of this function are analyzed, and key locations are identified to be at maxima and minima. These maxima and minima are inherently located at places in the image with high variations. The higher these variations, the more stable the locations are for identification in future images with large amounts of translation, rotation, and scaling. Each key location yields a feature vector, known as a SIFT key, to describe its sampled location with respect to its scale-space coordinate frame. The locations of these SIFT keys are also characterized at each pixel, by their calculated

gradient and orientation. In [25], this is done at each pixel, $A_{i,j}$, using equation (2.10) to calculate the gradient of the SIFT key, and equation (2.11) to calculate its orientation. This characterization helps improve the stability of the SIFT keys [25].

$$M_{i,j} = \sqrt{(A_{i,j} - A_{i+1,j})^2 + (A_{i,j} - A_{i,j+1})^2} \quad (2.10) [25]$$

$$R_{i,j} = \text{atan2}(A_{i,j} - A_{i+1,j}, A_{i,j+1} - A_{i,j}) \quad (2.11) [25]$$

Using these SIFT keys, the local image region can be described in a way that is invariant to translation, rotation, and scaling of the image, assuming that each key has a stable orientation, location, and scale [25]. This is done by including information in the SIFT key vector about the key point as well as the pixels around it. This process opens a circle with a radius of a set number of pixels around the key location, and computes information about every pixel in that circle relative to the key. In [25], this process results in each key vector containing 160 elements, yielding enough knowledge about each SIFT key to make it highly unique and specific [25].

The last steps of SIFT are done to match up keys from one frame to the next as well as use that information to calculate the translation, rotation and scale differences from the previous image to the current image. There are many different ways of matching keys from one frame to the next, with each one having a trade-off between computational intensity and accuracy. In [25], two different methods are explained. The first method is performing a close to exhaustive search, which goes through nearly every possibility for matching the SIFT keys and selects the best one. This gives the exact solution almost

every time but, it is also very computationally intensive. If computational power and time are not issues, this is the best possible solution, but in most cases these are limiting factors. Therefore, other methods must be explored in order to significantly cut down on needed computations while still achieving acceptable levels of accuracy in the results. An attempt to do this is described in the second method in [25], referred to as the best-bin-first approach. This approach is described in [26] as a modified kd-tree algorithm attempting to locate the nearest-neighbor in an image to a previously seen SIFT key. For each estimated nearest neighbor, a probability of correctness is associated with it depending on is the quality of the match. These probabilities are then used to weigh the estimated matches from the previous image to the current image and a solution can be determined. This procedure provides a solution with high accuracy and a dramatic cut in computations needed. Once a matching solution is determined, the locations of previous SIFT keys (x,y) with the locations of their estimated matched keys in the current image (u,v), can be used to solve equation (2.12). Given at least three matching SIFT keys between two images this equation can be used to solve for translation 't', as well as the rotation, scale, and stretch parameters contained in 'm'[25].

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix} \quad (2.12) [25]$$

(note: this shows only one SIFT key match, each additional match adds two rows to the first and last matrix)

2.4.3 Frame Rotation

Any motion of a rigid body can be expressed mathematically using a translational motion with a rotation. The equations used to define rotations are especially useful in fields such as inertial navigation for rotation between frames. Frame rotation is generally based on roll pitch and yaw parameters. For example, the rotational motion of an aircraft can be described using a 3D axis in the body frame in which the positive x-axis is pointing out of the nose, the positive y-axis is pointing out of the right wing, and the positive z-axis is pointing out of the bottom of the aircraft. Using this frame we can quantify that any rotation about the z-axis can be described using equation (2.13) where ψ is the amount of yaw seen by the aircraft. Similarly, rotation around the y-axis is described in equation (2.14) with θ equal to the amount of pitch and, ϕ in equation (2.15) describes the amount of roll seen by rotation around the x-axis. These can be individually used if there is rotation around only one axis, but in most cases there is at least some rotation in multiple axes. Therefore, equation (2.16) is used to derive equation (2.17) to encompass rotation in all three axes using one matrix. This rotation matrix is referred to as a direction cosine matrix (DCM) and, is the fundamental equation used in frame rotation.

$$\mathbf{C}_1 = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.13) [15]$$

$$\mathbf{C}_2 = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix} \quad (2.14) [15]$$

$$\mathbf{C}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & \sin \varphi \\ 0 & -\sin \varphi & \cos \varphi \end{bmatrix} \quad (2.15) [15]$$

$$\mathbf{C} = [\mathbf{C}_3 \mathbf{C}_2 \mathbf{C}_1]^T \quad (2.16) [15]$$

$$\mathbf{C} = \begin{bmatrix} \cos \theta \cos \psi & -\cos \varphi \sin \psi + \sin \varphi \sin \theta \cos \psi & \sin \varphi \sin \psi + \cos \varphi \sin \theta \cos \psi \\ \cos \theta \sin \psi & \cos \varphi \cos \psi + \sin \varphi \sin \theta \sin \psi & -\sin \varphi \cos \psi + \cos \varphi \sin \theta \sin \psi \\ -\sin \theta & \sin \varphi \cos \theta & \cos \varphi \cos \theta \end{bmatrix} \quad (2.17) [15]$$

2.4.4 Kalman Filtering

In 1960, Rudolf Emil Kalman published a paper describing a recursive solution to discrete linear filtering problems [27]. It is an optimal data processing algorithm which uses stochastic models to estimate the solution of a linear differential equation [28]. This filter is typically used to combine two complementary sensors to exploit the strengths of both and minimize their weaknesses. The most common example of this is to combine INS data, which is highly accurate for short periods of time, with GNSS data, which always contains error but, does not degrade due to its operation time increasing. The aim of the Kalman filter for this case is to combine the advantages of the short term accuracy of the INS with the long term consistency of the GNSS.

2.4.4.1 Linear Kalman Filter

In its linear form, the Kalman Filter linearizes its equations around a normal trajectory that does not depend on measurement data. Laying out this process begins with assuming the system can be modeled with two equations. The first is the general system model equation. It is used to estimate the state vector, $\mathbf{x}_k$, and is shown in equation (2.18). The second equation, displayed in equation (2.19), is the general measurement model equation. This is used to estimate the

measurement vector, $\mathbf{z}_k$. The accuracy at which the linear Kalman filter operates depends heavily on how well it can estimate the random process for which these equations are modeled after[29].

$$\mathbf{x}_k = \mathbf{\Phi}_{k-1}\mathbf{x}_{k-1} + \mathbf{w}_k \quad (2.18)[29]$$

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \quad (2.19)[29]$$

In order for the two process model equations to estimate the characteristics of the system being modeled, the system's expected dynamics must be modeled. The more accurate the system can be modeled, the more accurate equations (2.18) and (2.19) can estimate the process. The process noise, w_k , the error variance, e_k , and the measurement error variance, v_k , must be estimated and are assumed to be Gaussian due to the central limit theorem. The measurement matrix, $\mathbf{H}_k$, and the system dynamics matrix, $\mathbf{F}$, must also be modeled given the expected use of the system and its inputs. Using these variables, an initial measurement covariance matrix, $\mathbf{R}_k$, an initial error covariance matrix, $\mathbf{P}_k$, a transition matrix, $\mathbf{\Phi}$, and a system noise covariance matrix, $\mathbf{Q}$, can be obtained using equations (2.20), (2.21), (2.22), and (2.23) respectively [29].

$$\mathbf{R}_k = E \left[\mathbf{v}_k^- \mathbf{v}_k^{-T} \right] \quad (2.20)[29]$$

$$\mathbf{P}_k = E \left[\mathbf{e}_k^- \mathbf{e}_k^{-T} \right] \quad (2.21)[29]$$

$$\mathbf{\Phi}_k = \exp^{\mathbf{F}\Delta t} \quad (2.22)[29]$$

$$\mathbf{Q} = E \left[\mathbf{w}_k^- \mathbf{w}_k^{-T} \right] \quad (2.23)[29]$$

As the system is in use, a set of optimized update and prediction equations are used in an iterative fashion to solve for the parameters necessary for the system to estimate its current state. The first update equation in the process computes the Kalman gain, $\mathbf{K}$. For each time step in the iterative process, the Kalman gain is calculated to minimize a specific type of error in the system. For this paper, the Kalman gain will be computed using equation (2.24), which will minimize the mean-square estimation error [27]. The next update equation, is used to estimate the state vector using the current measurements. This is shown in equation (2.25), where $\hat{\mathbf{x}}_k$ denotes the estimate of the current state vector, using the '^' notation to indicate an estimate. The error covariance matrix must also be updated in this process. Equation (2.26) shows how this matrix is updated. The first of the prediction equations, shown in equation (2.27), is used to estimate the next state vector using the current estimated state vector and the calculated transition matrix. The last of the prediction equations, shown in equation (2.28), projects the error covariance matrix forward. At each time epoch in which the filter is used, these equations go through one iteration, and their outputs are used to solve for the system's estimated state[29].

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R})^{-1} \quad (2.24)[29]$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H} \hat{\mathbf{x}}_k^-) \quad (2.25)[29]$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_k^- \quad (2.26)[29]$$

$$\hat{\mathbf{x}}_{k+1}^- = \Phi_k \hat{\mathbf{x}}_k \quad (2.27)[29]$$

$$\mathbf{P}_{k+1}^- = \Phi_k \mathbf{P}_k \Phi_k^T + \mathbf{Q} \quad (2.28)[29]$$

2.4.4.2 Extended Kalman Filter

In many situations, the previously described linear Kalman filter is an appropriate choice, but when a problem cannot be described accurately using a linear system model, a nonlinear process must be used. The nonlinear process developed for this case is known as extended Kalman filtering. This process linearizes around a trajectory in a similar fashion to the linear Kalman filter. Contrarily, the state estimates used to update this trajectory are derived from the observed measurements, unlike the linear Kalman filter which does not use measurement data to update its state estimates[28].

The process of implementing an extended Kalman filter is similar to the process laid out above for the linear Kalman filter with a few changes. The two process models shown in equations (2.18) and (2.19) remain the same, meaning that it is still assumed that these functions are a valid estimate for the random process of the system. The modeling process is done using all of the same equations and methods except for the derivation of the geometry matrix. $\mathbf{H}$ is no longer expressed as a constant term that is calculated once for the entire filter's use. In the extended Kalman filter, the $\mathbf{H}$ matrix is derived using equation (2.29). In this equation the Jacobian of the observation matrix $\mathbf{h}$, is taken with respect to the state vector. This equation becomes the first update equation in the iterative process discussed for the linear Kalman filter. Using the continuously

updated geometry matrix, the rest of the update and prediction equations remain unchanged[29].

$$\mathbf{H}_k = \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}} \big|_{\mathbf{x}=\hat{\mathbf{x}}_k^-} \quad (2.29)[29]$$

Although the linear and extended Kalman filters are designed in a robust manner, they are still susceptible to certain errors which may cause the filter to diverge. A few of these errors, such as miss-modeling, having the wrong uncertainties or, having numerical problems can be detected by monitoring the systems innovations, $\mathbf{I}$. These innovations are derived using equation (2.30), and should contain no information, meaning they are statistically white. While evaluating the innovations, if they do not appear to be white, it may be an indicator that something is wrong with the system. Monitoring these innovations is an easy and effective way to check for certain errors in the system [29].

$$\mathbf{I} = \mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^- \quad \text{or} \quad \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-) \quad (2.30)[29]$$

2.4.4.3 Kalman Filter Coupling When data is being acquired from two complementary sources, their information can be coupled in multiple ways. For the purpose of this paper, the two types of data that will be combined are INS data and platform position and attitude change, which will be estimated by tracking the motion of observed features in the sensors environment. The three most accepted and used styles of coupling are referred to as loose, tight and deep (also known as ultra-tight) integration.

A loose integration style encompasses the least complex architecture of the three types[30]. This style combines the INS with the platform position change estimates at the position level[29]. The INS output and the position change estimates are combined as two independent navigation solutions to create a combined solution as the output. One of the key advantages to this type of integration is the increased redundancy based on the use of two independent solutions [30].

In a tightly coupled filter, the data that is integrated together comes directly from the sensors. Unlike the loosely coupled filter, which uses the navigation solutions of the two types of data, the tightly coupled solution integrates the data at the measurement level[29]. Therefore, this filter calculates one navigation solution based on the gyroscope, accelerometer, and feature position change measurements[30]. Tight integration has a more complex architecture than a loose integration strategy, but it generally provides a more accurate solution. Another important advantage lies in the fact that when the motion of too few features is observed to solve the change in attitude and change in position, equations (2.8) and (2.9) respectively, the INS measurements can be used to supplement this lack of measurements and obtain an accurate solution [30]. If too few measurements are observed for solving the change in position and attitude equations using a loosely coupled system, these equations cannot be solved and therefore, the INS must be used on its own with no aid from the sensors.

The last integration strategy is using a deeply coupled approach. This approach consists of the most complex system architecture of the three styles, but excels in systems with high dynamics. This advantage stems from the use of INS data to assist in the tracking

loops of the system tracking feature motion. The use of the INS data to improve the tracking loops places the INS in the systems architecture in a position where the feature tracking system is dependent on the INS. Therefore, a poor INS has a great effect the feature motion tracking, which in turn will seriously degrade the entire system[30]. Each of these three integration strategies provides a set of advantages and disadvantages which must be taken into consideration when choosing which style to use for any given system. For this research, a tightly coupled integration method will be used based on the measurements that are available to us, and the dynamics of our data collections.

CHAPTER 3: SENSORS AND DATA SETUP

This chapter begins by discussing the three main sensors used in the scope of this thesis, the SwissRanger SR-3000 camera, the Point Grey Firefly, and the Honeywell HG1930. After describing each sensor, the data collection platform and the environment chosen to collect data in are described in detail. The details of performing a necessary camera calibration along with the tools that it provides are then discussed. The last section of this chapter describes the processes used to combine the collected data from the different sensors with respect to the time it was collected.

3.1 CSEM SwissRanger SR-3000 Camera

The Swiss Center for Electronics and Microtechnology (CSEM), designed the SR-3000 camera, shown in Figure 11, to provide a low latency 3D imaging solution. This solid state device consists of an array of 55 eye safe near-infrared diodes which emit light at a wavelength of 850nm [10]. The average illumination power of this sensor is 1 Watt, and it has a field of view of 47.5x39.6 degrees[32]. When collecting data, the light sources are modulated by a sinusoidal signal at a frequency of 20MHz, yielding a 15m unambiguous range resolution. This signal is used for range determination using the time of flight (TOF) principle, calculating the range to an object, R by evaluating the following equation:

$$TOF = \frac{2R}{c} \tag{3.1}$$

The TOF principle uses phase measurements to accurately obtain the amount of time it takes for the signal to leave the sensor, be reflected off an object, and return back to the sensor. This quantity of time is referred to as TOF. Once the range of a pixel is calculated, a 3D point cloud can be constructed depicting the surroundings of the camera in Cartesian coordinates. Along with the 3D point cloud, expressed in x, y, z body frame coordinates; the SR-3000 collects intensity data which can be used to show the image that the sensor is observing in a 2D grey scale format with resolution of 176x144 pixels, also known as Quarter Common Intermediate Format(QCIF). Therefore, the two outputs of this active camera are a 3D point cloud with no depth uncertainty and a low resolution 2D grey scale image [32].

Figure 11. CSEM SwissRanger SR-3000 Camera.

3.2 Point Grey Firefly

The Point Grey Firefly MV, displayed in Figure 12, is a passive sensor that delivers 2D digital images in either color or black-and-white. It operates using less than one watt of power and can capture data at a rate of up to 60 frames per second(FPS). The

standard resolution is 640x480 pixels, but the camera has up to 752x480 active pixels. It has a near infrared quantum efficiency (QE), the likelihood of infrared light to pass through the imaging sensor and be detected, of greater than 35% [33].

The imaging sensor used by the Firefly for transforming the sensor's observed light into electrical signals is a Micron MT9V022 1/3" progressive scan complementary metal oxide semi-conductor(CMOS) [33]. This imaging sensor uses a progressive scan as opposed to the other possibility of an interlaced scan. The largest advantage of a progressive scan compared to an interlaced scan is that it yields better results when capturing images of moving objects[34]. Along with the 3D imagers intensity image, the Firefly MV is used to collect 2D monochrome imagery of the environment. The optical flow and feature movement in the imagery as a function of time will be used to deduce the user motion. Since the target application is a moving rather than a stationary platform, the progressive scan was desirable.

Figure 12. Point Grey Firefly.

3.3 Honeywell HG1930

Honeywell's HG1930 IMU, shown in Figure 13, was designed with the intent of providing tactical grade inertial data for projectiles, smart munitions, missiles, and unmanned vehicles. The HG1930 is a gun hard micro electromechanical system(MEMS) and has a mean time before failure(MTBF) of greater than 20,000 hours[35]. This system weighs less than .35 pounds and consumes less than 3 Watts of power.

This IMU is a hermetically sealed package containing three MEMS accelerometers, three MEMS gyroscopes, and an ARM7 microprocessor. Each of the accelerometers used has an input range of $\pm 30g$ and a bias of 4mg for 1 standard deviation. The three MEMS gyroscopes have an input range of ± 1440 deg/sec and an angular rate bias with a 20 deg/hr for 1 standard deviation. This IMU can output data at a rate of 2,400 Hz and has an angular random walk error of maximum 0.15deg/hr [35].

