A Robust INS/PnP Fusion Method for GNSS-Denied Navigation of Small UAVs

A DISSERTATION

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Chen-Chi Chu

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Doctor of Philosophy

Advisor: Demoz Gebre-Egziabher

March, 2014

Acknowledgements

There are many people I would like to pay my deepest gratitude to who made this dissertation possible. Foremost, I would like to thank my thesis advisor Prof. Demoz Gebre-Egziabher for his guidance and invaluable advices throughout my PhD years. I sincerely appreciate the freedom he gave me for exploring the research direction and searching for my interests. His insightful comments on my research have always inspired me to seek out and discover the uncharted territory. I enjoyed having discussion with him and was always motivated by his enthusiasm in navigation research. I also want to thank for his careful revision of my dissertation and papers. Last but not least, I can focus on my PhD research because of his financial support throughout the years. I owe him a great debt.

I would like to acknowledge my committee members Prof. Gary Balas, Prof. Nikolaos Papanikolopoulos and Prof. Peter Seiler for their constructive comments and suggestions which make this dissertation complete. I would also like to thank my Preliminary Oral Exam committee member Prof. Yiyuao Zhao who shared the words of wisdom and life experiences with me.

I owe my colleagues at University of Minnesota a debt of gratitude: Zhefeng Li, Fidelis Adhika Pradipta Lie, Hamid Mokhtarzadeh, Zhiqiang Xing, Guijing Zheng and Susmita Bhattacharyya. I deeply appreciate their friendship and enjoy the insightful discussions with them.

I am thankful to my dearest friends (Matt Chao-Lun Mai, Andy Yu-Chi Liang and Katherine Yi-Yin Liu) who I can shared joys and tears with throughout five years in U.S. They are always the inspirations in my life. Also, I owe special thanks to Sharon Hsi-Jung Lin who always standing by my side and be patient with me.

None of this would have been possible without the support of my family. I would like to acknowledge members in my family, my grandmother, uncles, aunts and cousins, for their strong faith in me. From the bottom of my heart, I would like to express my gratitude to my parents, Ker-Jean Chu and Lee-Chen Yu. Their unconditional love and sacrifice fulfill my dream and make this journey came true. Words simply cannot express my deepest appreciation to my dearest parents. I would like to dedicate this dissertation to them.

Lastly, I am grateful to the individuals who inspired me. Without them, I would not be the person who I am and definitely wouldn't have the opportunity to experience such an amazing journey.

Abstract

This thesis presents three INS/VISNAV integration architectures to deal with problem when operating UAVs. The first two integration architectures are called tight and loose INS/VISNAV integration, respectively. The tight and loose integration strategies are not new and have been used before. The tight approach fuses INS information with camera measurements at pixel level. It is considered as an optimal approach in terms of estimation accuracy. However, it has a tendency to diverge under certain conditions: (1) Unfavorable relative geometry between the camera and feature points used to construct the VISANV solution and (2) Large errors in the position and attitude solution about which the tight integration measurement equations are linearized. This latter condition can occur when VISNAV updates are infrequent or spaced far apart in time. Maintaining stability of the filter that fuses camera and INS information can be challenging when low quality (consumer/automotive grade) inertial sensors are used or the measurement update is less frequent. The loose integration approach is proposed as an alternative for solving the divergent problem.

The integration is loose in the sense that integration occurs at the level of position and attitude. This is in contrast to tight integration where information fusion occurs at the pixel level. While it is sub-optimal from a filtering point of view, the loose integration approach can be more robust to linearization errors which lead to solution divergence. A method for computing the covariance of position and attitude estimates of the VISNAV solution is presented.

The complementary advantages of loose and tight integration leads to the development of a novel, third sensor fusion methodology that enhances the robustness (here defined as resistance to divergence) of filters used to mechanize camera-aided inertial

navigation systems (INS) while preserving the estimation accuracy. This third approach is a hybrid filter that can switch between the optimal tight and suboptimal loose strategies "on the fly" depending on the geometry of the landmarks being tracked and the quality of the inertial sensor. The fusion strategy is based on dual hypothesis testing approach. The proposed approach has the advantages of enhancing the robustness while maintaining the estimation accuracy.

Contents

List of Tables

List of Figures

Chapter 1

Overview

This thesis investigates a backup navigation method for small unmanned aerial systems (SUAS) in the case of unavailability or denial of Global Navigation Satellite Systems (GNSS) services. The term UAS as used in this thesis is taken to mean a system consisting of an unmanned aerial vehicle (UAV) and associated support systems including a ground station and data link. UAVs span a wide range in size and complexity, where some of the largest ones such as the Global Hawk, Predator or Reaper, weigh several thousand pounds and are large enough to require the use of an airport for launch and recovery operations. For the applications considered here, the UAVs in question are small and in the 5-to-10-lb weight range. We will refer to these vehicles as small UAV or SUAV hereafter. SUAVs are small enough to fit in the trunk of a law-enforcement squad car for easy transportation. Therefore, they do not require large infrastructure for launch and recovery operations. They can be operated remotely by a human operator or autonomously by an automatic pilot. It is not uncommon for many SUAS that can operate autonomously during a major part of their mission to nevertheless require a human operator for the launch and recovery phase of operations. It is also not uncommon for many SUAS to be equipped with optical sensors (e.g. infrared/optical

cameras) for operational purpose. In this thesis we will discuss the feasibility of utilizing the onboard camera as alternative navigational sensor for autonomous SUAS operations in GNSS-Denied environments. To motivate this we will first discuss GNSS and GNSS vulnerability in the next sections.

1.1 Global Navigation Satellite Systems

Global Navigation Satellite Systems (GNSS) receivers require information from satellites to generate a navigation solution. In general, the GNSS consist of constellations of medium earth orbiting (MEO) satellites designed to provide free positioning and timing services for terrestrial and low orbit space users. The principle of GNSS positioning is based on a navigation technique known as multilateration. Given distances to at least three references at known locations, a user can unambiguously solve its own position. It is because the observer must be located on a sphere with a radius equals to the distance between the reference and the observer. With two measurement the observer lies on the intersection of the two spheres, which is a circle. When three (or more) spheres intersect with each other, three (or more) circles are defined and the position of the observer is uniquely determined on the intersection of all circles.

The GNSS receiver estimates its distance to GNSS satellites through measuring the time of travel of radio signals. The time of travel multiplied by the speed of light generates the distance measurement known as pseudo-range. By measuring the pseudo-ranges to the satellites at known locations, the receiver calculates its position using the multilateration approach as mentioned.

Currently, the most widely recognized operational GNSS system is the Global Positioning System (GPS). It consists 24^{+} satellites orbiting earth at 20200 km above surface with approximately 12 hours period. The system was originally designed and operated by U.S. Department of Defense for military purposes. It is then released for

civilian uses which creates a wide spectrum of the position sensitive applications. In addition to GPS, there are three GNSS constellations in operation or preparing to be commissioned. GLONASS is the constellation developed and maintained by the Russian Federation. The system was fully operational in the past and now only provides regional coverage due to the limitation of available satellites. Galileo constellation is the effort of the European Union. The first two of four satellites were launched to validate the system in October 2011. Full completion of the constellation is expected by 2019. China has its own GNSS constellation, COMPASS (or Beidou). It became operational in December 2011 with 10 satellites in orbit, providing service to China. In December 2012, it started providing converge in Asia-Pacific region and it is planned to provide global coverage in 2020.