Figure 13. Honeywell HG1930 [36]

3.4 Sensor Setup

To evaluate the performance of the methods discussed in this thesis, a test setup was developed. The first piece to this setup is the data collection platform, which can be

seen in Figure 14. This figure shows all of the sensors used in the data collections and their placement on the designed platform. These sensors and their interconnections are shown in the block diagram in Figure 15. The data collection processor, an Apple Mac Mini, is the central processor unit (CPU) shown in this figure. Connected to this CPU is the SwissRanger SR3000 3D imaging camera through USB, the Point Grey Firefly 2D imaging camera via Firewire, and the Honeywell HG1930 IMU through a custom Field Programmable Gate Array (FPGA) interface. There are also two Hokuyo laser scanners connected to the CPU via USB, but their data is not used in the scope of this thesis. As each of the individual sensors collect data, they are time-tagged with the local CPU time.

The other part of the test setup is the environment in which data was collected. A hallway in Stocker Center, the Ohio University engineering building, was chosen and can be seen in Figure 16. This hallway was chosen because it provides a "feature rich" environment for the sensors to collect data. In order to use this environment as a means of testing the performance of the algorithms discussed in this thesis, the chosen hallway was manually mapped out using laser range finders to obtain accurate dimensions of the environment.

Figure 14. Data Collection Platform

Figure 15. Data Collection Platform Connection Block Diagram

Figure 16. Data Collection Environment in Stocker Center(Ohio University Engineering Building)

3.5 Camera Calibration

When integrating information from multiple sensors on a single platform it is important to relate all the measurements to a common reference frame. To do so with 2D optical data it is important to perform an accurate calibration of the specified camera. This calibration consists of the estimation of the parameters necessary to form a projection from image frame coordinates to other types of coordinates. For the purpose

of this thesis, the transformation considered will be between the 2D image frame coordinates and 3D navigation (or world) frame coordinates and vice versa[37].

When collecting images using a camera, both the values of the intrinsic and extrinsic camera parameters must be solved. The intrinsic parameters encompass all internal properties of the camera necessary to establish the geometry of how images are captured by the specific camera, and the extrinsic parameters describe the necessary mathematics to transform the image from one reference frame to another. For simplicity, the equations will first be laid out assuming a camera with an ideal perspective, after which we will consider a non-ideal perspective camera. The extrinsic parameters are the parameters that define a rotation, $\mathbf{C}_n^b$, which is in the form of a direction cosine rotation matrix similar to one shown previously in equation (2.17), and the translation, $\Delta\mathbf{x}$, which encompasses the translational motion in the X, Y, and Z-directions. Using equation (3.2), a generic point in the navigation frame given by 3x1 vector $\mathbf{p}_i^n$ can be projected to the corresponding point $\mathbf{p}_i^b$ in the image frame [38].

$$\mathbf{p}_i^b = \mathbf{C}_n^b \mathbf{p}_i^n + \Delta\mathbf{x}^n \rightarrow \mathbf{p}_i^b = \begin{bmatrix} \mathbf{C}_n^b & \Delta\mathbf{x}^n \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i^n \\ 1 \end{bmatrix} \quad (3.2)$$

For the purpose of transforming the point $\mathbf{p}_i^b$ in the image frame, from the 3D coordinates used in equation (3.2) to 2D coordinates in the image frame, we must first define the intrinsic parameters via the standard projection matrix $\mathbf{\Pi}_0$ shown in equation (3.3) and the camera matrix for an ideal perspective camera, $\mathbf{K}_I$, shown in equation (3.4), where f is the focal length of the camera [38].

$$\mathbf{\Pi}_0 = [\mathbf{I}, 0] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (3.3)$$

$$\mathbf{K}_I = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.4)$$

Defining the 2D image coordinates by variables x and y , furthermore, defining a new variable λ as any arbitrary positive scalar. Combining equations (3.2), (3.3), and (3.4), we can form equation (3.5), which is the mathematical representation of the geometric model of an ideal perspective camera for projection between 3D navigation coordinates and 2D image coordinates [38].

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{C}_n^b & \Delta \mathbf{x}^n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i^n \\ 1 \end{bmatrix} \rightarrow$$

$$\lambda \mathbf{x}_{camera} = \mathbf{K}_I \mathbf{\Pi}_0 \begin{bmatrix} \mathbf{C}_n^b & \Delta \mathbf{x}^n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i^n \\ 1 \end{bmatrix} \quad (3.5)$$

Equation (3.5) shows the mathematical representation of the projection necessary to perform a transformation between the 3D navigation frame and the 2D image frame assuming that the camera used has an ideal perspective. If this is not the case more of the intrinsic parameters must be taken into account. The first intrinsic parameter introduced for this case is the principle point (or the center offset). This point (o_x, o_y) is the place at which the z axis of the camera frame intersects the image plane. The next intrinsic parameters introduced are the scale factors s_x and s_y . These scale factors are used to describe the horizontal and vertical lengths of the image, respectively. The last intrinsic parameter is the skew-factor s_θ , which describes the observed angle between the x and y

axes of the image frame. Using these intrinsic parameters, derived through camera calibration, equation (3.6) can be derived to describe, $\mathbf{K}_s$, which is what the previously derived ideal perspective camera matrix, $\mathbf{K}_I$, is missing in order to encompass a non-ideal perspective camera.

$$\mathbf{K}_s = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.6)$$

Combining equations (3.4) and (3.6), a new matrix $\mathbf{K}$ is derived in equation (3.7). This new matrix is known as the calibration matrix and is the main tool that is derived in the calibration process.

$$\mathbf{K} = \mathbf{K}_s \mathbf{K}_I = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.7)$$

With the derivation of the calibration matrix including the variables necessary to deal with a non-ideal perspective camera, equation (3.5) can be re-written in the form shown in equation (3.8), which shows the full form of the mathematical model needed for the projection between the 3D navigation frame and the 2D image frame.

$$\begin{aligned} \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} &= \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{C}_n^b & \Delta \mathbf{x}^n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i^n \\ 1 \end{bmatrix} \rightarrow \\ \lambda \mathbf{x}_{camera} &= \mathbf{K} \Pi_0 \begin{bmatrix} \mathbf{C}_n^b & \Delta \mathbf{x}^n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i^n \\ 1 \end{bmatrix} \end{aligned} \quad (3.8)$$

The calibration used for the research described in this thesis was done using the "Camera Calibration Toolbox for Matlab", developed by Jean-Yves Bouguet at the California Institute of Technology. This toolbox was used to derive all of the intrinsic parameters of the 2D Point Grey Firefly camera. For more information regarding this toolbox the reader is referred to [39]. In order to obtain the data necessary to perform the camera calibration using this toolbox, a data collection scenario was set up as shown in Figure 17. The camera was directed at a custom made calibration board shown in Figure 18. Images were taken of this board at different distances from the camera, while a laser range finder was used to obtain the true distance to the board as a reference. The images from the Firefly, an example of which can be seen in Figure 19, were then sent to the calibration toolbox to complete the necessary calibration of the camera. The results of this process provided the calibration matrix shown in equation (3.9), with all values having units in pixels.

$$\mathbf{K} = \begin{bmatrix} 977.7534 & 0 & 315.5037 \\ 0 & 979.9607 & 240.2856 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.9)$$

Figure 17. Data Collection Scenario for Camera Calibration

Figure 18. Camera Calibration Board

Figure 19. Firefly Image Used for Camera Calibration

3.6 Data Collection and Matching

Using the sensor platform shown previously in Figure 14, data was collected using the SwissRanger SR-3000 3D imager, the Point Grey Firefly 2D imager, and the Honeywell HG1930 IMU. The data from these sensors were stored and time-tagged with the local CPU time as the collection progressed. The IMU data was output at rate of ~90 Hz, the 3D imager data was output at a rate of ~5 Hz, and the 2D imager provided data at ~15 Hz.

Since these sensors don't have their clocks synchronized and furthermore don't capture data at the same time-epoch, their time-tags must be matched and interpolation on

the data may be performed to line the data up. In order to accomplish this task, the sensor that provides data at the slowest rate will be used as the base data set to determine the measurement update rate of the Kalman filter. The timing of the 3D imaging data will, therefore, form the basis for the filter update and other preprocessing steps.

Based on the use of the different types of data in the system described in this thesis, the 2D imager's data will be matched up using a different process than that of the IMU data. The data from the 2D imager will only be used when an image is captured at a time nearly simultaneous to that of the 3D imager capturing data. *In an operational implementation, the times of image capture will be synchronized using the external time input of these sensors. In case of the FireFly MV this capability is already present, however, the SR3000 we have in possession does not have a time synchronization capability.* Using equation (3.10), the closest 2D imager's time-tag, $tt(x)_{2D \text{ imager}}$, to a specific time tag from the 3D imager, $tt(x)_{3D \text{ imager}}$, can be found by examining a vector of time tags from the 2D imager, $\mathbf{tt}_{2D \text{ imager}}$. This process can be visualized as shown in Figure 20. From the actual values in Figure 20, it can be seen that some of the time tags from the 3D imager match up very closely with a corresponding time tag from the 2D imager, but there are also some instances where the 3D imager's time-tags are evenly spaced between two of the 2D imager's time tags. Since interpolating the 2D images is possible but very complex, it was decided that the closest 2D imager's time-tag to each time-tag from the 3D imager must be used even if another time tag from the 2D imager has a similar difference to the 3D imager's time tag. *Again, in an operational setup, interpolation could be performed in the feature domain; the features would be extracted*

and the location of the features would be interpolated to the correct time epoch. This causes some of the 2D matched data to correlate better with the 3D data than others. So, in our simplified approach, the smaller the difference in time-tags between the 3D image and the corresponding 2D image, the better the correlation. This inherently increases the accuracy. Depending on the system dynamics, the difference between time-tags in this process can create serious problems. Given the dynamics of the system described in this thesis, the use of 2D data solely for constraint equations, and a maximum difference between matched time-tags of the two sensors being ~30 milliseconds, this timing misalignment will have a minimal effect on the outcome of the navigational solution. Although this procedure is expected to work for this system due to its low dynamics, it would not be an accurate choice in most other situations, and it is highly recommended that time tags are synchronized when using this process in the future.

$$tt(x)_{2D \text{ imager}} = \min |tt(x)_{3D \text{ imager}} - tt_{2D \text{ imager}}| \quad (3.10)$$

Figure 20. Time Tag Matching for 3D Imager and 2D Imager.

Given the process by which the IMU data will be fused with the data of the imagers, combined with the knowledge that IMU data can be interpolated, a much more accurate process can be performed. First, using equation (3.10) and replacing the 2D imager's time-tags with that of the IMU's time-tags, the closest time-tag from the IMU data to a specific time-tag from the 3D imager can be found. Next, contrary to the approach used between the 2D imager's data and the 3D imager's data, the next closest time tag in the IMU's data is located, which inherently identifies the two time tags for the IMU that would be directly before and after the time-tag of interest from the 3D imager, if the time tags were listed in numerically increasing order. Finally, the IMU measurements are interpolated to provide an accurate estimate of the data at the exact time the 3D image was captured; this concept is illustrated in Figure 21. In the figure, the orange box shows the interpolation process in which the interpolated IMU measurements will be the combination of 89% of the IMU measurements of the time sample that occurs after the shown 3D imager measurement, and 11% of the IMU data captured at the time sample that occurs before that same measurement. Also, the measurements contained between the points of interpolation in the IMU data are used in the mechanization. Therefore, every bit of data collected by the IMU is used, which inherently means the measurements suffer from nearly no loss in dynamics. The interpolated points will be used at the exact times that the Kalman filter updates, and will create a very accurate comparison of motion between the 3D imager data and the IMU data.

Figure 21. Time Tag Matching for 3D Imager and IMU.

CHAPTER 4: PRE-PROCESSING AND ASSOCIATION METHODOLOGY

This chapter first discusses a derived wavelet de-noising technique used to filter out some of the noise on the used images. Next, feature extraction algorithms are discussed for planar surfaces, points, corners, and line segments. Finally, derived feature association algorithms are stated and explained. Figure 22 displays where the feature extraction and association steps take place in the total process described in this thesis.

Figure 22. Block diagram of full system.

4.1 Wavelet De-Noising

Given that the corner and point extraction methods used in this thesis search for intensity variations between neighboring pixels to find features of interest, a grainy image can very easily cause these algorithms to identify false features. To reduce these false detections, a wavelet de-noising algorithm was used to smooth both the 2D intensity images and the 2D digital images.

In general a wavelet transform is a multi-scale transformation which can be used effectively in place of a Fourier transform in certain applications. A wavelet transform produces a time-frequency representation of a signal using “basis functions” which are scaled and shifted copies of a “mother wavelet.” The details of wavelet transform theory are presented in [40]. Wavelets have proven useful in a number of practical applications, including de-noising, data compression, and signal modulation. The Discrete Wavelet Transform (DWT) can be repeated to an arbitrary number of levels which decomposes the signal to finer frequency resolutions. In addition there are a number of possible “mother wavelets” which can be chosen to achieve desired transform characteristics.

Wavelet de-noising is a technique described in [43]. First, a Stationary Wavelet Transform (SWT) provides a time-frequency (for a 2D image, frequency is interpreted as spatial frequency) representation of a signal. The SWT produces 4 sub images, called approximation (a), horizontal (h), vertical (v), and diagonal (d). The (a) image contains the low pass information from the original image, while the (h), (v), and (d) images contain the high frequency information at the corresponding orientation. The (h), (v), and (d) are then soft threshold to eliminate noise, and then the image is reconstructed using an Inverse SWT (ISWT).

The Matlab wavelet toolbox provides utilities for image de-noising. Many options are available including decomposition level, soft versus hard thresholding, and wavelet type. For this application an algorithm was developed for de-noising images using a discrete stationary 2D wavelet transform. A 2D low pass filter (LPF) was also tested, but yielded very bad results with respect to the corner detection. A basic Haar wavelet was

used at a level 3 decomposition for de-noising the 2D intensity image. The results of this operation are shown in Figure 23. This figure shows the original image on the left and the de-noised image on the right. The Haar wavelet was also used for de-noising the 2D digital image. Since the 2D digital image has a much higher resolution than the 2D intensity image, less smoothing was required, and only a level 2 decomposition was used. The results of the 2D imagery de-noising operations can be observed in Figure 24, where the original image is on the left and the de-noised image is on the right. Unlike Figure 23, the difference between the two images in Figure 24 appears to be minimal, but Figure 25 shows how even this small amount of de-noising makes a significant difference in the feature extraction output. The image on the left of Figure 25 shows the results of the corner extraction algorithm when performed on the original image, and the image on the right shows the results of the same corner extraction algorithm performed on the wavelet de-noised image. Figure 25 also displays on the middle image the results of the corner extraction algorithm on an image after a 2D low pass filter is used. It can be observed from this figure that the corner extraction process identifies nearly random places in the original image and the low pass filtered image as corners. The corner extraction performs well on the wavelet de-noised image. Due to the results in Figure 25, along with numerous other experimental tests, it was concluded that for the images and algorithms used in this thesis, de-noising was a necessary step in order to achieve accurate results.

Figure 23. Original and De-noised Intensity Images.

Figure 24. Original and De-noised Digital Images.

Figure 25. Corner Extraction on Original, LPF, and Wavelet De-Noised Image.

4.2 Feature Extraction

4.2.1 3D Planar Extraction

The 3D planar extraction method used in this thesis was fully developed by Don Venable. For a more in depth overview of this method, the reader is referred to [10]. The planar extraction process described in [10] and shown in Figure 27, begins by taking a frame of data from the 3D imager and dividing it up into smaller sub windows of a pre-determined size. These sub-windows are then examined to find different spatial features: i.e. planar surfaces. To detect planar surfaces an ordinary least squares planar fit method is used on all the 3D point cloud points associated with each of the sub-window pixels. By comparing the sum-of-the-squared-residuals to a predefined threshold, the method stochastically determines which sub-windows have a high probability of containing a planar surface. After all the sub-windows are identified that have a high probability of containing a planar surface, a derived histogramming method is performed. This method takes these identified sub-windows and tests them to accurately identify all planar surfaces, and the points each plane contains [10]. The results of this planar extraction process can be seen in Figure 26, and the thresholds used in this algorithm are listed in Table 2.

Figure 26. Planar Extraction Results.

Figure 27. Planar Extraction Process[10]

note: (taken from [10] with permission from the author)

After the planar features are extracted, a basic set of parameters is stored for each planar surface that uniquely describes it. For each plane, a centroid, $\mathbf{c}$, and a normal vector, $\mathbf{n}$, are determined which are used in equations (4.1) and (4.2) to calculate the given plane's normal point $\mathbf{v}$, and the distance to that normal point ρ . After this is

complete, the centroid, normal point, normal distance, and 3D normal vector are stored to characterize each specific plane.

$$\mathbf{v} = -\text{sign}(\mathbf{c} \cdot \mathbf{n})\mathbf{n} \quad (4.1)$$

$$\rho = |\mathbf{c} \cdot \mathbf{n}| \quad (4.2)$$

4.2.2 3D and 2D Line Extraction

Other features of interest investigated within the scope of this thesis are line segments extracted from both the 2D digital image and the 3D point cloud. The process for this extraction, shown in Figure 28 for the 2D intensity image, contains multiple steps to determine where the strongest line segments in an image are located. The first step of this process is the use of an edge detector on both the 2D digital image and the 2D intensity image, which corresponds to the 3D point cloud data by taking out the depth information. Using the built-in edge detector capability in Matlab's image processing toolbox, some edge detector examples were implemented and shown in Figure 29, possible line segments are identified since all line segments would be identified as edges, although not all edges will be used as line segments. For this thesis, the Canny edge detector was selected as the edge detector of choice. The Canny edge detection results are shown in part (b) of Figure 28.

The next step in the algorithm is the elimination of all edges that are shorter than a specified length threshold. The necessary length of these lines is an adjustable threshold

that was empirically determined by inspecting a large set of 2D and 3D images individually. The results of this filtering process can be seen in part (c) of Figure 28.

After the line segments deemed to be too short to be reliable are eliminated, a split and merge method, explained in detail in the next section, is performed on each of the remaining edges. This process breaks down each edge into smaller line segments in which the furthest outlying point of a segment is less than a pre-determined distance from a line connecting the edge's endpoints. Since this algorithm breaks up edges into smaller line segments, the length criteria mentioned above is applied again to eliminate all line segments that are not long enough. The results of this step for the 2D intensity image can be seen in part (d) of Figure 28 and is the final step in the line extraction process. The final results of this line extraction process for the digital image are shown in Figure 30, but the information in the 2D intensity image must undergo more manipulation. For the 2D intensity image, the coordinates of the lines extracted must be mapped to the 3D world coordinates. Each pixel in the 2D intensity image corresponds directly to a point in the 3D point cloud and therefore, the 3D world coordinates for each extracted line segment can be obtained easily. Once the 3D coordinates are known for each pixel contained in the line segments, a 3D line fit is performed on these points. The variance of the residuals of this line fit is compared to a threshold, and lines are eliminated if this threshold is exceeded. This process assures that no lines are kept that contain a significant amount of noise. Once this process is complete, a 2D line fit is performed on the lines from the 2D digital image, and the results of 3D line fit, are used to identify the end

points of each lines line fit. These end points are then stored for further use. The pre-determined thresholds for this process can be found in Table 2.

Figure 28. Line Extraction Process Performed on a 2D Intensity Image

Figure 29. Different Edge Detectors Performed on a 2D Intensity Image

Figure 30. Line Extraction Results from a 2D Digital Image

4.2.2.1 Split and Merge

The split and merge methodology contains two major steps. The first step is performing the split method, shown in Figure 31. This procedure is an iterative process of dividing any set of connected lines and breaking them down into individual lines that satisfy specific pre-determined criteria. Step (a) shows an extracted edge from the image in black with the starting point of this edge labeled '1' and the end point labeled '2'. Examining this edge, a line is drawn from point '1' to point '2', shown in blue, and from this line a perpendicular line, in red, is drawn to the point on the edge that is the furthest distance from the blue line. If the red line is less than a predetermined maximum length, then the points contained between points '1' and '2' are considered to be one line. If the red line is greater than the maximum length allowed, the intersection of the red line and the black edge becomes point '2' and the process is iterated

again. Part (b) of Figure 31 shows where the new point '2' is located and that every point in the edge past that point is temporarily ignored. It can also be observed in part (b) that in this case with perfectly straight lines, the length of a perpendicular line drawn in the same manner as the red line in part (a) is equal to 0. Due to the length of this perpendicular line being less than any possible threshold, all points between points '1' and '2' in part (b) are considered to be one line. After a line is determined, it is taken out of the original black edge, and the remaining edge go through the same process. The remaining parts of Figure 31 show how these iterations occur. In part (f), it can be seen that this algorithm extracts three different line segments, divided by colors, from the original black edge shown in part (a).

Figure 31. Split Procedure of the Split and Merge Method.

After the split method has been completed, the merge method is performed. This procedure, for both 2D and 3D lines, takes all of the line segments extracted by the split method and compares them to identify and merge line segments that may have been broken up due to noise. This method compares three different aspects of lines to determine whether or not to merge them together. First the process compares the unit vectors of different lines. It determines whether two lines consist of unit vectors that differ by less than a small pre-determined percentage. If two lines have similar enough unit vectors, their endpoints are compared. The endpoints for each of the two lines are examined to determine if they would contain the others end points if extended forever given a pre-determined variance on the unit vector of each line. Finally, if the previous two criteria are met, then the shortest distance between the two line segments is derived. If this distance is shorter than a pre-determined length, the two lines are merged into one line and points are interpolated to fill the gap between the lines. All of the pre-determined thresholds for this process are listed in Table 2, and Figure 32 shows examples of this process.

Figure 32. Merge Procedure of the Split and Merge Method.

4.2.3 3D and 2D Corner Extraction

With corner extraction being a desired output of this thesis, different extraction methods were examined. An algorithm designed by Edward Rosten and Tom Drummond called Features from Accelerated Segment Test (FAST), was chosen. This process systematically finds points of interest by evaluating the Univalue Segment Assimilating Nucleus (USAN) measure. The USAN measure is a way of quantifying places in an image that highly resemble a wedge shape.[41] These values are useful under the assumption that a corner in an image, when examined at the pixel level, resembles a blurred wedge shape. The smallest values of the USAN measure are considered points of interest and examined using the point segment test described in [41]. The point segment test, displayed in Figure 33, first takes a point of interest and opens a circular search

space centered on it, consisting of 16 square pixels. The 16 pixels in the search space are displayed in Figure 33 as blue boxes. The algorithm then compares the intensity of the pixel of interest, identified in Figure 33 by the green box, to each one of the 16 pixels in the search space. A search is then conducted between the 16 examined pixels, to determine if a set of connecting pixels exists, in which all of the pixels are brighter than the point of interest plus a threshold, or darker than the point of interest minus a threshold[41]. The section of a red circle in Figure 33 illustrates this process by tracing through a set of pixels from the 2D digital image and identifying a set of 10 connecting pixels that all have an intensity value less than the intensity of the point of interest minus the threshold value specified in Table 2. For this thesis, the minimum number of connecting pixels that satisfy the previously mentioned criteria was set to be 9. Therefore, the example displayed in Figure 33 shows a scenario in which a corner would be extracted from the set of pixels examined because it has more than 9 connecting pixels that satisfy the determined criteria. Both the 2D digital and intensity images are analyzed to identify corner locations. The 2D intensity images corners are then mapped to the 3D point cloud and stored with the digital images corners. A sample of the results of this algorithm performed on the 2D digital image can be seen in Figure 34. For more information on the FAST algorithm and the USAN measure, please refer to [41] and [42].

Figure 33. 9 Point Segment Test for FAST Corner Extraction. (adapted from [41])

Figure 34. FAST Corner Extraction Performed on 2D Digital Image.

4.2.4 3D and 2D Point Extraction and Association

The final type of feature utilized in this thesis is the point feature type. The method chosen to obtain these is using the SIFT algorithm designed by Lowe, and is discussed in more detail in chapter 2 of this thesis and in [25]. One of the largest

problems with SIFT is its computational complexity in the matching and association portions of the algorithm, but since the work completed in this thesis was done from a post processing stand point, the computation required to process this algorithm is not a concern.

The SIFT algorithm first extracts a significant number of point features on a given image, a sample is shown in Figure 35. The algorithm then takes the next consecutive image from the data, performs the point extraction process, and then identifies all of the common points between the two images. This is done by comparing the distances between every point in the first image individually with every point in the second image. If the distance between a point in the first image and a single point in the second image multiplied by the threshold specified in Table 2, is less than the distance between the specified point in the first image and any other point in the second image, then the two points are matched. Using this matched data, up to 6 matches are chosen to use in the navigation solution, based on how well they match. A sample of the results of this process can be seen in Figure 36. Holding to only 6 matches is done to artificially create data that would be more realistic with respect to a real time solution. With the eventual goal of all navigation techniques being run in real time, dealing with fewer point features per frame is more realistic than hundreds of points. This elimination process also eliminates possible mismatches, as well as limits the observable motion that can be obtained from these features. Therefore, the true capabilities of SIFT are significantly limited by the expulsion of a majority of its matched points to force the data to be more realistic for time analysis. Once points are extracted on both the 2D digital and intensity

images using the thresholds described in Table 2, the 2D digital corners are stored, and the corners from the 2D intensity image are mapped into their corresponding 3D point cloud and stored.

Figure 35. SIFT Point Extraction Performed on 2D Digital Image.

Figure 36. Chosen Associated Point Features from SIFT algorithm.

4.2.5 Determined 3D and 2D Extraction Thresholds

The thresholds for each of the extraction processes shown in Table 2 were chosen based on calculations and experimental results. These thresholds were chosen at their listed value specifically for the data set used in this work.

Table 2. Calculated Thresholds for Feature Extraction.

Feature Type	Purpose	Threshold Value (units)
2D lines	Minimum number of pixels per line	120 pixels
	Maximum distance for outliers in split and merge method	8 pixels
	Percent variation in unit vector for split and merge	7%
	Maximum distance between lines for merge criteria	50 pixels
3D lines	Minimum number of pixels per line	70 pixels
	Maximum distance for outliers in split and merge method	3 pixels
	Percent variation in unit vector for split and merge	10%
	Maximum distance between lines for merge criteria	20 pixels
3D planes	Minimum number of pixels per plane	200 pixels
	Size of each subwindow	16 x 16 pixels
2D corners	Intensity difference between corner pixel and surrounding pixels	2.5
3D corners	Intensity difference between corner pixel and surrounding pixels	15
2D point features	Minimum of difference of Gaussian function	0.0067
3D point features	Minimum of difference of Gaussian function	0.0067

4.3 Feature Association

Feature association is the process that provides the information necessary to calculate observed feature motion between frames. In order to complete feature association, two steps must occur. The first step is to take previously observed features and predict the position of those features in the current frame, and the second step is to mathematically match previously seen features with their new positions in the current frame.

In order to predict where previously identified features will be located in a current frame, the IMU measurements will be used. The IMU measurements will provide a measured motion estimate of the sensor platform through estimates of its translational motion, $\Delta\hat{\mathbf{x}}$, and rotational motion $\hat{\mathbf{C}}_{t_k}^{t_{k+1}}$. Using these estimates, it is possible to predict the locations of previously seen features based on how the sensor platform observing them has moved. This process can be done for any point in 3D space, $\mathbf{v}_i(t_k)$, using equation (4.3) where $\mathbf{v}_i(t_{k+1})$ represents the point's predicted location. An example of this prediction process can be observed in Figure 37. In this figure, the top image shows a previous image frame with extracted corner features highlighted in green. The middle image shows how the features in the original scene change position after the imaging sensor encounters translational motion, and the bottom image shows the same after the sensor also encounters a rotational motion.

$$\mathbf{v}_i(t_{k+1}) = \hat{\mathbf{C}}_{t_k}^{t_{k+1}} [\mathbf{v}_i(t_k) + \Delta\hat{\mathbf{x}}] \quad (4.3)$$

Figure 37. Image Translation and Rotation.

This projection process can be performed between any two frames given the time of capture for both frames and the inertial measurements between them. Consequently, the use of IMU measurements for this process inherently means that the process suffers from the same drift errors as the INS. Therefore, just as an INS, the longer inertial measurements are used for feature position predictions with no correction, the more erroneous the predictions become. With this taken into consideration, it can be stated that the further forward in time features are projected, the worse the accuracy is on their

estimated positions. Due to this, features are only projected forward for a maximum of 5 image frames in this research. Since the images are being evaluated at a rate of $\sim 5\text{Hz}$., it can be deduced that no more than 1 second of IMU data will be used to project feature positions forward. This will greatly limit the amount of drift that will be contained in the IMU measurements used for feature position predictions, and therefore make our predictions more accurate.

Once a feature position prediction can be completed, a mathematical process must be laid out to determine if projected features from previous frames correspond to the same features in the current frame. This process must be carefully tailored to each different type of feature and both the 2D and 3D data. It is very important that common features between frames are identified correctly. Any error in identifying common features between frames will add false motion to the observed motion estimates and therefore inject an error that will never be removed, inherently adding a permanent bias to the navigational solution. The effects of such an error are described in more detail in section 6.2.2 of [28] and evaluated for the SLAM algorithm in [24]. An example of this association process can be viewed in Figure 38, where the blue lines represent correct associations and the red line represents a false match. The two features connected by the red line in Figure 38 show the importance of the feature position prediction being incorporated with mathematical testing for common features. When examined at a close range, these two features appear to be very similar, but by using the IMU measurements to predict the positions of features, it can be seen that the two features associated incorrectly in Figure 38 are not actually the same feature.

Figure 38. Association Between Two Frames.

4.3.1 3D Feature Association

Given the association steps listed above, combined with the four types of 3D features used in this thesis, a detailed process can be laid out to perform association on all of the 3D data. With projection being the first step in the process, a procedure must be determined to project forward all of the necessary information to accurately predict the positions of each feature in future frames. For point features, this process is not necessary since the SIFT procedure provides the association steps. Corner features are stored as a single point in 3D space, so equation (4.3) will provide the necessary mathematics to predict the location of these. The chosen descriptive characteristics of a line feature are its end points, as well as its unit vector, $\hat{l}_i(t_k)$. Just as the corner features were projected forward in time using equation (4.3), the end points of the line can be projected in the same way. For the unit vector each line a different method must be used, shown in equation (4.4). Equation (4.4) shows the necessary computation to estimate how a unit vector will change in orientation with respect to the sensor platform. In a similar process,

the location of planar features can be predicted using equations (4.3) and (4.5). Planar surfaces are represented in this work by a statistically derived centroid and a normal vector. The centroid of each plane can be projected forward using equation (4.3) and the orientation of the normal vector can be projected forward using equation (4.5).

$$\hat{\mathbf{l}}_i(t_{k+1}) = \hat{\mathbf{C}}_{t_k}^{t_{k+1}} * \hat{\mathbf{l}}_i(t_k) \quad (4.4)$$

$$\hat{\mathbf{n}}_i(t_{k+1}) = \hat{\mathbf{C}}_{t_k}^{t_{k+1}} * \hat{\mathbf{n}}_i(t_k) \quad (4.5)$$

Once all of the necessary 3D feature descriptors have been projected into estimated positions in the current frame of interest, features in the current frame must be compared to the projected features to find common features. 3D corner features are nothing more than a single point in the 3D point cloud, and therefore are associated between frames based solely on the difference between their location in the current frame and their predicted positions. If this difference does not exceed a pre-determined threshold, specified in Table 3, the two corner features being compared are associated. The information used to describe the line features in the current frame and the previously seen projected lines, their end points and unit vector, can be used in equations (4.1) and (4.2) to determine each lines normal point location, and the distance to that normal point. When comparing the previously seen line features to the currently seen line features, the difference between any of these properties cannot exceed the experimentally derived thresholds specified in Table 3. Therefore line features are associated based on comparing their unit vectors, normal points, and normal distances. Planar features are

associated in a similar process as line features by comparing their centroids, normal vectors, normal distances, and normal points. Once association is done between frames for the 3D data, the information produced is fed into the Kalman Filter for navigational purposes. The results of the 3D association process can be visualized in Figure 39, where the images display previously observed features associated with features extracted from the frame shown. The image on the left shows these features in the 3D point cloud, and the image on the right shows the same features displayed on the corresponding intensity image.

Figure 39. Results of 3D Feature Association

4.3.2 2D Feature Association

If the process used for 3D feature association, discussed in section 4.3.1, is used for 2D feature association, a significant problem arises. All of the 2D features are represented by 2D coordinates, while the IMU data used to project previously seen

features forward is strictly in 3 dimensions. Due to this problem, a mathematical relationship between the 2D image frame coordinates and the 3D world coordinates to which they correspond to must be established.