In the absence of interference and obstructions, GNSS provides a navigation solution which is unparalleled in accuracy by most other navigation systems. In conventional use, GNSS provide an estimate of a users position and velocity only. Multiple GNSS receiver-antenna pairs can be used to generate a users attitude estimate. But the update rates of GNSS are not as high and are normally between 1 and 10 Hz [2]. Integrating GNSS with other sensors without this short coming is one way to success. In this regard, the integration of Inertial Navigation Systems (INS) with GNSS is a widely used approach. This approach has been widely deployed in both high end and low end navigation systems. Different integration architectures has been proposed and studied over the years [3, 4]. The fact that GNSS measurement errors are bounded is the underlying reason for the success of this integration scheme.

1.2 The GNSS-Denied Problem

Nonetheless, INS/GNSS integration still has the shortcoming in that GNSS signals are vulnerable to signal loss or degradation. In spite of the deployment of multiple

GNSS constellation, the navigation services they provide can be affected by malicious or un-intentional radio-frequency interference. For example, signal obstruction in urban canyons, valleys, and beneath dense foliage limits the utility of GNSS. The GNSS signal quality can be affected by three environmental factors: Radio frequency (RF) interferences or jamming, physical obstructions or signal reflection and atmospheric disturbances. The following paragraphs briefly discuss their impacts on the GNSS signals.

1.2.1 Radio Frequency (RF) Interferences or Jamming

The GNSS satellites transmit L-band (1-2 GHz range) RF signals from the space approximately 20,000 km away. As the signal reaches the Earth's surface, the power is on the order of 10^{-16} watts which is significantly below the natural noise floor. Depending on bandwidth of the receiver, the noise can be 60-400 (2 MHz bandwidth) to 600-4000 (20 MHz bandwidth) times stronger than GNSS signal [2]. The signal is essentially unobservable by any receivers prior to the special code correlation processing technique. The means that any intentional or unintentional RF emission within this band can easily block the satellite's signal. This becoming a growing concern due to the increasing demand in RF spectrums for high speed communication or broadcasting such as was demonstrated recently by the LightSquared [5] issue. For the past few years, the hard lesson learned is the extreme vulnerability of the GNSS signal. While RF interference can threat the GNSS signal, there is a bigger concern to the GNSS, called GNSS spoofer. The GNSS spoofer is a transmitter which deliberately broadcasts fake GNSS signal to "fool" the GNSS receiver. The receiver that uses spoofer's signal for position computation will report the erroneous position solution.

1.2.2 Physical obstructions or Signal Reflection

GNSS positioning is based on measuring the line of sight (LOS) distance from the receiver to GNSS satellites. When the receiver is operated in urban environments as shown in Figure 1.1(a), the LOS signal from the satellites can be partially blocked or reflected by the building structures causing the loss of signal or incorrect LOS measurement (known as multi-path problem). The problem is even more difficult to tackle if the receiver is operated in indoor environments (Figure 1.1(b)) which has no visibility to the satellites causing severe signal attenuation while penetrating floors or ceilings to reach the receiver. The urban and indoor environments post a significant challenge and is still unsolved for both stand-alone GNSS or INS/GNSS integration applications.

(a) Urban Environment (b) Indoor Environment

Figure 1.1: Urban Canyon and Indoor Problems

1.2.3 Atmospheric Disturbances

The atmospheric disturbances can be regarded as unintentional interference caused by nature. As the signal reaches the receiver, it passes through three different layers of the atmosphere: ionosphere, stratosphere and troposphere. Among all three, the ionosphere has the most significant impact on GNSS signal. The ionosphere is located at the upper

atmosphere where the atoms and molecules are electrically charged mainly due to the solar radiation. This layer of the air can delay the propagation of the wave causing the phase shift as well as signal attenuation. The ionospheric activities are highly correlated with the solar activities. Severe solar storm can trigger serious ionospheric fluctuation which lead to large positioning error or even signal outage. These activities are also regional distributed where low and high latitude regions are more active than the mid latitude regions. This phenomenon is known as scintillation [6]. Besides the ionosphere, troposphere also affects the GNSS signal due to the water content in the atmosphere which can distort the signal [7, 8, 9].

1.3 Solution for the GNSS-Denied Problem

In the light of the above discussion, the need to provide GNSS-denied navigation capability has become a very important topic. To deal with this issues, a myriad of alternatives and augmented systems are proposed to serve as a backup in GNSS-denied environments. However, just like GNSS has its own drawbacks, these alternative solutions all have their own disadvantages. In the following sections, two of the widely recognized alternatives associate with their limitations will be discussed.

1.3.1 Alternative RF Sources

In the absence of GNSS signal, other RF transmissions can be used as the replacement of the GNSS signals. Most of the urban areas, where the GNSS signals are stressed, are in the coverage of certain RF signals (e.g. WiFi or GSM) which makes them as ideal alternative signal sources. Several types of RF location technologies have been proposed and implemented. They are categorized into five types [3]: (1) Proximity, (2) Direction of Arrival (DOA) or Angle of Arrival (AOA), (3) Doppler, (4) Signal strength, and (5) timing or phrase. [10, 11] proposed an indoor navigation system uses RFID tags

as reference signal sources. RFID is a wireless device designed for short range, low speed data transfer. Most of the applications are related to access management or shipment tracking. Since the device provides the proximity information, it can be used for positioning especially for close proximity environments such as indoors. However, the strength is also its weakness. For the applications require wide coverage, the deployment of large number of RFID seems to be unpractical. Other signal sources such as WiFi [12, 13, 14] or GSM [15, 16] signals can provide wider coverage. But the challenge is that these RF signals also encounter the same multi-path problem similar to what GNSS has, makes the approaches difficult to provide robust and accurate position solution in cluttered environment.

1.3.2 Vision-Based Navigation or VISNAV

Vision-based Navigation (VISNAV) [17] is also an alternative solution. VISNAV relies on a camera as the primary sensor to record the pixel coordinates of objects (called feature points) whose location can be estimated in real-time [18, 19, 20] or known *a priori* [21, 22]. The pixel and feature point coordinates are subsequently used to generate a position and attitude solution of the observer [23, 24, 25]. Like the INS approach, it is also a self contained system. Two common approaches has been proposed: relative VISNAV and absolute VISNAV. In relative VISNAV, is also called visual odometry, identified and overlapping features between consecutive image frames are used to reconstruct the relative changes in position and attitude. With the drawback similar to an INS, the visual odometry has the unbounded position and attitude error grows due to the error accumulation with respect to time. In contrast, absolute mechanization requires the absolute position of the selected features resolved in a fixed navigational frame (n-frame). The position and attitude are calculated and expressed relative to the absolute n-frame with bounded error characteristic. Both relative and absolute VISNAV require

to identify features from the images. However, the feature selection algorithms are vulnerable to lighting condition changes. As a result, VISNAV is difficult to maintain the robustness and continuity.

1.3.3 Integrated Navigation Solutions

As noted in the discussion above, each types of sensor possess strengths and weakness which constraints the sensor usage to limited applications. Some of the sensors have complementary characteristics. For example, GNSS has bounded position error, but it is not self-contained and has to rely on extremely weak and vulnerable GNSS signals. On the contrary, INS error is not bounded, but it is self-contained and it has consistent performance under most of the operation conditions. For VISNAV and alternative RF sources, they can provide bounded solution in GNSS-challenged environments. Thus, the goal of multi-sensor integration system is to design a navigation system which can accommodate wide ranges of applications, ultimately cover all cases, by merging all available information from the sensors. The integrated navigation system is a mathematical algorithm which is designed to seamlessly blend sensor measurements and generates an optimal and robust state estimate. The performance depends on the selection of available sensors. For example, INS/GNSS integration can enhance the performance compared with the INS or GNSS alone. However, this enhancement is only available for the cases with good GNSS signal reception such as outdoor with open sky. Places such as indoor might not benefit from the integration. If sensors with indoor capability (e.g. VISNAV) can be easily added to the system, the overall performance is then truly enhanced. Thus, part of the design of integrated navigation systems is designing flexible integration algorithms which can be easily tailored to accommodate different available sensors.