To establish the relationship between 2D image frame coordinates and 3D world coordinates, a priori knowledge of the vision sensor and the principles of multi-view geometry, shown in Figure 40, are the necessary components. The knowledge of the vision sensor is established through the camera calibration process, discussed in section 3.5 of this thesis, and is used to establish the calibration matrix, previously stated in equation (3.7) and stated again as equation (4.6). The camera calibration matrix can then be used in equation (4.7), to mathematically establish a way of transforming any set of 2D image coordinates into a 3D world frame. Once all of the information necessary to use equation (4.7) is obtained, 2D features can be transformed into 3D features. Then the IMU data containing measured translational motion, $\Delta \mathbf{x}$, and rotational motion, $\mathbf{C}$, can be used to predict the observed motion of each feature. The transformed 3D features can then be used in equation (4.7) again to obtain an estimate of the predicted 2D coordinates of any 2D feature after motion captured by the IMU takes place. Figure 40 visually shows how this process is implemented. The rectangular box on the left side of the figure represents the first image frame on which the location of a line feature can be observed. From this, using equation (4.7) the line can be projected into 3D coordinates, and the translation and rotational motion of the sensor can be applied to it. Using this, the 3D line feature can be projected back onto the image frame after motion has occurred, shown in the rectangular box on the right side of the image to predict where the 2D feature will be

located. It is important to note here that the projection established in equation (4.7) and Figure 40 does not resolve depth. The depth is artificially added to the projection using the scalar λ . Therefore, this projection is an estimate and contains some error, but since depth is not known for any of the 2D features, this is an unavoidable consequence. Although the depth of each of these coordinates is artificially added, the results from this projection remain accurate enough to use for this research.

$$\mathbf{K} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \text{ (from equation (3.7))} \quad (4.6)$$

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{C} & \Delta \mathbf{x} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \text{ (from equation (3.8))} \quad (4.7)$$

Figure 40. 2D Multi-View Line and Point Geometry [38]

Using the position prediction process discussed above for 2D features, association can be completed in a similar fashion to that of the 3D features. 2D association must be

completed on 2D corner features, and 2D line features. For this work, since SIFT was used for the 2D point features, the association step of these features is done in the SIFT algorithm. Therefore, no association process is needed for 2D point features. For 2D corner features, the association process is completed by examining the distance between the predicted position of previously observed corners and the positions of currently observed corners. Just as for 3D corner features, if this distance is less than a pre-determined threshold, specified in Table 3, then the two corners are associated. When examining the association process for 2D line segments, after the 2D position prediction process is completed, the same process used for 3D line association can be used. If the endpoints of a projected line are contained in a line in the current frame given a specified difference, the difference in slope between the two lines is less than a specified percentage, and the normal point and normal distance between the two line segments are less than the pre-determined thresholds established in Table 3, then the two lines are associated. A sample of the results for this 2D association process can be seen in Figure 41.

Figure 41. Results of 2D Feature Association

4.3.3 Determined 3D and 2D Association Thresholds

Given the importance of the association phase of the process discussed in this thesis, it was crucial to go through a rigorous process to determine the thresholds necessary to avoid associating features incorrectly. As discussed previously, a miss-association will inject false motion information into the Kalman filter, which will never be removed from the final solution. To limit the probability of a miss-association, a two step approach was taken to determine the thresholds displayed in Table 3. First by examining the output of the extraction process, estimates of the thresholds were mathematically obtained. The second step was to experimentally collect data and tune these estimates to yield the best results possible. Using the idea displayed in Figure 42, data was experimentally collected using random audits on 50 frames of image data to determine the values of the variables used in the association process for correct and incorrect associations. Once this data was collected, it was observed that a small portion

of the values that would indicate an association was interlaced with a small portion of the values that indicate a miss-association. This is displayed in Figure 42 by the intersection of the two Gaussian curves. Due to this intersection, thresholds were chosen to be values that completely eliminated all false association observed in the random audits. This does not mean that the probability of a miss-association is 0, but since the audits were random, the probability should be very small. As a direct result of this, a small portion, $\sim 3\%$, of the true associations will also be eliminated. Due to the adverse affect of miss-associations, it was felt that setting the thresholds to eliminate all of the observed false associations outweighed the cost of losing a small percentage of the actual associations. The empirical approach discussed above works well for association between frames of the collected data sets used in this thesis, but these experimentally determined thresholds will likely not perform well on different data sets. To improve the performance of the association algorithms, more complex techniques can be explored such as those discussed in [28].

Figure 42. Probability Density Functions of Association and Miss-association

Table 3. Calculated Thresholds for Feature Association.

Feature Type	Purpose	Threshold Value (units)
2D lines	Maximum distance between projected normal point and observed normal point	0.1 pixels
	Maximum difference between projected normal distance and observed normal distance	8 pixels
	Percent variation in unit vector	8%
3D lines	Maximum distance between projected normal point and observed normal point	0.05meters
	Maximum difference between projected normal distance and observed normal distance	0.1meters
	Percent variation in unit vector	10%
3D planes	Maximum distance between projected normal point and observed normal point	0.2 meters
	Maximum difference between projected normal distance and observed normal distance	0.2 meters
	Percent variation in unit vector	15%
	Maximum distance between projected centroid and observed centroid	0.1 meters
2D corners	Maximum distance between projected pixel and observed pixel locations	$\sqrt{2}$ pixels
3D corners	Maximum distance between projected pixel and observed pixel locations	$\sqrt{2}$ pixels
2D point features	Distance multiplier threshold for SIFT	1.5
3D point features	Distance multiplier threshold for SIFT	1.3

CHAPTER 5: MULTIPLE FEATURE TYPE INTEGRATION MECHANIZATION

As previously mentioned, the approach taken in this thesis for combining the data captured from the 2D imager, the 3D imager, and the IMU are integrated using a complementary Extended Kalman Filter. When designing an EKF, the first few steps are to define a state vector, set up a dynamics model, examine all of the available measurements obtained, and find the relationships between the observed parameters of the state vector.

The state vector for this work is shown in equation (5.1), and is based on the state vector described in [10]. The states were chosen with the intent of estimating the error states of the IMU. These error states include position displacement error, $\delta \mathbf{r}^n$, velocity error, $\delta \mathbf{v}^n$, tilt or miss-orientation, $\boldsymbol{\psi}^n$, gyro bias, $\delta \boldsymbol{\omega}^b$, and accelerometer bias, $\delta \mathbf{f}^b$. The corresponding state transition matrix, $\mathbf{F}$, to this state vector is later derived in equation (5.38).

$$\mathbf{x} = \begin{bmatrix} \delta \mathbf{r}^n \\ \delta \mathbf{v}^n \\ \boldsymbol{\psi}^n \\ \delta \boldsymbol{\omega}^b \\ \delta \mathbf{f}^b \end{bmatrix} \quad (5.1)$$

5.1 Planar Surface Navigation Equations

The first set of measurements that has been examined consists of the measurements that define planar surfaces. These measurements, previously analyzed in [10], provided us with knowledge of each plane's normal vector, $\mathbf{n}_i$, and the shortest distance to the planar surface (normal point), ρ_i , where 'i' represents an index for each of N currently observed and associated planar surfaces. For each change in time, t_{k-1} to t_k , between measurements relating to the planar surfaces, the IMU provides an estimated translational motion of the camera, $\Delta \hat{\mathbf{f}}_{INS}(t_k, t_{k-1})$, and an estimated orientation change of the camera, $\Delta \hat{\mathbf{C}}_{t_{k-1}}^{t_k}$. With the available information, we can now form expressions for the change in shortest distance to the planar surface which can be used to estimate the velocity and position errors in the IMU measurements, one expression based on the 3D point cloud measurements and one expression derived from the inertial measurements; equations (5.2) and (5.3). The complementary filter "measurement" is formed by the difference between the results of equation (5.2) and (5.3) as shown in equation (5.4): the measurement vector, $\mathbf{z}_{\text{planes,position}}$, used for planar surfaces in the EKF.

$$\Delta \boldsymbol{\rho}_{\text{planes}}(t_k) = \begin{bmatrix} \rho_1(t_k) - \rho_1(t_{k-1}) \\ \vdots \\ \rho_N(t_k) - \rho_N(t_{k-1}) \end{bmatrix} \quad (5.2)$$

$$\Delta \boldsymbol{\rho}_{\text{INS}}(t_k) = \begin{bmatrix} \mathbf{n}_1(t_{k-1}) \\ \vdots \\ \mathbf{n}_N(t_{k-1}) \end{bmatrix} [\hat{\mathbf{C}}_n^b(t_{k-1}) \Delta \hat{\mathbf{f}}_{INS}(t_k, t_{k-1})] \quad (5.3)$$

$$\mathbf{z}_{\text{planes,position}}(t_k) = \Delta \boldsymbol{\rho}_{\text{INS}}(t_k) - \text{sign}(\Delta \boldsymbol{\rho}_{\text{planes}}(t_k)) * |\Delta \boldsymbol{\rho}_{\text{planes}}(t_k)| \quad (5.4)$$

After the state vector and the measurement vector for planar surfaces are determined, the observation matrix, $\mathbf{H}_{\text{planes,position}}$ can be obtained. As previously mentioned in chapter 2, the $\mathbf{H}$ matrix describes the relationship between the state vector and the measurement vector with noise taken into consideration. This relationship is restated in equation (5.5). This observation matrix for planar surfaces, a row of which is shown in equation (5.6), will be an $N \times 15$ matrix, where each of the 15 columns correspond to an error state of the state equation, and each row represents a single plane, i , of N total planes.

$$\mathbf{z}(t_k) = \mathbf{H}(t_k)\mathbf{x}(t_k) + \mathbf{v}(t_k) \quad (\text{originally equation (2.19)}) \quad (5.5)$$

$$\mathbf{H}_{\text{planes,position}}(t_k) = [\Delta t * \hat{\mathbf{n}}_i^T * \hat{\mathbf{C}}_n^b(t_{k-1}) \quad \mathbf{0}_{3 \times 1} \quad \mathbf{0}_{3 \times 1} \quad \mathbf{0}_{3 \times 1} \quad \mathbf{0}_{3 \times 1} \quad] \quad (5.6)$$

Equations (5.4) and (5.6) yield the necessary math to correct for position and velocity errors, but do not provide the capability of estimating attitude errors. By forming a different and complementary set of measurement and observation equations, attitude error can be estimated. Beginning with the statement shown in equation (5.7), equation (5.8) can be derived to measure the error in attitude of the normal vector of each plane.

$$\mathbf{n}(t_k) = \mathbf{C}_n^b(t_k) * \mathbf{C}_b^n(t_{k-1})\mathbf{n}(t_{k-1}) \quad (5.7)$$

$$\mathbf{z}_{\text{planes,attitude}}(t_k) = \begin{bmatrix} \hat{\mathbf{n}}_1(t_k) \\ \vdots \\ \hat{\mathbf{n}}_N(t_k) \end{bmatrix} - \hat{\mathbf{C}}_n^b(t_k) * \hat{\mathbf{C}}_b^n(t_{k-1}) * \begin{bmatrix} \hat{\mathbf{n}}_1(t_{k-1}) \\ \vdots \\ \hat{\mathbf{n}}_N(t_{k-1}) \end{bmatrix} \quad (5.8)$$

Given that $\hat{\mathbf{C}}_n^b(t_k) = \mathbf{C}_n^b(t_k)[\dot{\boldsymbol{\psi}} \times]$ equation (5.8) can be rewritten in terms of the error sources as:

$$\begin{aligned}
 &= \Delta t \mathbf{C}_n^b(t_k)[\dot{\boldsymbol{\psi}} \times] * \hat{\mathbf{C}}_b^n(t_{k-1}) * \begin{bmatrix} \hat{\mathbf{n}}_1(t_{k-1}) \\ \vdots \\ \hat{\mathbf{n}}_N(t_{k-1}) \end{bmatrix} \Rightarrow \\
 &= -\Delta t \mathbf{C}_n^b(t_k) * \hat{\mathbf{C}}_b^n(t_{k-1}) * \begin{bmatrix} \hat{\mathbf{n}}_1(t_{k-1}) \\ \vdots \\ \hat{\mathbf{n}}_N(t_{k-1}) \end{bmatrix} \times \dot{\boldsymbol{\psi}} \Rightarrow \\
 &= [-\Delta t \mathbf{C}_n^b(t_k) * \hat{\mathbf{C}}_b^n(t_{k-1}) * \begin{bmatrix} \hat{\mathbf{n}}_1(t_{k-1}) \\ \vdots \\ \hat{\mathbf{n}}_N(t_{k-1}) \end{bmatrix} \times * \mathbf{C}_b^n] \delta \boldsymbol{\omega} \tag{5.9}
 \end{aligned}$$

Using equation (5.9), equation (5.10) can be written to describe $\mathbf{H}_{\text{planes,attitude}}$, the observation matrix corresponding the measurement vector, $\mathbf{z}_{\text{planes,attitude}}$. This second set of observation and measurement equations will create matrices and vectors of a different size than the first set of equations for planar surfaces. For each planar surface observed, the $\mathbf{H}_{\text{planes,attitude}}$ matrix will gain three additional rows making it a $3N \times 15$ matrix, where N is the total number of planes. Correspondingly, the measurement vector, $\mathbf{z}_{\text{planes,attitude}}$, will also gain three additional rows for each observed plane making it a $3N \times 1$ vector.

$$\mathbf{H}_{\text{planes,attitude}}(t_k) =$$

$$\begin{bmatrix} \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & -\Delta t \mathbf{C}_n^b(t_k) * \hat{\mathbf{C}}_b^n(t_{k-1}) * \begin{bmatrix} \hat{\mathbf{n}}_1(t_{k-1}) \\ \vdots \\ \hat{\mathbf{n}}_N(t_{k-1}) \end{bmatrix} \times * \mathbf{C}_b^n & \mathbf{0}_{3 \times 3} \end{bmatrix} \tag{5.10}$$

Combing the two different sets of observation and measurement equations used for planar surfaces, equations (5.11) and (5.12) can be stated. For N planes, the observation matrix, equation (5.11), will be a $4N \times 15$ matrix, and the measurement vector, will be a $4N \times 1$ vector. By using equations (5.11) and (5.12) a full solution can be obtained to estimate the errors described in the previously mentioned state matrix.

$$\mathbf{z}_{\text{planes}} = \begin{bmatrix} \mathbf{z}_{\text{planes,position}} \\ \mathbf{z}_{\text{planes,attitude}} \end{bmatrix} \quad (5.11)$$

$$\mathbf{H}_{\text{planes}} = \begin{bmatrix} \mathbf{H}_{\text{planes,position}} \\ \mathbf{H}_{\text{planes,attitude}} \end{bmatrix} \quad (5.12)$$

5.2 3D Point and Corner Navigation Equations

Mathematically, the measurements of the 3D corners and points are identical. Therefore, the equations used in the EKF for these two types of features will be the same. It is assumed that the known derived 3D point measurement expressed in the body frame, $\hat{\mathbf{p}}_i^b(t_k)$, where ‘i’ is an index indicating all available 3D points, will only have a noise component as shown in equation (5.13).

$$\hat{\mathbf{p}}_{\text{sensor},i}^b(t_k) = \mathbf{p}_i^b(t_k) + \varepsilon_i(t_k) \quad (5.13)$$

Expressions for the 3D points can also be derived using the inertial mechanization outputs, $\Delta\hat{\mathbf{r}}_{INS}$ and $\hat{\mathbf{C}}_b^n$, and the relationship given in equation (5.14) relates points at time epoch t_{k-1} to t_k using the inertial measurements.

$$\begin{aligned}\mathbf{C}_b^n(t_k)\mathbf{p}_i^b(t_k) &= \mathbf{C}_b^n(t_{k-1})\mathbf{p}_i^b(t_{k-1}) - \Delta\mathbf{r}(t_k, t_{k-1}) \Rightarrow \\ \mathbf{p}_i^b(t_k) &= \mathbf{C}_n^b(t_k)\hat{\mathbf{C}}_b^n(t_{k-1})\mathbf{p}_i^b(t_{k-1}) - \mathbf{C}_n^b(t_k)\Delta\hat{\mathbf{r}}(t_k, t_{k-1})\end{aligned}\quad (5.14a, b)$$

Rewriting equation (5.14) and substituting orientation and translation estimates from the inertial mechanization yields the following estimate of the point location:

$$\hat{\mathbf{p}}_{INS,i}^b(t_k) = \hat{\mathbf{C}}_n^b(t_k)\hat{\mathbf{C}}_b^n(t_{k-1})\mathbf{p}_{INS,i}^b(t_{k-1}) - \hat{\mathbf{C}}_n^b(t_k)\Delta\hat{\mathbf{r}}_{INS}(t_k, t_{k-1}) \quad (5.15)$$

Given that $\hat{\mathbf{C}}_b^n(t_k) = [\mathbf{I} - \boldsymbol{\Psi}(t_k)]\mathbf{C}_b^n(t_k)$ or $\hat{\mathbf{C}}_n^b(t_k) = \mathbf{C}_n^b(t_k)[\mathbf{I} + \boldsymbol{\Psi}(t_k)]$, equation (5.15) can be rewritten in terms of the error sources as:

$$\begin{aligned}\hat{\mathbf{p}}_{INS,i}^b(t_k) &= \mathbf{C}_n^b(t_k)[\mathbf{I} + \boldsymbol{\Psi}(t_k)][\mathbf{I} - \boldsymbol{\Psi}(t_{k-1})]\mathbf{C}_b^n(t_{k-1})\mathbf{p}_{INS,i}^b(t_{k-1}) \\ &\quad - \mathbf{C}_n^b(t_k)[\mathbf{I} + \boldsymbol{\Psi}(t_k)](\Delta\mathbf{r} + \delta\mathbf{r}) \\ &= \mathbf{C}_n^b(t_k)\mathbf{C}_b^n(t_{k-1})\mathbf{p}_{INS,i}^b(t_{k-1}) \\ &\quad + \mathbf{C}_n^b(t_k)\left[\mathbf{I} + \boldsymbol{\Psi}(t_k) - \boldsymbol{\Psi}(t_{k-1}) - \underbrace{\boldsymbol{\Psi}(t_k)\boldsymbol{\Psi}(t_{k-1})}_{\approx \mathbf{0}}\right]\mathbf{C}_b^n(t_{k-1})\mathbf{p}_{INS,i}^b(t_{k-1}) \\ &\quad - \mathbf{C}_n^b(t_k)[\mathbf{I} + \boldsymbol{\Psi}(t_k)](\Delta\mathbf{r} + \delta\mathbf{r}) \\ &\approx \mathbf{p}_i^b(t_k) - \mathbf{C}_n^b(t_k)[\boldsymbol{\Psi}(t_k) - \boldsymbol{\Psi}(t_{k-1})]\mathbf{C}_b^n(t_{k-1})\mathbf{p}_{INS,i}^b(t_{k-1}) - \mathbf{C}_n^b(t_k)[\mathbf{I} + \boldsymbol{\Psi}(t_k)]\delta\mathbf{r}^n\end{aligned}$$

$$\begin{aligned}
&\approx \mathbf{p}_i^b(t_k) - \mathbf{C}_n^b(t_k)[\Psi(t_k) - \Psi(t_{k-1})]\mathbf{C}_b^n(t_{k-1})\mathbf{p}_{INS,i}^b(t_{k-1}) - \mathbf{C}_n^b(t_k)\delta\mathbf{r}^n \\
&\approx \mathbf{p}_i^b(t_k) - \mathbf{C}_n^b(t_k)\left[\dot{\Psi} \cdot [t_k - t_{k-1}]\right]\mathbf{C}_b^n(t_{k-1})\mathbf{p}_{INS,i}^b(t_{k-1}) - \mathbf{C}_n^b(t_k)\delta\mathbf{r}^n \\
&= \mathbf{p}_i^b(t_k) - \Delta t \mathbf{C}_n^b(t_k)[\dot{\Psi}^n \times]\mathbf{C}_b^n(t_{k-1})\mathbf{p}_{INS,i}^b(t_{k-1}) - \mathbf{C}_n^b(t_k)\delta\mathbf{r}^n \\
&= \mathbf{p}_i^b(t_k) - \Delta t \mathbf{C}_n^b(t_k)[\dot{\Psi}^n \times]\mathbf{p}_{INS,i}^n(t_{k-1}) - \mathbf{C}_n^b(t_k)\delta\mathbf{r}^n \\
&= \mathbf{p}_i^b(t_k) + \Delta t \mathbf{C}_n^b(t_k)\mathbf{p}_{INS,i}^n(t_{k-1}) \times \dot{\Psi}^n - \mathbf{C}_n^b(t_k)\delta\mathbf{r}^n \\
&= \mathbf{p}_i^b(t_k) + [\Delta t \mathbf{C}_n^b(t_k)\mathbf{p}_{INS,i}^n(t_{k-1}) \times]\mathbf{C}_b^n(t_k)\delta\omega^b - \mathbf{C}_n^b(t_k)\delta\mathbf{r}^n \\
&= \mathbf{p}_i^b(t_k) + [\Delta t \mathbf{C}_n^b(t_k)\mathbf{p}_{INS,i}^n(t_{k-1}) \times \mathbf{C}_b^n(t_k)]\delta\omega^b - \mathbf{C}_n^b(t_k)\delta\mathbf{r}^n
\end{aligned} \tag{5.16}$$

Subtracting equations (5.13) and (5.16), yields the measurement equation for a single feature:

$$\begin{aligned}
\mathbf{z}_{3Dpoints,i} &= \\
&\mathbf{p}_i^b(t_k) + \varepsilon_i(t_k) - \mathbf{p}_i^b(t_k) - [\Delta t \mathbf{C}_n^b(t_k)\mathbf{p}_{INS,i}^n(t_{k-1}) \times \mathbf{C}_b^n(t_k)]\delta\omega^b + \mathbf{C}_n^b(t_k)\delta\mathbf{r}^n \\
&= \mathbf{C}_n^b(t_k)\delta\mathbf{r}^n - [\Delta t \mathbf{C}_n^b(t_k)\mathbf{p}_{INS,i}^n(t_{k-1}) \times \mathbf{C}_b^n(t_k)]\delta\omega + \varepsilon_i(t_k)
\end{aligned} \tag{5.17}$$

It can be seen from equation (5.17) that each observed feature contributes three rows to the measurement vector. Similarly, the observation matrix will get an additional three rows for each feature. When inspecting the geometry of knowing the 3D position change of a point or corner in space, if enough features can be seen, position change can be estimated. Therefore, through these features, position displacement error, $\delta\mathbf{r}$, is directly observable. This yields an observation matrix for 3D corners and points where the

contribution of one observed feature is shown in equation (5.18), and will be a matrix of size $3N \times 15$, where N is the number of 3D corners or 3D points.

$$\mathbf{H}_{3\text{Dcorners/points}}(t_k) = \begin{bmatrix} \mathbf{C}_n^b(t_k) & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & -[\Delta t \mathbf{C}_n^b(t_k) \mathbf{p}_{INS,i}^n(t_{k-1}) \times \mathbf{C}_b^n(t_k)] & \mathbf{0}_{3 \times 3} \end{bmatrix} \quad (5.18)$$

5.3 2D Point and Corner Navigation Equations

Just like 3D points and corners, the distinction between 2D point and corner features, is non-existent in the mathematical sense and, therefore, both will be treated the same. Also, assuming that situations will constantly arise in which there are an insufficient number of 2D corners or points to perform a full motion estimate using a technique such as SIFT, it can be stated, from the principles of multi-view point geometry shown in [38], that if only two different frames are compared to observe the motion of a limited number of points in 3D space, a non-ambiguous result cannot be obtained. With only two frames being compared to observe motion, and depth remaining uncertain on each individual frame, 2D points and corners will not provide the necessary observability required to obtain a full navigation solution with respect to their use in this thesis. Although a full navigation solution will not be obtainable from these features, they can be used to form constraint equations that can be implemented in the EKF and will add robustness and reliability to the results by constraining the possible motion that could have been observed by the sensor platform.

To form the constraint equation that will be used to aid the EKF in performing state estimation, we start by defining observed point features in the body frame at a specific time, $\mathbf{p}_i^b(t_k)$, through equation (5.19) by using the unit vector to the point, $\mathbf{e}_i^b(t_k)$, and the range to the point, $\rho_i(t_k)$. The point's unit vector is obtained using equation (5.20) by knowing the point's image frame coordinates (x,y) , the focal length, f, and the principal point (o_x, o_y).

$$\mathbf{p}_i^b(t_k) = \rho_i(t_k) \mathbf{e}_i^b(t_k) \quad (5.19)$$

$$\mathbf{e}_i^b(t_k) = \frac{[(x-o_x)(y-o_y) \ f]}{\text{norm}([(x-o_x)(y-o_y) \ f])} \quad (5.20)$$

Assuming that the observed point in 3D space is stationary, equation (5.14b), can be used once more to find an equation dependent on the inertial measurements. Substituting equation (5.14) into equation (5.19) yields:

$$\begin{aligned} \mathbf{p}_i^b(t_k) &= \mathbf{C}_n^b(t_k) \mathbf{C}_b^n(t_{k-1}) \mathbf{p}_i^b(t_{k-1}) - \mathbf{C}_n^b(t_k) \Delta \mathbf{r}(t_k, t_{k-1}) \Rightarrow \\ \rho_i(t_k) \mathbf{e}_i^b(t_k) &= \mathbf{C}_n^b(t_k) \mathbf{C}_b^n(t_{k-1}) \rho_i(t_{k-1}) \mathbf{e}_i^b(t_{k-1}) - \mathbf{C}_n^b(t_k) \Delta \mathbf{r}(t_k, t_{k-1}) \end{aligned} \quad (5.21)$$

Equation (5.21) represents the 3D change point location from time t_{k-1} to time t_k after undergoing a specific translation and rotation. Note that the vision sensor will provide an estimate of the unit vector, $\mathbf{e}_i^b$, but not of the range ρ_i^b . Defining a new vector, $\mathbf{e}_{i,\perp}^b(t_k)$ as a vector perpendicular to the unit vector $\mathbf{e}_i^b(t_k)$, multiplying both sides of equation (5.21)

by this factor, and expanding the result, yields the constraint equation given in (5.22). It is this constraint equation that will be used for 2D corner and point features.

$$\begin{aligned}
 & \underbrace{\rho_i(t_k) \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{e}_i^b(t_k)}_{=0} \\
 &= \rho_i(t_{k-1}) \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \mathbf{C}_b^n(t_{k-1}) \mathbf{e}_i^b(t_{k-1}) - \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \Delta \mathbf{r} \Rightarrow \\
 & \rho_i(t_{k-1}) \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \mathbf{C}_b^n(t_{k-1}) \mathbf{e}_i^b(t_{k-1}) - \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \Delta \mathbf{r} = 0
 \end{aligned} \tag{5.22}$$

Off course, this **constraint** will not equal zero due to errors in all components of equation (5.22).

$$\mathbf{z}_{2Dpoints} = \underbrace{\hat{\rho}_i(t_{k-1}) \hat{\mathbf{e}}_{i,\perp}^{b,T}(t_k) \hat{\mathbf{C}}_n^b(t_k) \hat{\mathbf{C}}_b^n(t_{k-1}) \hat{\mathbf{e}}_i^b(t_{k-1})}_{\text{See equation (5.18b)}} - \underbrace{\hat{\mathbf{e}}_{i,\perp}^{b,T}(t_k) \hat{\mathbf{C}}_n^b(t_k) \Delta \hat{\mathbf{r}}}_{\text{See equation (5.18a)}} \tag{5.23}$$

Let's break equation (5.23) into two separate terms and substitute the estimates by an error-free parameter and an additional error term. Next, we will expand both terms and neglect all the second-order error terms. Let's start with the second term:

$$\begin{aligned}
 & \{\mathbf{e}_{i,\perp}^{b,T}(t_k) + \delta \mathbf{e}_{i,\perp}\} \mathbf{C}_n^b(t_k) [\mathbf{I} + \mathbf{\Psi}(t_k)] (\Delta \mathbf{r} + \delta \mathbf{r}) \approx \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \Delta \mathbf{r} + \delta \mathbf{e}_{i,\perp} \mathbf{C}_n^b(t_k) \Delta \mathbf{r} + \\
 & \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \mathbf{\Psi}(t_k) \Delta \mathbf{r} + \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \delta \mathbf{r} \\
 & \{\rho_i(t_{k-1}) + \delta \rho_i\} \{\mathbf{e}_{i,\perp}^{b,T}(t_k) + \delta \mathbf{e}_{i,\perp}\} \mathbf{C}_n^b(t_k) [\mathbf{I} + \mathbf{\Psi}(t_k) - \mathbf{\Psi}(t_{k-1})] \mathbf{C}_b^n(t_{k-1}) \{\mathbf{e}_i^b(t_{k-1}) + \delta \mathbf{e}_i\} = \\
 & \rho_i(t_{k-1}) \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \mathbf{C}_b^n(t_{k-1}) \mathbf{e}_i^b(t_{k-1}) + \delta \rho_i \mathbf{e}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \mathbf{C}_b^n(t_{k-1}) \mathbf{e}_i^b(t_{k-1}) +
 \end{aligned} \tag{5.24a}$$

$$\begin{aligned} & \rho_i(t_{k-1})\delta\mathbf{e}_{i,\perp}\mathbf{C}_n^b(t_k)\mathbf{C}_b^n(t_{k-1})\mathbf{e}_i^b(t_{k-1}) + \rho_i(t_{k-1})\Delta t\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)[\dot{\Psi}^n \times]\mathbf{e}_i^n(t_{k-1}) + \\ & \rho_i(t_{k-1})\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\mathbf{C}_b^n(t_{k-1})\delta\mathbf{e}_i \end{aligned} \quad (5.24b)$$

Assuming that the terms due to the camera error are Gaussian or can be approximated to be Gaussian, equations (5.24a) and (5.24b) can be rewritten as:

$$\begin{aligned} & \{\mathbf{e}_{i,\perp}^{b,T}(t_k) + \delta\mathbf{e}_{i,\perp}\}\mathbf{C}_n^b(t_k)[\mathbf{I} + \Psi(t_k)](\Delta\mathbf{r} + \delta\mathbf{r}^n) \approx \\ & \underbrace{\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\Delta\mathbf{r} + \mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\Psi(t_k)\Delta\mathbf{r} + \mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\delta\mathbf{r}^n + \varepsilon(t_k)}_{Errorfree\ term} \approx \\ & \underbrace{\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\Delta\mathbf{r} + \mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)[\Psi^n \times]\Delta\mathbf{r} + \mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\delta\mathbf{r}^n + \varepsilon(t_k)}_{Errorfree\ term} \approx \\ & \underbrace{\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\Delta\mathbf{r}}_{Errorfree\ term} - [\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\Delta\mathbf{r} \times]\Psi^n + \mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\delta\mathbf{r}^n + \varepsilon(t_k) \end{aligned} \quad (5.25a)$$

$$\begin{aligned} & \{\rho_i(t_{k-1}) + \delta\rho_i\}\{\mathbf{e}_{i,\perp}^{b,T}(t_k) + \delta\mathbf{e}_{i,\perp}\}\mathbf{C}_n^b(t_k)[\mathbf{I} + \Psi(t_k) - \Psi(t_{k-1})]\mathbf{C}_b^n(t_{k-1})\{\mathbf{e}_i^b(t_{k-1}) + \delta\mathbf{e}_i\} = \\ & \underbrace{\rho_i(t_{k-1})\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\mathbf{C}_b^n(t_{k-1})\mathbf{e}_i^b(t_{k-1})}_{Errorfree\ term} - \\ & [\rho_i(t_{k-1})\Delta t\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\mathbf{e}_i^n(t_{k-1}) \times \mathbf{C}_b^n(t_k)]\delta\omega^b + \varepsilon(t_k) \end{aligned} \quad (5.25b)$$

Subtracting equation (5.25a) from equation (5.25b), substituting equation (5.22) and rearranging the terms yields:

$$\begin{aligned} z_{2Dpoints} = & -\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\delta\mathbf{r}^n - [\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\Delta\mathbf{r} \times]\Psi^n + \\ & [\rho_i(t_{k-1})\Delta t\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\mathbf{e}_i^n(t_{k-1}) \times \mathbf{C}_b^n(t_k)]\delta\omega^b + \varepsilon(t_k) \end{aligned} \quad (5.26)$$

Note that the “constraint measurement” $z_{2Dpoints}$ is a scalar; hence, each additional two-dimensional point feature adds one more constraint. For the $\mathbf{H}$ -matrix this means that for each addition 2D point feature, one row is added:

$$\mathbf{H}_{2Dpoints}(t_k) =$$

$$\begin{bmatrix} -\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k) & \mathbf{0}_{1 \times 3} & -[\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\Delta\mathbf{r} \times], & [\rho_i(t_{k-1})\Delta t \mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_n^b(t_k)\mathbf{e}_i^n(t_{k-1}) \times \mathbf{C}_b^n(t_k)] & \mathbf{0}_{1 \times 3} \end{bmatrix}$$

(5.27)

Given the above equations for the use of 2D points and 2D corners, it is still necessary to derive processes to calculate $\mathbf{e}_{i,\perp}^{b,T}(\mathbf{t}_k)$, and $\rho_i(t_{k-1})$. To obtain $\mathbf{e}_{i,\perp}^{b,T}(\mathbf{t}_k)$, a two step approach was determined. First by examining the unit vector, $\mathbf{e}_i^b(\mathbf{t}_k)$, pointing at $\mathbf{p}_i^b(\mathbf{t}_k)$, the absolute value of each element in the unit vector is taken. By examining the absolute value of these elements, the index of the smallest term can be identified. i.e. if the unit vector is [0.590,-0.766,0.255], then the smallest element of the absolute value of this vector is .255, which is the 3rd index in the vector. Equation (5.28) is then implemented based on the determined index as follows:

If index = 1:

$$\mathbf{e}_{i,\perp}^{b,T}(t_k) = \frac{[1,0,0] \times \mathbf{e}_i^b(t_k)}{norm([1,0,0] \times \mathbf{e}_i^b(t_k))}$$

If index = 2:

$$\mathbf{e}_{i,\perp}^{b,T}(t_k) = \frac{[0,1,0] \times \mathbf{e}_i^b(t_k)}{norm([0,1,0] \times \mathbf{e}_i^b(t_k))}$$

If index = 3:

$$\mathbf{e}_{i,\perp}^{b,T}(t_k) = \frac{[0,0,1] \times \mathbf{e}_i^b(t_k)}{\text{norm}([0,0,1] \times \mathbf{e}_i^b(t_k))} \quad (5.28)$$

To estimate $\rho_i(t_{k-1})$, the measurements of the platforms motion from the IMU, and the observed change in the unit vector for each feature is used in a least squares method. Defining $\Delta \mathbf{r}$ as the measured position change of the platform, the process is shown as:

$$\begin{aligned} \mathbf{H} &= [\hat{\mathbf{C}}_b^n(t_{k-1}) \mathbf{e}_i^b(t_{k-1}), -\hat{\mathbf{C}}_b^n(t_k) \mathbf{e}_i^b(t_k)] \\ \begin{bmatrix} \rho_i(t_{k-1}) \\ \rho_i(t_k) \end{bmatrix} &= \text{inv}(\mathbf{H}' \mathbf{H}) * \mathbf{H}' \Delta \mathbf{r} \end{aligned} \quad (5.29)$$

5.4 2D and 3D line Navigation Equations

When examining the measurements provided by 3D and 2D line features, a number of observations can be made. The first observation is that due to the extraction process, noise and other errors on the analyzed images, and the nature of line features, the end points of any finite line in 3D space cannot be consistently identified from image to image. Since the endpoints of a line are not consistently identifiable, a consequence of this is that the only point on a line that can be consistently identified is the point on a given line that is closest to the sensor observing it. These points can provide unambiguous information about a platforms translational motion if they are derived from multiple non-parallel lines. With the scope of this thesis focusing on indoor navigation, these points will not be used since line features in an indoor environment are quite often

close to parallel. Therefore, since these points will not be used for this work, no points on a line will be considered consistently identifiable between frames. With this limitation, and only observing motion of a line feature from one frame to another, translational motion will not be observable through line features for this thesis.