1.4 Problem Statement

An important goal in designing any integrated, multi-sensor navigation system is to develop sensor fusion algorithms which generate an optimal solution. The term "optimal solution" is normally used to mean the most accurate solution or one with the least error. When dealing with integrated navigation systems designed around low cost inertial sensors, however, a sub-optimal navigation solution may be an acceptable compromise for an estimate with increased stability and robustness.

As is done in most INS/GNSS integration architectures, the Extended Kalman Filter (EKF) and its close variant the Iterated Extended Kalman Filter (IEKF) are used as the INS/GNSS/VISNAV or INS/VISNAV integration filters [21, 22]. The integration filter fuses the information from the IMU, camera, and/or GNSS receiver to generate an optimal estimate of the navigation state vector (position, velocity and attitude) and corrections for the IMU output errors. However, the challenges with integrating VISNAV into the system is the stability of VISNAV solution due to its nonlinear nature. To see why this is the case, consider how INS is integrated with VISNAV in optimal approach. The INS is used as the inner loop sensor which provides a navigation solution at a high rate. Periodically, when VISNAV solutions become available they are used to reset the INS solution. Since the mathematical equations used for integrating VISNAV solutions with INS solutions are non-linear (i.e., a Kalman filter measurement update equation), they are linearized about the best current estimate of the navigation state, which is the INS solution. If the INS solution has become corrupted by large drift errors, linearizing about this INS estimate may lead to filter instabilities because the assumption of small errors (i.e., ignoring higher-order terms in the linearization) is violated. With the INS/GNSS integration this is not a problem as the non-linearities involved are mild; the large distances between a terrestrial user and the GNSS satellites minimize the effects of linearization errors. With VISNAV, however, this is not the

case. Depending on the geometry of the visual landmarks being used at some given epoch, the non-linearities can be severe and very sensitive to small linearization errors. Therefore, if the INS/GNSS/VISNAV navigation system is operated in the GNSS-denied environments where the system solely relies on INS and VISNAV, there is a potential that the solution generated might be unstable. To achieve seamless INS/GNSS/VISNAV integration, robustness is as important as the accuracy.

The purpose of this thesis is focusing on the integration of INS/GNSS/VISNAV in the GNSS-denied environments. We propose a sub-optimal absolute INS/VISNAV integration algorithm and compare it against the traditional and optimal approach as mentioned above. Analog to the INS/GNSS integration, depending on the level of information being used for integration, these two algorithms are called loose and tight integrations, respectively. In the loose integration approach, information from the INS and VISNAV generate navigation solutions independently. The INS provides position, velocity and attitude solutions whereas the VISNAV generates position and attitude. Then, a EKF based fusion filter combines the two solutions (at position and attitude level) to generate an estimate of the navigation state vector and IMU error states. Since INS and VISNAV calculate navigation solution independently, INS error has no effect on VISNAV. As a result, the integrated solution is immune to large INS error, hence increases the robustness. For optimal tight integration, the INS position and velocity solution are used to linearize the non-linear equations relating feature point pixel coordinates to navigation state vector. The linearized equations are then used in an EKF or IEKF measurement update equation to calculate navigation state vector and IMU error corrections. These corrections are added to the INS solution to generate a final and optimal estimate of the navigation state vector. As is done in most INS/GNSS, both loose and tight architectures are implemented under the EKF (or IEKF) framework. Therefore, the integration of INS/GNSS/VISNAV is then straightforward.

1.5 Prior Work in INS/VISNAV Navigation System

The idea of integrating INS and VISNAV is also not new and there is an extensive body of work in this area. The approaches can be categorized into two general categories: The relative VISNAV and the absolute VISNAV. In relative VISNAV, identified features across consecutive images frame are used to determine the relative changes in position and attitude from frame to frame, these relative changes will be integrated with the solution from INS for generating the final solution. This approach is also known as visual odometry [26, 27]. Various algorithms has been proposed for this approach [28, 29]. Some of the algorithms are deployed for UAV applications [19, 20] others might be used for ground vehicle navigation [29, 30]. The visual odometry approach can slow down the error grows of the INS but it is still unable to solve the accumulated error problem.

A survey of key works in the area of integrating INS and VISNAV for absolute VISNAV formulation is given in [31]. Relative to the work reported in this thesis, the relevant integrated INS/VISNAV systems are described in [22, 31, 32, 33, 34, 35]. In [33] and [34], a tightly integrated (integration at the pixel level) INS/VISNAV system is presented. The INS solution in the [33] and [34] systems is based on a tactical grade IMU. The filter features feedback in that an estimate of inertial sensor output errors is generated and used to correct subsequent inertial sensor measurements. The architecture in [35] is also a tight INS/VINAV integration with tactical grade inertial sensors but uses a feed-forward formulation. In this case, the filter outputs are used to correct the navigation state vector (position, velocity and attitude) estimates of the INS. The filter state in [35] includes the position coordinates of the features being tracked by the VISNAV algorithm. The dynamics of the feature points error states are modeled as a simple random walk to prevent the Kalman gain associated with those states from going to zero. The approach in [31] is a tight integration between VISNAV and INS. Like [33] and [34] it includes feedback of inertial sensor errors. Similar to [35] it includes the

location of features as states in the filter and accounts for their uncertainty in estimating the position, velocity and attitude of the user. This approach is conceptually similar to visual simultaneous localization and mapping (vSLAM) [36]. A SLAM problem models the location of feature points as unknowns and includes them as states. The algorithm simultaneously solve for the position as well as the location of the features.

1.6 Thesis Contribution

This thesis makes the following contributions to the algorithm development of the absolute INS/GNSS/VISNAV integration system for GNSS-denied applications:

- It formulates the tightly coupled INS/VISNAV integration and shows that this approach has stability issue while integrating with low cost INS. The explantation is visualized by showing the existence of multiple local solutions. Large INS error might lead to the erroneous local solution. Also, it shows that the location and distribution of those local solutions are related to the geometry of feature points and camera.

- It proposes an alternative sub-optimal two stages approach (loose architecture) for integrating the INS with VISNAV which can improve the robustness of INS/VISNAV system. The formulation is derived and its performance is evaluated against the tightly coupled architecture through simulation and experimental data.

- The proposed sub-optimal architecture and the traditional optimal method have complementary strengths and weaknesses. A framework of integrating both architectures is proposed. This framework is designed to benefit from the advantages of both architectures while mitigating their drawbacks. This thesis also validate the proposed method by compare the result against the stand-alone architectures through simulation and experimental studies.