Although translational motion will be unobservable, the relationship between rotational motion and 3D line features can still be examined. The unit vector of a 3D line, $\mathbf{l}_i^b(t_k)$, can be accurately and consistently identified between frames. From this, equation (5.30) can be stated to establish the effect of rotational motion on the unit vector of a 3D line. Equation (5.30) can then be manipulated into the form of equation (5.31) by multiplying by a new vector, $\mathbf{l}_{i,\perp}^b(t_k)$, which is a unit vector perpendicular to $\mathbf{l}_i^b(t_k)$ and calculated using the same process as the one used for 2D points. Using equation (5.31), equation (5.32) can be stated, and is the constraint equation that will be used for 3D lines. Re-writing equation (5.32) in terms of the errors it contains, equation (5.33) can be derived. From equation (5.33), the observation matrix for 3D lines, $\mathbf{H}_{3Dlines}$, is established in equation (5.34). The observation matrix for 3D lines will be an Nx15 matrix where N represents the number of 3D line features.

$$\mathbf{l}_i^b(t_k) = \mathbf{C}_n^b(t_k) * \mathbf{C}_b^n(t_{k-1}) \mathbf{l}_i^b(t_{k-1}) \quad (5.30)$$

$$\mathbf{l}_{i,\perp}^{b,T}(t_k) \mathbf{l}_i^b(t_k) = \mathbf{l}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) * \mathbf{C}_b^n(t_{k-1}) \mathbf{l}_i^b(t_{k-1}) \quad (5.31)$$

$$0 = \mathbf{l}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) * \mathbf{C}_b^n(t_{k-1}) \mathbf{l}_i^b(t_{k-1}) - \mathbf{l}_{i,\perp}^{b,T}(t_k) \mathbf{l}_i^b(t_k) \implies$$

$$\mathbf{z}_{3Dlines} = \mathbf{l}_{i,\perp}^{b,T}(t_k) \hat{\mathbf{C}}_n^b(t_k) * \hat{\mathbf{C}}_b^n(t_{k-1}) \mathbf{l}_i^b(t_{k-1}) - \mathbf{l}_{i,\perp}^{b,T}(t_k) \mathbf{l}_i^b(t_k) \quad (5.32)$$

$$\mathbf{z}_{3Dlines} = \Delta t \mathbf{l}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) [\dot{\Psi} \times] \hat{\mathbf{C}}_b^n(t_{k-1}) \mathbf{l}_i^b(t_{k-1})$$

$$\begin{aligned}
&= \Delta t \mathbf{l}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \hat{\mathbf{C}}_b^n(t_{k-1}) \mathbf{l}_i^b(t_{k-1}) \times \boldsymbol{\psi} \Rightarrow \\
&= [\Delta t \mathbf{l}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \hat{\mathbf{C}}_b^n(t_{k-1}) \mathbf{l}_i^b(t_{k-1}) \times] \delta \boldsymbol{\omega} \quad (5.33)
\end{aligned}$$

$$\mathbf{H}_{3Dlines} = \begin{bmatrix} \mathbf{0}_{3 \times 1} & \mathbf{0}_{3 \times 1} & \mathbf{0}_{3 \times 1} & [\Delta t \mathbf{l}_{i,\perp}^{b,T}(t_k) \mathbf{C}_n^b(t_k) \hat{\mathbf{C}}_b^n(t_{k-1}) \mathbf{l}_i^b(t_{k-1}) \times] & \mathbf{0}_{3 \times 1} \end{bmatrix} \quad (5.34)$$

The geometry provided by 2D lines is similar to that of the 3D lines, except depth is not resolved for the 2D line features. Using the equations derived for 3D line features, combined with the absence of depth information for 2D line features, it can be stated that by observing the change in a limited number of 2D line features between two frames, no parameters of the state vector are observable. Therefore, for this thesis, 2D line features will provide no information to the EKF.

When attempting to extract information from the measurements provided by 2D lines, it can be found that association between three frames is needed to obtain any observability of motion. According to [38], if a single 2D line is observed in three different frames, the measurements provided by these three observations can be used to form the constraint equation shown in equation (5.35). This equation provides a third set of measurements allowing for the calculation of $\mathbf{p}$, where $\mathbf{p}$ is a point in 3D space where all three of the different line observations would intersect. For more information on the derivation of equation (5.35), refer to [38].

$$\begin{aligned}
&\mathbf{l}_{i,\perp}^{b,T}(t_k) \left[\Delta \hat{\mathbf{f}}_{INS}(t_{k-1}, t_{k-2}) \mathbf{p}^T \mathbf{C}_n^b(t_{k-1}) * \mathbf{C}_b^n(t_{k-2})^T - \right. \\
&\quad \left. \mathbf{C}_n^b(t_k) * \mathbf{C}_b^n(t_{k-1})^T \mathbf{p} \Delta \hat{\mathbf{f}}_{INS}(t_k, t_{k-1})^T \right] = 0 \quad (5.35)[38]
\end{aligned}$$

5.5 Extended Kalman Filter Equations

With the state vector, measurement equations and observation equations determined previously in sections 5.1-5.4, the remaining pieces of the Kalman filter can be stated and the necessary mechanization can be shown. For the features that yield the measurements necessary to obtain full observability of platform motion, Figure 43 describes how their measurements are used to obtain estimates of the INS drift terms. As stated before, planar surfaces, 3D points, and 3D corners provide the measurements necessary to be used in the mechanization shown in [15]. For 3D line, 2D point, and 2D corner features that provide measurements used to create constraint equations for this thesis, a slightly different mechanization is used in the same Kalman filter and is shown in Figure 44. The implementation of the mechanization shown in Figure 44 will add accuracy and robustness to the output of the Kalman filter, but it cannot obtain a solution in a standalone manner due to unresolvable ambiguities in its solution. Therefore, there must be a sufficient number of planar surfaces, 3D points, or 3D corners to obtain an estimate of the INS error terms in order for the constraint equation mechanization to be used.

Figure 43. Extended Kalman Filter Mechanization

Figure 44. Extended Kalman Filter Mechanization Involving Constraint Equations

Based on the state vector shown in equation (5.1), the process noise matrix, $\mathbf{W}$, can be defined by equation (5.36). The elements of this matrix are defined as, $\sigma_{p,r}^2$, the position error process noise, $\sigma_{p,v}^2$, the velocity error process noise, $\sigma_{p,\psi}^2$, the attitude error process noise, $\sigma_{p,\omega}^2$, the gyro bias process noise, and $\sigma_{p,f}^2$, the accelerometer bias process noise. For the process noise matrix, $\mathbf{I}$ represents a 3x3 identity matrix and $\mathbf{0}$ represent a 3x3 matrix of all 0's.

$$\mathbf{W} = \begin{bmatrix} \sigma_{p,r}^2 \mathbf{I} & 0 & 0 & 0 & 0 \\ 0 & \sigma_{p,v}^2 \mathbf{I} & 0 & 0 & 0 \\ 0 & 0 & \sigma_{p,\psi}^2 \mathbf{I} & 0 & 0 \\ 0 & 0 & 0 & \sigma_{p,\omega}^2 \mathbf{I} & 0 \\ 0 & 0 & 0 & 0 & \sigma_{p,f}^2 \mathbf{I} \end{bmatrix} \quad (5.36)$$

Due to the noise on each measurement collected, the measurement error covariance matrix, $\mathbf{R}$, must be established. The variables in the $\mathbf{R}$ matrix, shown in equation (5.37), are defined as, σ_{planes}^2 , the measurement noise on planar features, $\sigma_{3Dpoints}^2$, the measurement noise on 3D point features, $\sigma_{3Dcorners}^2$, the measurement noise on 3D corner features, $\sigma_{3Dlines}^2$, the measurement noise on 3D line features, $\sigma_{2Dcorners}^2$, the measurement noise on 2D corner features, and $\sigma_{2Dpoints}^2$, the measurement noise on 2D point features. The size of this matrix dynamically changes with each update of the Kalman filter to become a square matrix with the number of rows and columns equal to the total number of rows of the total measurement vector for the current iteration. The $\mathbf{R}$ matrix will have non-zero values on the diagonal shown in equation (5.37) and the rest of the matrix will be populated with 0's. Also, each $\mathbf{I}$ matrix used in equation (5.37) is an identity matrix of size equal to the number of features of the type designated, times the number of rows contributed by each feature to its corresponding measurement vector. i.e. if there are four planes observed at a specific iteration, and each plane contributes four rows to the planar surface measurement vector, then the $\mathbf{I}$ matrix multiplied by σ_{planes}^2 will be a 16x16 identity matrix. Similarly, if there are two 2D corners observed at a specific iteration, the $\mathbf{I}$ matrix multiplied by $\sigma_{2Dcorners}^2$

will be a 2x2 identity matrix since each 2D corner contributes only one row to its measurement vector.

$$\mathbf{R} = \begin{bmatrix} \sigma_{planes}^2 \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_{3D\ points}^2 \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma_{3D\ corners}^2 \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{3D\ lines}^2 \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{2D\ corners}^2 \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{2D\ points}^2 \mathbf{I} \end{bmatrix} \quad (5.37)$$

The calculation of the remaining parameters of this EKF involve the $\mathbf{G}$ matrix, derived as 15x15 identity matrix, and the state transition matrix, $\mathbf{F}$, shown in equation (5.38). In equation (5.38), $\mathbf{f}^n$ represents the current specific forces, $\mathbf{C}_b^n$ is the current body to navigation frame rotation matrix, $\mathbf{I}$ is a 3x3 identity matrix, and $\mathbf{0}$ is a 3x3 matrix of 0's. Using the $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{W}$ matrices, equations (5.39), (5.40), (5.41), and (5.42) can be used to define the transition matrix, Φ , and the system noise covariance matrix, $\mathbf{Q}$.

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{f}^n \times & \mathbf{0} & \mathbf{C}_b^n \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{C}_b^n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (5.38)$$

$$\mathbf{A} = \begin{bmatrix} -\mathbf{F} & \mathbf{G}\mathbf{W}\mathbf{G}' \\ \mathbf{0}_{15 \times 15} & \mathbf{F}' \end{bmatrix} \quad (5.39)$$

$$\mathbf{B} = \text{expm}(\mathbf{A}) = \begin{bmatrix} \mathbf{B}_{\text{upper left}} & \mathbf{B}_{\text{upper right}} \\ \mathbf{B}_{\text{lower left}} & \mathbf{B}_{\text{lower right}} \end{bmatrix} \quad (5.40)$$

$$\Phi = \text{expm}(\mathbf{A}) \quad (5.41)$$

$$\mathbf{Q} = \Phi * \mathbf{B}_{\text{upper right}} \quad (5.42)$$

Using previously stated equations, the remaining parts of the Kalman filter can be established. From the organization of equation (5.37), combined with equations (5.11), (5.17), (5.23), and (5.32), a full calculation of the measurement vector, $\mathbf{z}$, can be stated in equation (5.43). Using equation (5.43), the observation matrix, $\mathbf{H}$, for the entire Kalman filter can be written by combining equations (5.12), (5.18), (5.27), and (5.34). The observation matrix is displayed in equation (5.44). With all the necessary equations in place, the update and prediction equations displayed in equations (5.45), (5.46), (5.47), (5.48), and (5.49), can be used to execute the Kalman filter.

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_{planes} \\ \mathbf{z}_{3D\ points} \\ \mathbf{z}_{3D\ corners} \\ \mathbf{z}_{3D\ lines} \\ \mathbf{z}_{2D\ corners} \\ \mathbf{z}_{2D\ points} \end{bmatrix} \quad (5.43)$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{planes} \\ \mathbf{H}_{3D\ points} \\ \mathbf{H}_{3D\ corners} \\ \mathbf{H}_{3D\ lines} \\ \mathbf{H}_{2D\ corners} \\ \mathbf{H}_{2D\ points} \end{bmatrix} \quad (5.44)$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^t (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R})^{-1} \quad (\text{from equation (2.24)[29]}) \quad (5.45)$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H} \hat{\mathbf{x}}_k^-) \quad (\text{from equation (2.25)[29]}) \quad (5.46)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_k^- \quad (\text{from equation (2.26)[29]}) \quad (5.47)$$

$$\hat{\mathbf{x}}_{k+1}^- = \Phi_k \hat{\mathbf{x}}_k \quad (\text{from equation (2.27)[29]}) \quad (5.48)$$

$$\mathbf{P}_{k+1}^- = \Phi_k \mathbf{P}_k \Phi_k^T + \mathbf{Q} \quad (\text{from equation (2.28)[29]}) \quad (5.49)$$

CHAPTER 6: RESULTS

This chapter discusses the theoretical results and effectiveness of the navigation procedure described in this thesis. To better control all the error sources, the proposed methods are evaluated using a simulated environment created to be similar to the data collection environment discussed in chapter 4. Use of the data-collection data to evaluate the algorithms was omitted in the context of this thesis, since the data collection platform was not sufficiently calibrated to support various evaluations. A new data collection is planned for the future. This approach allowed for testing of the navigational algorithms without the accuracy of the algorithms being affected by the systematic errors in the sensors and data collection platform.

6.1 Simulation Description

In order to test the effectiveness of the navigation algorithms discussed in chapter 5, a simulation was created. First, a hallway environment was made containing observable static features including, points, corners, lines and planar surfaces. Using the positions of each of these features, a path was created through the constructed hallway. This artificial environment is illustrated from multiple angles in Figure 45. After the path was determined, raw IMU measurements were synthesized, at an update rate of 100Hz. The time elapsed from the beginning of the path to the end was ten minutes. This error free IMU data was used to generate plots of position change in the navigation frame, and attitude change in the navigation frame (local), for each of the axes with respect to time. The error free attitude change data, shown in Figure 46, and the error free

position change data, shown in Figure 47, are used to evaluate the performance of the error correcting capabilities of the filter discussed in chapter 5, once errors are injected into the system.

Figure 45. Environment for Navigation Simulation

Figure 46. Error Free Attitude Change Data

Figure 47. Error Free Position Change Data

Using the previously described hallway environment and matching IMU data, an algorithm was developed that determined the position of each feature in the 3D world environment, and the position of corner and point features in the 2D image frame at any given point on the platform path. The 2D image frame was created and projected using a focal length of 6mm and a resolution of 500x500 pixels. This algorithm also implements sensor characteristics such as Field-of-View and maximum range to determine which features contained in the environment would be observable to the platforms sensors. This is accomplished by using a 60-degree Field-of-View, centered around the heading vector of the platform, and a 10 meter maximum range, to determine if a feature would be seen. This process, performed at a 1Hz. rate, yields the capability of determining observed position change of realistically usable features at every second on the platforms path. Using the observed feature motion determined by this procedure, combined with the created IMU data, the performance of the Kalman filtering process described in chapter 5 can be evaluated by introducing known errors into the IMU data and the positions of the features.

6.2 Simulation Errors

6.2.1 Inertial Errors

In order to test the error estimation capabilities of the Kalman filter, an error model was created to introduce bias and random walk errors into the accelerometer and gyro outputs of the IMU data. To create this, specifications of multiple inertial units were used to calculate reasonable error models for two different inertial sensors of varying quality. The values for these error sources can be seen in Table 4.

Table 4. Inertial Error Values.

	Perfect IMU	Tactical IMU	Commercial IMU
Gyro Rate Bias	0 deg/hr.	60deg/hr.	6000deg/hr.
Gyro Measurement Noise	$N(0, 0)$ deg/hr.	$N(0, (0.6)^2)$ deg/hr.	$N(0, (6)^2)$ deg/hr.
Accelerometer Bias	0 mg	1 mg	10 mg
Accelerometer Measurement Noise	$N(0, 0)$ mg	$N(0, 0.1)$ mg	$N(0, 1)$ mg

6.2.2 Feature Errors

Along with having the capability of controlling the inertial errors, the simulation was also designed with the ability to add noise to the parameters that define the features. To give the simulation this capability, three different additive noise error processes were defined. The first process uses Equation (6.1), and describes additive Gaussian noise (in meters) to the location of 3D corners, 3D points, and planar surface centroids, $[x \ y \ z]^T$, in other words, the derived features after the feature extraction and association process.

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} N(0, \sigma) \\ N(0, \sigma) \\ N(0, \sigma) \end{bmatrix} \quad (6.1)$$

The next process, shown in Equation (6.2), allows for the addition of noise in pixels to the image plane location of 2D points and 2D corners, $[x_i y_i]^T$.