1.7 Thesis Organization

The remainder of the thesis is organized as follows: Chapter 2 will describe the key sensors and architecture used for their fusion. The chapter will start with a description of inertial sensors and sensor error models. Then a brief description of cameras and the mathematical model for camera errors will be presented. The feature point selection algorithm is also briefly introduced in this chapter. The derivation of tight integration is presented in chapter 3 It will followed by the simulation results and the multiple minimum problem that tight integration encountered. The multiple minimum problem motivates the development of the loose integration. The loose integration will be presented in Chapter 4. An algorithm for solving position and attitude using camera measurement is introduced. This algorithm is used to provide position and attitude estimates solely from the VISNAV. Then the EKF architecture used for integrating the INS with VISNAV in loosely coupled scheme is described. This chapter will close by presenting simulation and experimental results showing the performance of the loose integration scheme. Chapter 5 will propose a fusion scheme which leverages the strengths of loose and tight integration architectures. This approach is called Multiple Hypothesis Filter and can be easily extensible to integration strategies for multi-sensor integration. The chapter starts by describing and deriving the basics of the multiple hypothesis filter. Then an adaptation of this concept, called dual hypothesis filtering, to the INS/VISNAV integration scheme is presented. The chapter closes with simulation studies and experimental results comparing loose, tight and dual hypothesis approaches. Chapter 6 will summarize the research carried out in this thesis and suggest the direction of future research.

Chapter 2

Sensors and System Architecture

In this chapter the architecture of navigation systems which fuse inertial navigators with cameras is described. The sensors that make up the system and their error characteristics are described in some detail. Knowledge of the various sensors' strengths and weakness is key to understanding the approaches. The algorithms discussed in subsequent chapters.

2.1 Inertial Navigation Systems

An Inertial Navigation System (INS) is a navigation method whereby the position and orientation of a platform is determined from a history of the platform's kinematic. This is accomplished by measuring the acceleration and angular rate, respectively. The calculation of position and orientation relies solely on the relationships between acceleration, velocity, position, and angular rate and orientation. Therefore, the solution can be generated continuously without any information from external sources. Inertial navigation is a subset of a class of navigation methods known as dead reckoning.

Inertial navigation relies on the measurements from two types of sensor: Accelerometers (to measure specific force or acceleration) and rate gyros (to measure angular velocity). A three dimensional inertial navigation system normally requires three orthogonal triads of accelerometers and gyros to measure the six degrees of freedom motion. The term inertial measurement unit (IMU) is used to refer to the collection of an accelerometer and gyro triads. Even though this term is sometimes used interchangeably with the Inertial Navigation System (INS). In this thesis, the term INS is used to refer a combination of an IMU and a computer that can execute the navigation algorithm for providing the position, velocity and orientation solutions.

Low cost IMUs as also called consumer/automotive grade IMUs (in reference to their application) have been readily available for some time. They are the enablers of many novel guidance, navigation and control (GNC) applications. Prime examples of this are Unmanned Aerial Vehicles (UAV) and autonomous robot operations. An INS which uses IMU as the primary sensor is advantageous of being self contained (i.e., do not require information obtained from sources external to the user), and, thus, its operation is immune to external interference in any terrestrial or water environment. Furthermore, the computational burden of the INS is small, so a navigation solution can be generated at a high rate ($\geq$ 50 Hz). The impact of the wide availability of these low cost sensors, however, is tempered by the fact that their outputs are corrupted by large noise and time varying biases. This is particularly problematic when they are used to generate an estimate of a vehicles navigation state vector (position, velocity and attitude) due to the cumulative nature of the system errors. Even the smallest measurement error will lead to growing solution error with respect to time. As a result, the error of the navigation state estimates solely derived from these IMU measurements rapidly becomes unbounded which makes them potentially unusable.

2.1.1 Accelerometers

The term accelerometer is not an accurate terminology of what we expect the sensor to measure. These sensors sense the acceleration as the combination of applied mechanical force plus the gravitational effect ($\mathbf{g}$) acting on a test mass. For a leveled non-moving sensor, it produces a reading of $g = 9.81\ m/s^2$ (value varies with location) in the upward direction which is due to the gravitational acceleration ($\mathbf{g}$). In fact, the specific force is what we need for motion construction and should be extracted from the accelerometer readings. The extraction of specific force requires knowledge of the attitude of the accelerometers so that the projection of $\mathbf{g}$ vector onto the three directions of a triad can be correctly removed from the accelerometers measurements.

Accelerometers can be visualized as being mechanized using a mass-spring system. When the sensors experience acceleration, a small proof mass inside the sensor is displaced causing the deflection of a spring connected to the mass block. By measuring the spring deflection, the acceleration can be derived given a proof mass. In practice the proof mass and spring are combined and made from: piezoelectric, piezoresistive or capacitive devices. Their purpose is to convert the mechanical deflection into electrical signal. Most of the modern accelerometers are fabricated using Micro Electro-Mechanical Systems (MEMS) process. As a result, the size and price are greatly reduced.

2.1.2 Rate Gyros

Rate gyro is a sensing device which can provide the measurement rotation rate. The development of this device can be traced back to the 19th century. When it was realized that it had potential of being used for military applications such as heading reference or vehicle stabilization. Traditional gyro consists of a massive, well manufactured spinning wheel mounted on a three-axes platform. This fast spinning wheel provides a stable axis which is not affected by the platform rotation. The angular information can be

measured by the sensors installed on three freely rotating joints. Modern gyro designs use a different physical principle for sensing the angular information. For example, the Ring Laser Gyro (RLG) or Fiber Optics Gyro (FOG) operate on the principle of the Sagnac interference. A light beam is split into two beams and passing through a same closed path but in the opposite directions. These two beams exit the path from the point of entry. If there is no angular rotation, the phase of these two beams are synchronized. The relative phase shift of the two beams depends on the angular rate of this enclosed path. Another poplar way to mechanize gyros is using MEMS technology. MEMS gyros have been widely deployed in many applications due to their low cost. These gyros have a vibrating element for sensing the rotation. By measuring the capacitance change between two electrodes due to Coriolis motion, the angular rate can be calculated.

2.1.3 INS Mechanization Equations

The IMU sensors provide the measurement of the motion. To construct the history of the motion, it requires a mathematical model which can take in the measurement and build the trajectory by using the equations of motion where the model is parameterized in a local navigation frame, the equations have the following form:

$$\dot{\mathbf{p}} = \mathbf{T}\mathbf{v}^n \tag{2.1a}$$

$$\dot{\mathbf{v}} = \mathbf{C}_b^n \mathbf{f}^b - \{[2\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n]^\times\}\mathbf{v}^n + \mathbf{g}^n \tag{2.1b}$$

$$\dot{\mathbf{C}}_b^n = \mathbf{C}_b^n([\boldsymbol{\omega}_{ib}^b]^\times - [\boldsymbol{\omega}_{in}^b]^\times) \tag{2.1c}$$

The vector $\mathbf{p}$ and $\mathbf{v}$ represent the position and velocity, respectively. $\mathbf{C}_b^n$ is the rotational matrix from body frame (b-frame) to navigation frame (n-frame). $\boldsymbol{\omega}_{ab}^c$ is defined as the rotation rate of a frame relative to b frame expressed in c frame. The subscript i, e, b and n represent inertial, earth, body and navigational frame, respectively. The

accelerometer provides the specific force $\mathbf{f}$ and rate gyro produces body frame to inertial frame ($\boldsymbol{\omega}_{ib}^{b}$) rotation rate. The notation $[\bullet]^{\times}$ used in the equation is the skew-symmetric matrix representation of a vector. $\mathbf{T}$ is a conversion matrix that can convert the linear velocity to the angular change in latitude(L) and longitude (λ):

$$\begin{bmatrix} \frac{1}{R_N+h} & 0 & 0 \\ 0 & \frac{1}{(R_E+h)\cos L} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.2}$$

where R_N and R_E are radius of curvature of the reference ellipsoid along the north/source and east/west direction. h is the altitude above the reference ellipsoid. Equation 2.1 states the change of position, velocity and attitude with respect to time. While the rotational matrix $\mathbf{C}_{n}^{b}$ can be used to parameterize the attitude. in this thesis, we select Euler angles with 3-2-1 rotation sequence of yaw (ψ), pitch (θ) and roll (ϕ) for attitude representation. The three Euler angles can be written in vector form:

$$\boldsymbol{\alpha} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{2.3}$$

The Euler angles represent the rotation from a navigational frame to a body frame. Note that the Euler angle representing in vector form is for convenience. It does not have the additive and multiplicative properties of a mathematically proper vector. The attitude differential equation parameterized by rotation matrix as shown in Equation 2.1c can be replaced by the Euler angles as follows:

$$\dot{\boldsymbol{\alpha}} = \mathbf{F}(\boldsymbol{\alpha})\boldsymbol{\omega}_{nb}^{b} \tag{2.4}$$

where

$$\mathbf{F}(\boldsymbol{\alpha}) = \frac{1}{\cos\theta} \begin{bmatrix} 1 & \sin\phi\sin\theta & \cos\phi\sin\theta \\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.5}$$

where $\boldsymbol{\omega}_{nb}^{b}$ is the angular rate of the body frame to the navigation frame resolved in the body frame. The angular rate $\boldsymbol{\omega}_{nb}^{b}$ is obtained from the gyro measurement $\boldsymbol{\omega}_{ib}^{b}$, the angular velocity of the body frame to the inertial frame minus the rotation rotation rate of the navigation frame relative to the inertial frame.

$$\boldsymbol{\omega}_{nb}^{b} = \boldsymbol{\omega}_{ib}^{b} - \boldsymbol{\omega}_{ib}^{b} \tag{2.6}$$

The numerical calculation of Equation 2.1 and Equation 2.4 can be conducted using simple Euler integration or more sophisticate scheme such as high order Runge-Kutta integration depending on the accuracy requirements and the performance of the IMU.

The differences of position, velocity and attitude are more useful than the quantities themselves from the algorithm calculation perspective. In the actual implementation, the position, velocity and attitude are updated based on applying the difference quantities to the initial values. The difference model of the equation of motion can be derived by applying first order linear approximation to Equation 2.1a, 2.1b and 2.4 to get:

$$\delta\dot{\mathbf{p}} = \mathbf{T}'\delta\mathbf{p}^{n} + \mathbf{T}\delta\mathbf{v}^{n} \tag{2.7a}$$

$$\delta\dot{\mathbf{v}}^{n} = [\mathbf{C}_{b}^{n}\mathbf{f}^{b}]^{\times}\delta\boldsymbol{\alpha} + \mathbf{C}_{b}^{n}\delta\mathbf{f}^{b} - \{[2\boldsymbol{\omega}_{ie}^{n} + \boldsymbol{\omega}_{en}^{n}]^{\times}\}\delta\mathbf{v}^{n} - \{[2\delta\boldsymbol{\omega}_{ie}^{n} + \delta\boldsymbol{\omega}_{en}^{n}]^{\times}\mathbf{v}^{n} + \delta\mathbf{g}\} \tag{2.7b}$$

$$\delta\boldsymbol{\alpha} = -[\boldsymbol{\omega}_{i}^{n}n]^{\times}\delta\boldsymbol{\alpha} + \delta\boldsymbol{\omega}_{in}^{n} - \mathbf{C}_{b}^{n}\delta\boldsymbol{\omega}_{ib}^{b} \tag{2.7c}$$

The error models above are usually used as time update equation when INS is integrated with other sensors in Kalman Filter.

2.1.4 Inertial Sensor Output Errors

The inertial sensor errors can be categorized into two different categories: The deterministic errors and the stochastic errors. Sensor misalignment, scaling factor error and the bias are considered as deterministic errors. Depends on the nature of the errors,

some are correctable through careful calibration but not all of them are straightforward to be compensated. For the errors which are generally time invariant and has fixed contribution to the output such as sensor misalignment, or the errors are correlated to the environmental factors but can be characterized or modeled (e.g. scaling factor error), calibration process can remove some of the errors from the output depends on the accuracy of the error characterization. However, to *completely* remove the errors from the output is almost impossible.

Time varying errors such as bias is another type of deterministic error. It comes from two sources: Turn-on bias and in-run bias. The turn-on bias results in a contribution to the sensor bias that can change each time when the sensor is used but remains constant during each run. This error can be calibrated but has to be done during each power on. The in-run bias is the challenge to be compensated since it can slowly changing during the span of each run. Furthermore, the mathematical model to describe this behavior is difficult to be well-established and it can vary from sensor to sensor. A common practice is to model the in-run bias as a first order Gauss Markov process as shown in equation 2.8.

The stochastic error is usually high frequency and can come from various sources. For example, the thermal noise inside an electric circuit is a major contributor to the stochastic error. The mechanical vibration from a spinning wheel or vibrating structure gyro can also contribute to the stochastic error. The stochastic noise is normally modeled as a random variable with known statistical characteristic. Zero mean Gaussian white noise is a common assumption.

$$\mathbf{n}_{(\bullet)} = \mathbf{b}_{0,(\bullet)}(t) + \mathbf{b}_{1,(\bullet)}(t) + \mathbf{w}_{(\bullet)}(t) \tag{2.8a}$$

$$\dot{\mathbf{b}}_{1,(\bullet)}(t) = -\frac{1}{\tau_{(\bullet)}}\mathbf{b}_{(\bullet)}(t) + \boldsymbol{\mu}_{(\bullet)}(t) \tag{2.8b}$$

The subscript $(\bullet)$ is a place holder to show that the error model can be applied to both rate gyro outputs (g) and accelerometer (a) triads. $\mathbf{b}_{1,(\bullet)}$ represents the turn-on bias and $\mathbf{b}_{1,(\bullet)}$ is the time-varying in-run bias. The stochastic noise $\mathbf{w}(t)$ and $\boldsymbol{\mu}(t)$ are assumed to be random variables with zero-mean and covariance

$$\mathrm{E}\{\mathbf{w}(t)\mathbf{w}(t+\Delta t)\} = \boldsymbol{\sigma}_{\mathbf{w}}^2 \delta(\Delta t) \tag{2.9a}$$

$$\mathrm{E}\{\boldsymbol{\mu}(t)\boldsymbol{\mu}(t+\Delta t)\} = \boldsymbol{\sigma}_{\boldsymbol{\mu}}^2 \delta(\Delta t) \tag{2.9b}$$

where $\delta(\Delta t)$ is the Dirac-Delta function. $\boldsymbol{\sigma}_{\mathbf{w}}^2$ and $\boldsymbol{\sigma}_{\boldsymbol{\mu}}^2$ are the standard deviation of the white noises, and their values depend on the sensor's quality.