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} N(0, \sigma) \\ N(0, \sigma) \end{bmatrix} \quad (6.2)$$

The last process yields the ability to add noise to planar surface normal vectors and 3D line direction vectors, $[i \ j \ k]^T$. This process is described by equation 6.3.

$$\begin{bmatrix} \hat{i} \\ \hat{j} \\ \hat{k} \end{bmatrix} = \frac{\begin{bmatrix} i \\ j \\ k \end{bmatrix} + \begin{bmatrix} N(0, \sigma) \\ N(0, \sigma) \\ N(0, \sigma) \end{bmatrix}}{\text{norm}(\begin{bmatrix} i \\ j \\ k \end{bmatrix} + \begin{bmatrix} N(0, \sigma) \\ N(0, \sigma) \\ N(0, \sigma) \end{bmatrix})} \quad (6.3)$$

Note that in all these cases, only zero-mean additive Gaussian noise was considered. Even though the Gaussian character of the error models is in line with some of the observed error behavior, the determination of the variance and its dependence on various parameters such as range to the feature, lighting conditions, angle of observation, etc has not been studied in detail in this thesis. Furthermore, the zero-mean model

assumption is more than likely invalid, even in nominal conditions due to systematic errors (processing biases, calibration errors etc.).

Along with the extreme simplification of using zero-mean additive Gaussian noise as the only feature position errors, many other errors have been simplified or overlooked as well. When considering the 3D flash lidar, the raw range measurements that it makes contain much more error than just noise. The largest of these errors to be ignored in this simulation is multipath, which inherently makes the ranges to features like corners or lines at the intersection of two orthogonal planes much more erroneous than ranges to a flat planar surface. Another large error not taken into consideration in this simulation is the effect of miss-associations. All of the feature association done for this simulation was perfect, but in a real system miss-associations are a high concern, and can cause serious errors in the navigation solution. To help quantify and characterize a few of these unaddressed errors such as the ones mentioned above, Table 5 was created.

Table 5. Unaddressed Errors

Error Source	Magnitude of error	Can this error be calibrated out or modeled in a real system
3D multipath	1-10 cm per range measurement	The multipath collected by the 3D flash ladar can be modeled with respect to its environment. Extensive modeling and signal processing could be used to attenuate this error, but it can never be fully removed.
Miss-association	1-20cm of false feature motion per miss-association	Intelligent association techniques and modeling could be used to decrease the possibility of a miss-association, but perfect association between frames of all data collection is unattainable.
2D camera calibration error	2D data can become to erroneous to use	By using a detailed camera calibration process, calibration errors of the 2D digital camera can be mitigated.
Lever arm correction	Catastrophic over time	It is impossible to have the 3D flash ladar, the 2D digital camera, and the IMU co-located. Therefore a lever arm correction must be made, or else every motion estimate made by each individual sensor cannot be accurately related to the other sensors. Not making an accurate lever arm correction would make this system to erroneous to use very quickly.

6.3 Simulation Observability

Due to the complementary nature of the Kalman filter described in chapter 5, it is necessary to test that all 15 elements of the state vector are observable. One such test is described in detail in [44], which offers equation (6.4). Equation (6.4) can be implemented using the $\mathbf{F}$ and $\mathbf{H}$ matrices derived in chapter 5, along with the knowledge that n is equal to the number of elements in the state vector. [44] states that if the rank of matrix $\mathbf{M}$ is equal to n , then the state vector is completely observable. Evaluating

equation (6.4) with respect to this system yields a matrix with rank 15, thus proving that the Kalman filter described in chapter 5 contains a completely observable state vector.

$$\mathbf{M} \triangleq [\mathbf{H}^T, \mathbf{F}^T \mathbf{H}^T, \dots (\mathbf{F}^T)^{n-1} \mathbf{H}^T] \quad (6.4)$$

6.4 Simulation Results

To test the effectiveness, accuracy, and limitations of the system described, a set of 25 different experiments were derived and performed. These experiments are shown and described in Table 6. Each row of Table 6 describes a single derived experiment along with the specified system inputs of the grade of the IMU, the 3D and 2D features used, and the magnitude of the noise variance injected onto each feature used. The rest of section 6.4 documents the results of these tests. For each test, a plot is provided that shows the vision-aided navigation systems error, which can be compared to the errors produced by the un-aided tactical and commercial grade IMU's, displayed in Figure 48 and Figure 49 respectively. Nine subplots are shown on each of these plots to show the error produced by the navigation systems for attitude(degrees), velocity(m/s), and position(m) about all three axis.

Table 6. Description of Simulation Experiments.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
1	Tactical Grade	Planes, Lines, Corners/Points	None	Points/Corners	None
2	Commercial Grade	Planes, Lines, Corners/Points	None	Points/Corners	None
3	Tactical Grade	Corners/Points	$(1)^2$ cm	None	None
4	Tactical Grade	Corners/Points	$(1)^2$ dm	None	None
5	Commercial Grade	Corners/Points	$(1)^2$ dm	None	None
6	Tactical Grade	1 Plane	None	Points/Corners	None
7	Tactical Grade	1 Plane	None	Points/Corners	$(1)^2$ pixel
8	Tactical Grade	1 Plane	None	Points/Corners	$(5)^2$ pixel
9	Commercial Grade	1 Plane	None	Points/Corners	$(5)^2$ pixel
10	Commercial Grade	1 Plane, Lines	None	None	None
11	Tactical Grade	1 Plane, Lines	$(0.001)^2$ (direction)	None	None
12	Tactical Grade	1 Plane, Lines	$(0.01)^2$ (direction)	None	None
13	Commercial Grade	1 Plane, Lines	$(0.01)^2$ (direction)	None	None
14	Tactical Grade	Planes	None	None	None
15	Tactical Grade	Planes	$(1)^2$ cm (centroid)	None	None
16	Tactical Grade	Planes	$(1)^2$ dm (centroid)	None	None
17	Commercial Grade	Planes	$(1)^2$ dm (centroid)	None	None
18	Tactical Grade	Planes	$(0.001)^2$ (normal vec)	None	None
19	Tactical Grade	Planes	$(0.01)^2$ (normal vec)	None	None
20	Commercial Grade	Planes	$(0.01)^2$ (normal vec)	None	None
21	Tactical Grade	Planes	$(1)^2$ cm (centroid) $(0.001)^2$ (normal vec)	None	None
22	Tactical Grade	Planes	$(1)^2$ dm (centroid) $(0.01)^2$ (normal vec)	None	None
23	Commercial Grade	Planes	$(1)^2$ m (centroid) $(0.1)^2$ (normal vec)	None	None
24	Tactical Grade	Lines	None	Points/Corners	None
25	Commercial Grade	Lines	None	Points/Corners	None

Figure 48. Results of Un-aided Tactical Grade INS

Figure 49. Results of Un-aided Commercial Grade INS

6.4.1 Experiments 1 and 2

These two tests were run to show the capability of this system using all possible features with no noise on them. Noise was only introduced on the inertial measurements (delta-v's and delta-theta's). Figure 50 displays that if the tactical grade IMU is used, the errors after 10 minutes are bounded to below 1cm for position and less than 0.04 degrees for attitude. If the commercial grade IMU is used, the errors in position are less than 1 m for position and 1.5 degrees for attitude, the results of which can be seen in Figure 51.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
1	Tactical Grade	Planes, Lines, Corners/Points	None	Points/Corners	None

Figure 50. Results of Experiment 1

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
2	Commercial Grade	Planes, Lines, Corners/Points	None	Points/Corners	None

Figure 51. Results of Experiment 2

6.4.2 Experiments 3 through 5

Experiments 3 through 5 were performed with the intent of testing the effect of feature position noise on the designed Kalman filter using only 3D points and corners. Given the tactical grade IMU and a Gaussian noise variance of $(1)^2\text{cm}^2$ on the position of each of the observed features, total errors are bounded to under 10cm for position, 0.02 m/s for velocity, and 1° for attitude. These results can be seen in Figure 52. Using the same IMU and increasing the noise variance to $(1)^2\text{dm}^2$ on the position of each observed feature, the results shown in Figure 53 were obtained. It can be seen in Figure 53 that the total errors never go above (1.1m, 0.15m/s, and 8°). To test the effect of a worse IMU,

the same feature noise variance of $(1)^2 \text{dm}^2$ was used in combination with the commercial grade IMU. The results of this experiment can be seen in Figure 54, showing that for the 10-minute simulation the vision-aided errors never exceed (1.6m, 0.5m/s, and 11°). These results show that the designed algorithms are very tolerant of noise on 3D points and corners.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
3	Tactical Grade	Corners/Points	$(1)^2 \text{cm}^2$	None	None

Figure 52. Results of Experiment 3

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
4	Tactical Grade	Corners/Points	$(1)^2 \text{ dm}^2$	None	None

Figure 53. Results of Experiment 4

5	Commercial Grade	Corners/Points	$(1)^2 \text{ dm}^2$	None	None
---	------------------	----------------	----------------------	------	------

Figure 54. Results of Experiment 5

6.4.3 Experiments 6 through 9

The purpose of experiments 6 through 9 are to test the limitations of the system described in this thesis while using all observable 2D points and corners, and 1 plane at every time step. Figure 55 shows the error correction performance of this system using the tactical grade IMU and having no feature position noise. It can be seen in this graph, and confirmed by equation 6.4 yielding a rank of 15, that a full solution can be obtained under the given constraints. Knowing that a full solution can be obtained, the effect of noise on the 2D point and corner image frame positions was tested. Figure 56 displays the results of experiment 7 in which the tactical grade IMU was used and Gaussian noise with a variance of $(1)^2$ pixel was injected onto the position of each 2D point and corner. It can be observed that the position and velocity corrections performed well for this experiment, limiting the total errors to under 11cm for position and 0.012m/s for velocity. It can also be seen by comparing Figure 55 with Figure 56, that by adding a small amount of error onto the 2D features, the attitude error cannot be estimated well. Although full observability of the state vector is obtained for the experiments in this section, the geometry provided for estimating attitude is very poor, and the given features do not give a robust or accurate way of estimating attitude under noisy conditions. The solution degrades even further in Figure 57, where experiment 7 is performed again except the noise variance on the 2D features is increased to $(5)^2$ pixels to create experiment 8. Figure 57 displays that under the given circumstances the errors for position, velocity and attitude are bounded to under (7m, 0.1m/s, 10°). The results of experiment 8 show that as more noise is added onto the 2D feature positions, the solution degrades quickly.

Similarly, Figure 58 displays the results of experiment 9, which combines the feature noise described in experiment 8 with the commercial grade IMU. This plot shows corrections that bound the errors to (9m, 0.25m/s, 20°). From these experiments it can be stated that under the established circumstances a full solution can be obtained that estimates position and velocity error well if noise is low, but even under low noise it does not provide good geometry to estimate attitude error.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
6	Tactical Grade	1 Plane	None	Points/Corners	None

Figure 55. Results of Experiment 6

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
7	Tactical Grade	1 Plane	None	Points/Corners	$(1)^2$ pixel

Figure 56. Results of Experiment 7

8	Tactical Grade	1 Plane	None	Points/Corners	$(5)^2$ pixel
---	----------------	---------	------	----------------	---------------

Figure 57. Results of Experiment 8

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
9	Commercial Grade	1 Plane	None	Points/Corners	$(5)^2$ pixel

Figure 58. Results of Experiment 9

6.4.4 Experiments 10 through 13

Experiments 10 through 13 were derived to test the limitations of the system described in this thesis while using exactly 1 plane, and all observable 3D lines at every time step. Under these constraints combined with the use of the commercial grade IMU, it can be seen in Figure 59 that if there is no noise on the 3D line direction vectors, the attitude error never exceeds 0.018° . It can also be seen in Figure 59 that the situation described by this section does not yield the capability of accurately estimating corrections for position and velocity. In this situation, the Kalman filter coasts on the inertial data for position and velocity because little to no information is gained by observing either feature. The inability to estimate these values can be explained by implementing equation

(6.4). The rank of the $\mathbf{M}$ matrix produced by these experiments is less than 15, meaning that there are states in the state vector that are not observable. Given the geometry yielded in this situation, these results were as expected. 3D line features provide no information about position change of the platform, and observing 1 planar surface does not provide enough data to make a position change calculation. Due to this, the results in Figure 59 verify the hypothesis that in the given circumstances, attitude error can be estimated well, but position and velocity errors cannot be estimated at all.

Proving that this situation can only be used to correct attitude, the effect of feature noise on the direction vectors of 3D lines was tested. Figure 60 shows the results of using the tactical grade IMU and injecting a Gaussian error with variance $(0.001)^2$ on the unit direction vector of the line features. Figure 60 shows that with this error, attitude error is bounded to under 1.5° . Increasing the variance of the line direction error to $(0.01)^2$ provides the results shown in Figure 61. It can be seen in Figure 61 that the attitude error is not estimated well. Although still erroneous, yaw and roll are both corrected to be quite a bit better than the un-aided solution, but the pitch error is almost as bad as the original un-aided errors. It can be seen though in Figure 62 that with the commercial grade IMU and the same noise on the observed features, the aided solution provides much more accurate attitude estimation than the unaided solution. Therefore, it can be stated that although noise on the 3D line features can significantly decrease the accuracy of the error estimation capabilities of this system, their effectiveness with respect to the unaided solution is also relative to the inertial used.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
10	Commercial Grade	1 Plane, Lines	.None	None	None

Figure 59. Results of Experiment 10

11	Tactical Grade	1 Plane, Lines	$(0.001)^2$ (direction)	None	None
----	----------------	----------------	-------------------------	------	------

Figure 60. Results of Experiment 11

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
12	Tactical Grade	1 Plane, Lines	$(0.01)^2$ (direction)	None	None

Figure 61. Results of Experiment 12

13	Commercial Grade	1 Plane, Lines	$(0.01)^2$ (direction)	None	None
----	------------------	----------------	------------------------	------	------

Figure 62. Results of Experiment 13

6.4.5 Experiments 14 through 23

To test the effect of noise on the parameters of the centroid location and normal vector direction for planar surfaces, experiments 14 through 23 were performed. For all of these experiments 3D planar surfaces were the only features used in the navigation equations. The first experiment performed used the tactical grade IMU, and no noise was placed on any parts of the planar surfaces. The results of this experiment are displayed in Figure 63, and it can be seen that the errors produced by the aided solution never exceed $(0.16\text{m}, 0.016\text{m/s}, \text{ and } 0.023^\circ)$. From these results, combined with equation (6.4) yielding an $\mathbf{M}$ matrix of rank 15 for the given circumstances, it can be stated that by using only 3D planar surfaces the state vector for this thesis is fully observable.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
14	Tactical Grade	Planes	None	None	None

Figure 63. Results of Experiment 14

Experiments 15 through 17 test the effect of noise on the centroid location of each of the observed planar surfaces. The first experiment uses the tactical grade IMU and adds noise with a variance of $(1)^2\text{cm}$ to the location of the each planar surfaces centroid. Figure 64 shows the results of this experiment, which produces errors bounded to $(0.42\text{m}, 0.017\text{m/s}, \text{ and } 0.024^\circ)$. Experiment 16 uses the tactical grade IMU with a noise variance of $(1)^2\text{dm}$, the results of which are displayed in Figure 65. The errors observed in experiment 16 are shown in Figure 65 to never exceed $(3.1\text{m}, 0.07\text{m/s}, \text{ and } 0.04^\circ)$. With the same noise variance of $(1)^2\text{dm}$ combined with the use of the commercial grade IMU, experiment 17 was performed. The results of this experiment, shown in Figure 66, yields errors that never exceed $(9.5\text{m}, 0.4\text{m/s}, \text{ and } 2.2^\circ)$. These experiments show that even under high noise on the centroid position, the inertial errors are corrected well with respect to the un-aided solution.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
15	Tactical Grade	Planes	$(1)^2\text{ cm (centroid)}$	None	None

Figure 64. Results of Experiment 15

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
16	Tactical Grade	Planes	$(1)^2\text{dm}$ (centroid)	None	None

Figure 65. Results of Experiment 16

17	Commercial Grade	Planes	$(1)^2\text{dm}$ (centroid)	None	None
----	------------------	--------	-----------------------------	------	------

Figure 66. Results of Experiment 17

Experiments 18 through 20 were designed in a similar fashion to experiments 15 through 17 except noise was placed on the normal vector instead of the centroid. Figure 67 displays the results of experiment 18 which uses the tactical grade IMU and injects noise onto the normal vector with a variance of $(0.001)^2$. Experiment 19 uses the same IMU, but increases the noise variance on the normal vector to $(0.01)^2$, the results of which can be seen in Figure 68. Figure 69 displays the results of experiment 20, which uses the same noise level on the unit vector as experiment 19, but uses the commercial grade IMU. By examining the results of these test, it can be seen that they follow a similar pattern to the results of experiments 15 through 17. Experiments 18 through 20 produce errors that are no greater than $(0.23\text{m}, 0.005\text{m/s}, \text{ and } 0.2^\circ)$, $(2.1\text{m}, 0.02\text{m/s}, \text{ and } 1.4^\circ)$, and $(7\text{m}, 0.31\text{m/s}, \text{ and } 3.5^\circ)$ respectively. These results show that when using only planar surfaces, the designed Kalman filters state estimation capabilities still perform well when observing high noise on the planar surface normal vectors.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
18	Tactical Grade	Planes	$(0.001)^2$ (normal vec)	None	None