2.2 Vision-Based Navigation

The concept of VISNAV is using camera as a sensor to determine the navigation state of a platform. Without a loss of generality let us assume a UAV is the platform equipped with an IMU and a calibrated camera mounted to the bottom of the vehicle as shown in Figure 2.1. Note that, with appropriate modifications, the results that follow can be easily extended to apply to other platforms such as robots or ground vehicles. The vehicle's navigation state vector of interest, denoted by $\mathbf{x}$, consists of position, velocity, attitude as well as IMU sensor bias states:

$$\mathbf{x} = \begin{bmatrix} (\mathbf{p^n})^{\mathrm{T}} & (\mathbf{v^n})^{\mathrm{T}} & (\boldsymbol{\alpha}^{\mathbf{n}})^{\mathrm{T}} & (\mathbf{b_a})^{\mathrm{T}} & (\mathbf{b_a})^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{2.10}$$

The first entry in the state vector is the vehicle's position, $\mathbf{p}^n$, expressed in an inertial frame or a navigation frame (n-frame). The second entry $\mathbf{v}^n$ is the vehicle's velocity vector which is also expressed in the n-frame. The third entry, $\boldsymbol{\alpha}^n$, is the camera's orientation. It is the rotation required to align the n-frame with the body frame (b-frame). A 3-2-1 Euler angle sequence of roll (ϕ), pitch (θ) and yaw (ψ) will be used to

parameterize the rotation required to align the n-frame to the b-frame. The fourth and fifth entries of $\mathbf{x}$ are IMU accelerometer and gyro output biases, respectively.

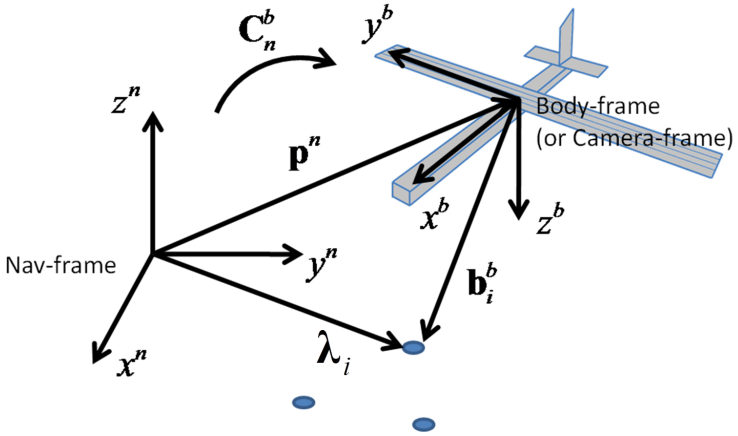

Figure 2.1: Vision-Based Navigation Problem

Each image captured by a digital camera used for navigation consists of large number of pixels whose location in the plane of the image is identified by a pair of cartesian coordinates. Objects of interest in the image are called feature points and are identified by the group of pixels that comprise them. Give two images of the same scene, feature point detection algorithm can be used to determine a one-to-one correspondence of feature points in the two images. More specifically, algorithms such as Scale Invariant Feature Transform (SIFT) [37, 38], Speeded Up Robust Feature (SURF) [39], or Harris corner detection [40] are used to extract features from the images captured by the camera and match them to a reference image. In absolute VISNAV, the navigation or n-frame position coordinates ($\boldsymbol{\lambda}_i$)of the features tracked are known *a priori*. This could be done by using an image database containing a number of images tagged with absolute position information. When a feature is detected, the matched features' pixel coordinates (u_i and v_i) and their associated absolute position ($\boldsymbol{\lambda}_i$) can be either used directly for tight integration as described in chapter 3. Alternatively, this can be used

as the input to some navigation algorithms (i.e. PnP algorithm) which is used to compute the vehicle's position, position covariance, attitude and attitude covariance. The position and attitude solution along with their responsive covariances can then be used for loose integration (chapter 4).

2.2.1 Pinhole Camera Model

The pinhole camera model is used to describe the projection of a three dimensional location onto a two dimensional image plane. In the absolute VISNAV, we assume the absolute position coordinates of the feature points ($\boldsymbol{\lambda}_i^n$) are known *a priori* resolved in n-frame. As shown in Figure 2.1, the line of sight vector from the vehicle body to a feature point can be expressed in terms of the position and orientation of the vehicle and the known position of the feature point. Mathematically, this can be expressed as follows:

$$\mathbf{b}_i^b = \mathbf{C}_n^b(\boldsymbol{\lambda}_i^n - \mathbf{p}^n) \tag{2.11}$$

where $\mathbf{b}_i^b = [b_{x,i}^b \; b_{y,i}^b \; b_{z,i}^b]^{\mathrm{T}}$ is the line of sight vector from the vehicle to the i^{th} feature point resolved in a coordination frame attached to the vehicle called body frame (b-frame). An additional frame is defined and attached to the camera known as camera frame (c-frame). The camera installation on the vehicle is usually calibrated, so the rotation matrix $\mathbf{C}_b^c$ is pre-determined. The line of sight vector expressed in b-frame can be expressed in c-frame by the conversion:

$$\mathbf{b}_i^c = \mathbf{C}_b^c \cdot \mathbf{b}_i^b \tag{2.12}$$

The subscript is the indication of i^{th} feature point. The superscript n, b, and c indicates a vector is resolved in n-frame, b-frame, and c-frame, respectively.

The mathematical model relating the camera's pixel measurements to the states of interest (i.e. position and attitude) is the pinhole camera projection model. It is also known as the perspective camera model [41]. The model is illustrated in Figure 2.13 and can be represented by the following equations:

$$u_i = f_x \frac{b^c_{x,i}}{b^c_{z,i}} + c_x \tag{2.13a}$$

$$v_i = f_y \frac{b^c_{y,i}}{b^c_{z,i}} + c_y \tag{2.13b}$$

Refer to Figure 2.2, u_i and v_i are the pixel coordinates on the image plane. f_x, f_y are the camera focal length in the x and y direction of the c-frame; (c_x, c_y) is the principle point of the camera. The principle point is defined as the point where the camera's optical axis intersects with the image plane. In perfect case, the optical axis is parallel to the z-axis of the camera frame.

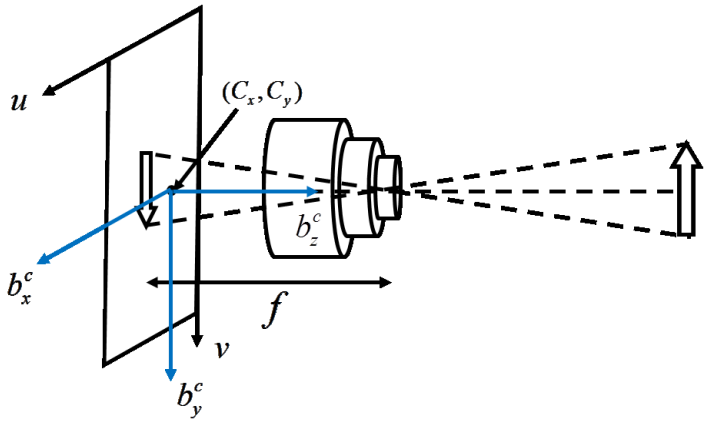

Figure 2.2: Pinhole Camera Model

With sufficient number of feature points and their corresponding pixel measurements, equations 2.11, 2.12 and 2.13 can be solved for the vehicle's position and orientation. The equations that have to be solved, however, are non-linear.

The pinhole camera model as shown in Equations 2.11, 2.12 and 2.13 can also be re-written using the homogeneous coordinate representation [41, 42]. To see how this is done, denote the homogeneous position coordinates of the i^{th} landmark as $\boldsymbol{\lambda}_i^n = \begin{bmatrix} \lambda_{x,i}^n & \lambda_{y,i}^n & \lambda_{z,i}^n & 1 \end{bmatrix}^T$ where $\lambda_{x,i}^n$, $\lambda_{y,i}^n$ and $\lambda_{z,i}^n$ are its cartesian position coordinates. The superscript n indicates that this landmark is expressed in n-frame. This landmark appears as a pixel at coordinates u_i and v_i on the camera's imaging plane. The homogeneous vector representation of the pixel coordinates in a frame attached to the camera imaging plane will be written as $\mathbf{e}_i^c = \begin{bmatrix} s_i u_i & s_i v_i & s_i \end{bmatrix}^T$ where s_i is an arbitrary scaling constant [41, 42]. The superscript c is used to indicate the c-frame. For clarity and without a loss of generality, in the derivation that follows, it is assumed that the body frame is the same as the camera frame and, thus, $\mathbf{e}_i^b = \mathbf{e}_i^c$ where the superscript b indicates the body frame, and $\mathbf{C}_b^c$ is equal to identity. If the body frame is not coincident with the camera frame but the transformation matrix between the body and the camera frame $\mathbf{C}_b^c$ is known (which is the case in many applications), only a trivial modification to the algorithm presented is required.

The projective transform that relates $\boldsymbol{\lambda}_i^n$ and $\mathbf{e}_i^c$ involves the the navigation states of interest ($\mathbf{C}_n^b$ and $\mathbf{p}^n$) and is given by:

$$\mathbf{e}_i = \begin{bmatrix} \mathbf{M}\mathbf{C}_n^b & | & -\mathbf{M}\mathbf{C}_n^b\mathbf{p}^n \end{bmatrix} \boldsymbol{\lambda}_i \tag{2.14}$$

$$= \mathbf{H}\boldsymbol{\lambda}_i \tag{2.15}$$

$$= \left[\begin{array}{c|c} \mathbf{H}_1 & \mathbf{H}_2 \end{array}\right] \boldsymbol{\lambda}_i \tag{2.16}$$

where $\mathbf{H}$ is a partitioned matrix with $\mathbf{H}_1 = \mathbf{M}\mathbf{C}_b^c\mathbf{C}_n^b$ and $\mathbf{H}_2 = -\mathbf{M}\mathbf{C}_b^c\mathbf{C}_n^b\mathbf{p}^n$. The matrix $\mathbf{M}$ is called the camera intrinsics matrix. Its elements are the camera intrinsics which govern the way that Euclidean points are projected onto the camera's imaging

plane.

$$\mathbf{M} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.17}$$

The elements in the camera intrinsics matrix are: f_x, f_y, the focal length along x and y axes in the c-frame. $(p_x,\ p_y)$, the principal point which is defined as the intersection of the optical axis with the image plane. The camera is said to be calibrated if $\mathbf{M}$ is known. The advantage of using homogeneous coordinate is that the pinhole camera model can be represented in the matrix form and the matrix operation can also be applied.

2.2.2 Camera and VISNAV Errors

VISNAV position and attitude errors arise from several sources. Misalignment in between the axes of the camera and the axes of the INS, errors in the camera calibration parameters, errors in the position coordinates of the i^{th} landmark and errors in registering its pixel location on the imaging plane (i.e., errors in u_i and v_i) will all contribute to errors in the final position and attitude. In the derivation that follows, the camera alignment and the camera calibration are assumed to be perfect. This is not unreasonable, since these error sources may be minimized through careful calibration. That leaves the *landmark survey* error and the *pixel registration* error, which will be described here.

The pixel registration error covariance matrix captures the magnitude and nature of the pixel registration error. The pixel registration errors are defined as the difference between the actual pixel coordinates of a feature or landmark on the image plane and the pixel location reported by the camera and feature detection algorithm. Mathematically, this can be written as:

$$\begin{bmatrix} \delta u_i \\ \delta v_i \end{bmatrix} = \begin{bmatrix} \hat{u}_i - u_i \\ \hat{v}_i - v_i \end{bmatrix} \tag{2.18}$$

Note that $\hat{u}_i$ and $\hat{v}_i$ are measured quantities and their error-free counterparts are u_i and v_i, respectively. The mean of the measured pixel coordinates are u_i and v_i, and the pixel registration errors are modelled as independent, zero-mean random variables that are uncorrelated with respect to time. Now we can define a covariance matrix for the errors associated with the registration of the i^{th} landmark as:

$$\mathcal{E}\left\{\begin{bmatrix} \delta u_i \\ \delta v_i \end{bmatrix}\begin{bmatrix} \delta u_i & \delta v_i \end{bmatrix}\right\} = \begin{bmatrix} \mathcal{E}\left\{(\hat{u}_i - u_i)^2\right\} & \mathcal{E}\left\{(\hat{u}_i - u_i)(\hat{v}_i - v_i)\right\} \\ \mathcal{E}\left\{(\hat{u}_i - u_i)(\hat{v}_i - v_i)\right\} & \mathcal{E}\left\{(\hat{v}_i - v_i)^2\right\} \end{bmatrix} \tag{2.19}$$

$$= \begin{bmatrix} \sigma_{v_i}^2 & \sigma_{u_i v_i} \\ \sigma_{u_i v_i} & \sigma_{u_i}^2 \end{bmatrix} \tag{2.20}$$

where $\mathcal{E}$ is the expectation operator. The diagonal entries are the variances of the pixel registration errors. The off-diagonal terms are a measure of the correlation between the errors in u_i and v_i. Another error source is the landmark registration error. This error is caused by the inaccuracy during the pre-surveying of the landmarks' position. It can also be modeled as independent, zero-mean random variables but for now we assume the error is relatively small compared to the pixel registration error. Therefore, the landmark registration error can be neglected or be absorbed by the pixel error.

2.2.3 Scale Invariant Feature Transform (SIFT) Algorithm

For an object recorded by an image, some selected feature points on the object can provide descriptions to represent the object. If the descriptions are invariant to the size change of the object in the image, the selected feature points are defined to be scale invariant. Scale-invariant feature transform (or SIFT) [37, 38] is a computer vision algorithm to identify scale invariant feature points and generate feature descriptions in images. It achieves the scale invariant by convoluting the input image with 2-D Gaussian kernel at different scales. Then, locally extremum points are selected as feature points.

The feature descriptions are calculated and assigned to each selected points. Speeded Up Robust Features (SURF) [**?**] is another feature scale-invariant selection algorithm with faster execution. The selection of SIFT feature points can be divided into four stages which are briefly introduced in the following:

Feature Points Localization

The first stage of the algorithm is to selected robust and repeatable feature points. These selected feature points should be scale invariant. Koenderink [43] and Lindeberg [44] suggested the Gaussian kernel function is an ideal function for extracting the scale information from an image. The scale space image $L(u, v, \sigma)$ is generated by performing the convolution of a Gaussian kernel function $G(u, v, \sigma)$ with an input image $I(u, v)$ as represented (2.21)

$$L(u, v, \sigma) = G(u, v, \sigma) * I(u, v) \tag{2.21}$$

where * is the convolution operation and Gaussian kernel function is defined

$$G(u, v, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(u^2+v^2)/2\sigma^2} \tag{2.22}$$

σ defines the scale level is being used for the operation. This operation can also be treated as the original input image filtered by a Gaussian low pass filter with bandwidth parameter σ. [38] proposed a criteria of selecting feature points by searching the extrema of the difference of two nearby scales separated by a constant factor k

$$\begin{aligned} D(u, v, \sigma) &= (G(u, v, k\sigma) - G(u, v, \sigma)) * I(u, v) \\ &= L(u, v, k\sigma) - L(u, v, \sigma) \end{aligned} \tag{2.23}$$

The locations and scales of the feature points are determined by the local maxima or minima of $D(u, v, \sigma)$. To detect the local extrema of $D(u, v, \sigma)$,each candidate points are compared against their eight neighbors in the same scale and nine neighbors in the scale above or below as shown in Fig.2.3. It is selected as feature points if it is larger (or smaller) than all 26 neighbors. The advantage of this method [38] is the replacement of the Laplacian of Gaussian (LoG) with DoG, which makes it computational efficient when determine the local extrema.