Figure 67. Results of Experiment 18

19	Tactical Grade	Planes	$(0.01)^2$ (normal vec)	None	None
----	----------------	--------	-------------------------	------	------

Figure 68. Results of Experiment 19

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
20	Commercial Grade	Planes	$(0.01)^2$ (normal vec)	None	None

Figure 69. Results of Experiment 20

The final three experiments of this section were done to show the effect of simultaneous noise on both the centroid and normal vector of each planar surface. Experiment 21 uses the tactical grade IMU, and injects noise onto the centroid location with a variance of $(1)^2\text{cm}^2$ and onto the normal vector with a variance of $(0.001)^2$. Experiment 22 uses the same IMU and increases the noise variances to $(1)^2\text{dm}^2$ on the centroid location and $(0.01)^2$ on the normal vector. The final experiment in this section uses the commercial grade IMU and injects a tremendous amount of noise onto the centroid and normal vector with variances of $(1)^2\text{m}^2$ and $(.1)^2$, respectively. As expected, the results shown in Figure 70, Figure 71, and Figure 72 show that when error is added onto both the centroid and normal vector simultaneously, the errors generated by the

aided navigation system are compounded compared to when they were added separately, but the navigation solution is still highly tolerant of unit vector and centroid noise.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
21	Tactical Grade	Planes	$(1)^2 \text{cm}^2$ (centroid) $(0.001)^2$ (normal vec)	None	None

Figure 70. Results of Experiment 21

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
22	Tactical Grade	Planes	$(1)^2 \text{dm}^2$ (centroid) $(0.01)^2$ (normal vec)	None	None

Figure 71. Results of Experiment 22

23	Commercial Grade	Planes	$(1)^2 \text{m}^2$ (centroid) $(0.1)^2$ (normal vec)	None	None
----	------------------	--------	---	------	------

Figure 72. Results of Experiment 23

6.4.6 Experiments 24 and 25

The final two experiments, the results of which are shown below in Figure 73 and Figure 74, were designed with the intent of testing the effectiveness of the algorithms described using only 3D lines, 2D points and 2D corners. Figure 73 displays the results of using a tactical grade IMU combined with the given features with no error on their positions. It can be seen that by using the corrections provided under the described constraints, attitude is estimated to the same magnitude error as the IMU with no correction, and the velocity and position solutions are estimated poorly as well. This can be explained using equation (6.4), displaying that under the given circumstances the rank of the $\mathbf{M}$ matrix is less than 15, meaning that there are unobservable states. Also, as mentioned previously, the depth is unresolvable for the 2D features and can only be estimated. This depth estimation is performed using the IMU data to synthesize a baseline between consecutive observations of a single feature, and since the IMU data contains error, this estimation is not accurate. The combination of having unobservable states and erroneous distance estimation yields an ineffective solution. To reiterate the effect of the unobservable states and unresolvable depth, experiment 25 was created. The results of this experiment can be seen in Figure 74, and show how poorly the inertial errors are estimated if a commercial grade IMU is used. Although the attitude is slightly corrected with respect to the un-aided solution due to the 3D lines, it is still very poor because of the 2D features. The velocity and position errors shown in Figure 74 are on the same magnitude as the un-aided solution. The results of these two experiments show that a

reliable navigation solution **cannot** be obtained unless 1 or more planes, 3D corners, or 3D points can be observed as well.

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
24	Tactical Grade	Lines	None	Points/Corners	None

Figure 73. Results of Experiment 24

Experiment Number	Inertial Unit Used	3D Features Used	3D Features Noise Variance	2D Features Used	2D Features Noise Variance
25	Commercial Grade	Lines	None	Points/Corners	None

Figure 74. Results of Experiment 25

CHAPTER 7: CONCLUSION

This thesis presented the motivation behind and the development of an image aided inertial navigation system specifically designed for indoor environments where GPS struggles. Multiple different navigation techniques and sensor combinations, discussed in chapters 2 and 3, were explored and studied before the approach discussed in this paper was determined. The chosen combination of 3D flash lidar, 2D digital imaging, and IMU data proved to be a novel concept that had not been thoroughly investigated throughout the available literature. Many parts of this process were extended from the work of others such as the use of planar surfaces for navigation discussed in [10], the extraction and association of point features presented in [25], and the extraction of corner features demonstrated in [41], but the combination of these ideas had never been formerly established.

The methods used for each step of this procedure were discussed in chapter 4 and chapter 5. Algorithms for collecting and organizing the data from the sensors were established. Advanced image-processing techniques were derived and implemented on the imaging data for feature extraction of planar surface, line, point and corner features. Through calculations and random testing on the feature extraction information, different algorithms were created to perform feature association accurately between image frames. The necessary equations and combination of the imaging data with the INS were derived and explained. This established the foundation necessary to use this data in a tightly coupled complementary extended Kalman filter to estimate the chosen error states of the

INS. The estimation of these error states yielded the capability of removing them from the INS data and attenuating the growth of its inherent errors.

Although the Kalman filter was not applied to the collected data, the navigation equations and procedure were theoretically tested and evaluated through a developed simulation discussed in chapter 6. A feature rich artificial environment was made, and IMU data was synthesized to move a platform through this environment. The platform's observations of the environment's features were obtained and used in the EKF. This simulation allowed for testing of the filters robustness, accuracy, and limitations of specific situations of interest. By examining these situations, a few statements can be stated about the process described by this thesis:

- If a vast amount of 2D and 3D features can be accurately observed over time, an accurate solution can be obtained with respect to the stand-alone INS solution.
- Given only planes, 3D corners, or 3D points a full navigation solution can be obtained using the derived equations.
- If only 3D lines are used, attitude can be accurately corrected, but under the derived equations position and velocity continue to drift at a similar rate to the stand alone INS solution.
- 2D points and corners contain an unresolvable ambiguity and cannot be used to obtain a navigation solution within the equations used.
- If multiple 2D points or corners can be accurately observed, a full solution can be obtained given only one 3D planar surface, corner, or point.

In conclusion, the entire process necessary for the use of the procedure described in this paper was established and theoretically proven to work. With the results of the feature extraction and association steps, combined with the results of the simulation, it can be stated that if more accurate sensors were used to collect data, the procedure described in this thesis provides an effective way of obtaining highly accurate and reliable inertial navigation measurements.

CHAPTER 8: FUTURE WORK

The fundamental problem for this work was to find an accurate navigation system that operates reliably in an indoor environment, which is challenging for GPS. The ideas presented in this paper address one possible solution to this problem. Although very promising results were obtained through simulation, the system presented here does not fully solve the problem. For this approach to be applied in practice, several areas of research and technology must be advanced.

Since the navigational algorithms discussed in this paper were only tested through simulated data, it is crucial that the next step in this research is to evaluate this system's performance with real data. Although the simulation provides a way of theoretically validating the equations implemented by the Kalman filter, it does not provide a true metric of the accuracy and reliability of this process. Until this system is tested using obtained sensor data, it can only be stated that this systems "theoretically works". It's actual use is yet to be verified.

Another thing that must be developed and analyzed is a real time version of the algorithms discussed. Since the image processing results shown in this thesis are obtained from a post-processing standpoint, they cannot be used to truly evaluate the effectiveness of this system. In creating a real time system, the computational requirement becomes an issue. In order to deal with the limitations of computational resources, the algorithms presented here must be made more efficient and/or parallelized. Implementing these algorithms in real time may require converting the algorithms to

another language, such as C or C++, or upgrading the hardware on the sensor platform to increase the systems computational power.

Along with making the algorithms discussed in the thesis more computationally efficient, they could also be altered or expanded to be more intelligent and adaptable. One portion of this process that must be altered is the choosing of thresholds for feature extraction and association. For this work, these hard thresholds were empirically set specifically for the data collected. Although they worked well to show results for this thesis, they would likely perform poorly in many other situations with different environment characteristics such as lighting, and feature ranges. Due to this, for reliable indoor navigation, these thresholds must be made adaptable based on the observed environment. Significant testing would have to be done to analyze the effect of different environments on the association and extraction processes in order to create intelligently adapting thresholds.

A separate algorithm that could be added into this process would involve taking the indoor environment into consideration to add constraints onto the possible navigation solution. This could be done by implementing elements from a SLAM algorithm to perform environment mapping. By constantly mapping the environment around the sensors, we could add motion constraints which would assure that the navigation solution output by the sensor platform did not travel through aspects of the map such as known walls or the ceiling. This would be especially useful based on the fact that the errors in the output of this system grow with time and will eventually, given enough time elapsed, put the navigation solution through a media in which the platform cannot physically

travel. If this were to be implemented, the possible error in the systems output could be bounded to a maximum, and therefore make this system reliable for longer amounts of time.

Another aspect of the system that could be significantly improved is further development of the sensors. Although sensors such as the Point Grey Firefly are reasonably cheap and highly accurate, the HG1930 IMU and the SwissRanger 3000 are much more expensive and still contain significant errors. As discussed previously, IMUs inherently contain errors, and in most cases, the more accurate the IMU the more expensive the sensor. As technology improves on these units, obtaining increasingly accurate IMU's will become more affordable, and therefore make IMU's that are more accurate than the HG1930 a more viable option for a system such as the one discussed in this paper. There is also significant room for improvement in the development of 3D imaging sensors such as the SwissRanger. These sensors are relatively immature in their development, and because of this the most accurate and useable versions are orders of magnitude more expensive than the SwissRanger. These more expensive sensors have highly accurate depth resolution, longer ranging capabilities, wider fields of view, and higher resolution. As the technology improves for 3D imagers, these higher quality sensors will become much more reasonable in price, and therefore a more viable option for a navigation system. If vision aided navigation solutions are to be used as a compliment for GPS, it is absolutely necessary to continue advanced research on the sensors used to increase their quality and decrease their price.

One further suggested area of research is to experiment with different sensor combinations. Some of these experiments would include testing the effects of adding another camera to create stereovision for resolving depths in 2D images, and adding one or multiple 2D laser scanners. Finding the optimal combination of types of sensors could significantly improve the performance of this system.

REFERENCES

- [1] John A. Volpe National Transportation Systems Center, “Vulnerability Assessment of the Transportation Infrastructure Relying on the Global Positioning System,” Final Report, Department of Transportation, 29 August 2001.
- [2] M. Uijt de Haag, D. Venable, M. Smearcheck, “Use of 3D laser radar for navigation of unmanned aerial and ground vehicles in urban and indoor environments,” *Proceedings of the SPIE - Volume 6550, SPIE Defense and Security Symposium*, Orlando, FL, April 9- 13, 2007.
- [3] Canny, John, “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, November 1986.
- [4] Marr,D., Hildreth, E., “Theory of Edge Detection,” *Proceedings of the Royal Society of London Series B.*, Biological Sciences Vol. 207, No. 1167, pp.187-217.
- [5] Harris, C., and Stephens, M.J., “A Combined Corner and Edge Detector,” *Alvey Vision Conference*, 1988.
- [6] Hough, P. V. C., “Machine Analysis of Bubble Chamber Pictures,” *Proceedings of the International Conference on High Energy Accelerators and Instrumentation*, 1959.
- [7] Lowe, D. “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.
- [8] Bay, H., T. Tuytelaars, L. van Gool, “Surf: Speeded Up Robust Features,” *Proceedings of the ECCV*, 2006.
- [9] Nguyen, V. et al., “A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics, *Proceedings of the Conference on Intelligent Robots and Systems, IROS 2005*, Edmonton, Canada.
- [10] Venable, D., “Implementation of a 3D Imaging Sensor Aided Inertial Measurement Unit Navigation System,” M.S.E.E. Thesis, Ohio University, 2008.
- [11] Farrell, J.L., “GNSS Aided Navigation and Tracking,” American Literary Press, 2007.
- [12] Parkinson, B.W., Gilbert, S.W. “NAVSTAR: Global positioning system—Ten years later,” *Proceedings of the IEEE*, 1983.
- [13] Misra, P., Burke, B.P., Pratt, M.M., “GPS Performance in Navigation,” *Proceedings of the IEEE*, 1999.

- [14] Massachusetts Institute of Technology,
<http://dspace.mit.edu/bitstream/handle/1721.1/39820/12-215Fall-2002/OcwWeb/Earth--Atmospheric--and-Planetary-Sciences/12-215Modern-NavigationFall2002/CourseHome/index.htm>, 2007.
- [15] Uijt de Haag, M., Inertial Navigation Notes, Ohio University, Athens, Ohio, 2009.
- [16] Titterton, D. H. and Weston J. L., "Strapdown Inertial Navigation Technology, Second Edition", Stevenage, UK: The American Institute of Aeronautics and Astronautics, Reston, Virginia, USA and The Institute of Electrical Engineers, 2004.
- [17] Sinopoli, B., Micheli, M., Donato, G., Koo, T.J., "Vision Based Navigation for an Unmanned Aerial Vehicle," UC Berkeley, Berkeley, California, 2001.
- [18] Hrabar, S., Sukhatme, G.S., "A Comparison of Two Camera Configurations For Optic-Flow Based Navigation of a UAV Through Urban Canyons", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.
- [19] Beardsley, P.A., Zisserman, A., Murray, D.W., "Sequential Updating of Projective and Affine Structure from Motion," *International Journal of Computer Vision* 23(3), 1997.
- [20] SCHECHNER, Y.Y., Kiryati, N., "Depth from Defocus vs. Stereo: How Different Really Are They?," *International Journal of Computer Vision* 39(2), 2000.
- [21] Lucas, B.D., Kande, T., "An Iterative Image Registration Technique with an Application to Stereo Vision," Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1981.
- [22] Carnegie Mellon University,
<http://www.cs.cmu.edu/afs/cs/usr/webb/html/tr.figure.id.8.gif>, July 14, 2009.
- [23] Beardsley, P.A., Zisserman, A., Murray, D.W., "Sequential Updating of Projective and Affine Structure from Motion," *International Journal of Computer Vision* 23(3), 1997.
- [24] Durrant-Whyte, H., Bailly, T., "Simultaneous Localization and Mapping: Part I," *IEEE*, 2006.
- [25] Lowe, D.G. "Object Recognition from Local Scale-Invariant Features," Computer Science Department, University of British Columbia, Vancouver, Canada, 1999.
- [26] Beis, J.S., Lowe, D.G., "Indexing without Invariants in 3D Object Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, 1999.

- [27] Brown, R. G. and Hwang, P. Y. C., "Introduction to Random Signals and Applied Kalman Filtering, Third Edition," John Wiley & Sons, Inc., New York, 1997.
- [28] Veth, M., "Fusion of Imaging and Inertial Sensors for Navigation," Ph.D Dissertation, Air Force Institute of Technology, September 2006.
- [29] Uijt de Haag, M., Integrated Navigation Notes, Ohio University, Athens, Ohio, 2009.
- [30] Gebre-Egziabher, "What is the Difference Between 'Loose', 'Tight', 'Ultra-tight', and 'Deep' Integration strategies for INS and GNSS?," *Inside GNSS*, January/February, 2007.
- [31] Guðmundsson, S. A., "Robot Vision Applications using the CSEM SwissRanger Camera," M.S. thesis, Technical University of Denmark, Denmark 2006.
- [32] Mesa Imaging AG, "SR-3000 Data Sheet," January, 2008.
- [33] Point Grey Research, "Firefly MV IEEE-1394 Digital Camera System Getting Started Manual," April 12, 2007.
- [34] Axis Communications,
http://www.axis.com/products/video/camera/progressive_scan.htm.
- [35] Honeywell International Inc, "Gun Hard low-cost MEMS IMU for the Most Demanding Projectile Guidance Requirements," June, 2006.
- [36] Honeywell International Inc,
<https://commerce.honeywell.com/wcsstore/B2BDirectMyAerospaceAssetStore/images/catalog/HG1930%20113%20x%20107.jpg>.
- [37] Faugeras, O.D., Luong, Q.T., and Maybank, S.J., "Camera Self-Calibration: Theory and Experiments."
- [38] Ma, Y., Soatto, S., Kosecka, J., and Sastry, S.S., "An Invitation to 3-D Vision", Springer Science and Business Media, Inc., New York, 2004.
- [39] Bouguet, J.Y., <http://www.vision.caltech.edu/bouguetj/index.html>, California Institute of Technology, June, 2008.
- [40] Mallat, S., "A Wavelet Tour of Signal Processing," Academic Press, January, 1998.

- [41] Rosten, E., Drummond, T., "Machine Learning for High-Speed Corner Detection," Department of Engineering, Cambridge University, UK, 2006.
- [42] Rosten, E., Drummond, T., "Fusing points and lines for performance tracking," *10th IEEE International Conference on Computer Vision*. Volume 2, Beijing, China, Springer, 2005.
- [43] Donoho, D.L., "De-Noising by Soft-Thresholding," *IEEE Trans. on Information Theory*, vol.41, 1995.
- [44] Maybeck, P.S., "Stochastic Models, Estimation, and Control," Academic Press, vol. 1, 1979.