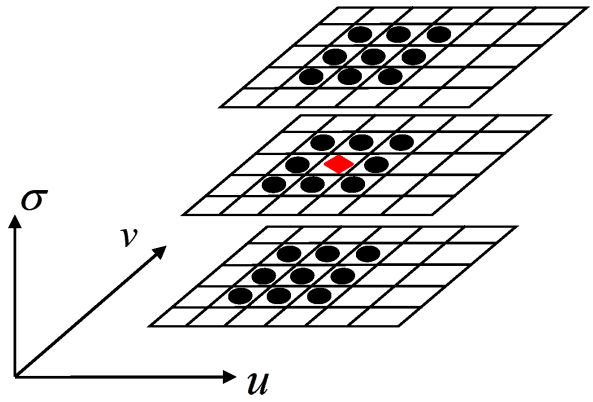

Figure 2.3: Local Extremum

Feature Points Accuracy

The image noise can corrupt the local extrema and further affecting the stability of the selected points. The stability can be improved by removing the points with low contrast. A method is proposed [45] to interpolate the feature points with sub-pixel accuracy. The approach is based on the Taylor expansion of the DoG function $D(u, v, \sigma)$ and is approximated by

$$D(\mathbf{x}) \approx D + \frac{\partial D^T}{\partial \mathbf{x}} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}}\mathbf{x} \tag{2.24}$$

where D and its derivative are evaluated at the selected extrema's locations. $\mathbf{x} = (u, v, \sigma)^T$ is the offset from the selected location. By taking the derivative of $D(\mathbf{x})$,we are able to determine the location of the extrema $\hat{\mathbf{x}}$.It is written as

$$\hat{\mathbf{x}} = -(\frac{\partial^2 D}{\partial \mathbf{x}^2})^{-1} \frac{\partial D}{\partial \mathbf{x}} \tag{2.25}$$

If we substitute 2.25 into 2.24, the scale-space function D at extrema is given

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2} \frac{\partial D^T}{\partial \mathbf{x}} \hat{\mathbf{x}} \tag{2.26}$$

this value at $\hat{\mathbf{x}}$ can be used to rejected feature points with low contrast. As proposed in [38], all extrema with $\|D(\hat{x})\| < 0.03$ are discarded. The value is assessed by the experimental results.

Orientation Assignment

The selected candidates are scale-invariant but they are sensitive to orientation changes. To achieve rotational invariant, each feature point is assigned with a canonical orientation based on the local property around neighboring pixels. The feature point descriptor will be expressed relative to this orientation. The orientation is determined by the gradient direction within a region surrounding the feature point. The gradient orientation $\theta(x, v)$ and magnitude $m(x, v)$ in the neighboring region can be computed using

$$\theta(u, v) = \tan^{-1}(L(u, v+1) - L(u, v-1))/L(u+1, v) - L(u-1, v) \tag{2.27}$$

$$m(u, v) = \sqrt{(L(u+1, v) - L(u-1, v))^2 + (L(u, v+1) - L(u, v-1))^2} \tag{2.28}$$

A orientation histogram with 36 bins covering 360 degree is formed. The calculated orientation at the selected feature point will contribute to the corresponding bin. The contribution is weighted by its magnitude and by a Gaussian-weighted function. The

highest peak among all the bins will be the canonical orientation of the feature point. If there are multiple peaks with similar magnitude in the bins, multiple feature points are generated at the same location and scale with different orientations.

Feature Points Descriptor

The previous sections discussed the method to create feature points and assign their positions, scales, and orientations. These feature points are invariant to image scaling and rotation. In this section, a partially invariant to illumination set (so called descriptor) is assigned to each feature point. The assigned sets are highly distinctive which makes them very useful for applications such as object recognition. The SIFT descriptor is a type of Distribution Based Descriptor. It is robust to small shifts in the local geometry. A window is defined surrounding the selected point. Similar to the operation for orientation assignment, the magnitude and orientation of every pixel within the box are calculated. The window is divided into sub-boxes known as descriptor. Pixels within the descriptor boxes are grouped together and their magnitudes and orientations are used to determine the canonical orientation and magnitude of the descriptor as shown in Figure 2.4.

The magnitudes and orientations are expressed relative to the canonical orientation of the selected feature point. Descriptors generated in each boxes create a multi-dimensional descriptor vector. A Gaussian weighting function is also applied to the magnitudes of every pixel within the box. This Gaussian weighting function assigns smaller weights to points that is further away from the feature points. This is the description of the selected feature points which can be used for applications such as feature matching or object recognition. Figure 2.5 shows the feature points selected by the SIFT algorithm from two images and their locations are marked by green circles. The selected feature points are matched as shown in the green lines in the figure. Note

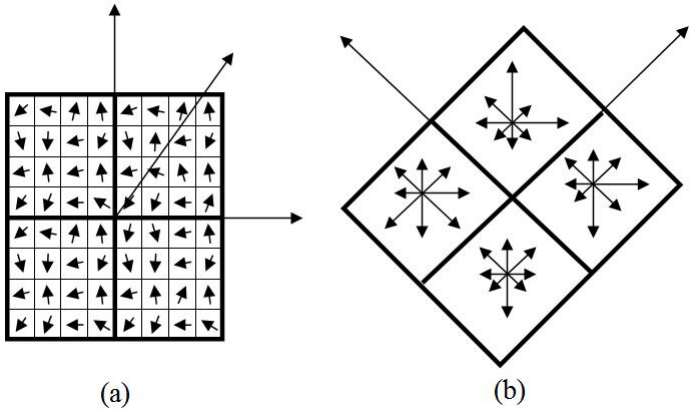

Figure 2.4: Feature point descriptor (a) orientation and (b) orientation histogram

that the SIFT algorithm can identify the correct matched pairs on the objects with different scale and orientation in the two sample images.

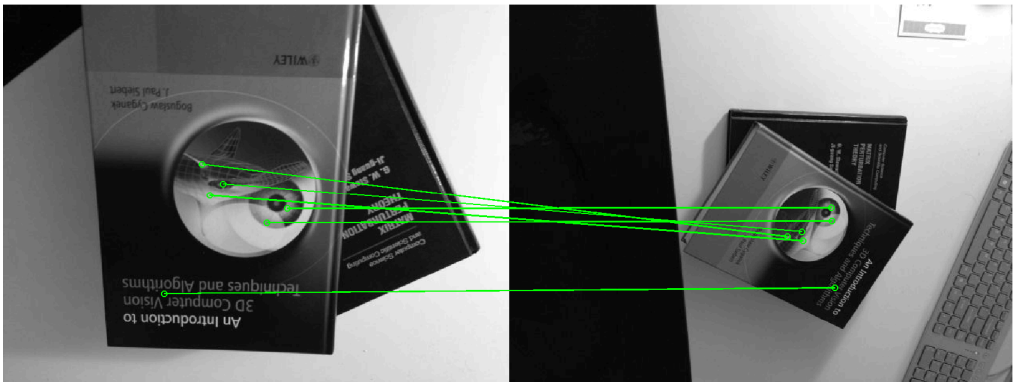

Figure 2.5: Selected Feature Points and Matched Pairs

2.2.4 Outlier Removal: RANSAC

The SIFT algorithm assumes a matched pair of feature points have "similar" feature descriptor. However, it is not necessary true if two points have similar feature descriptor are match. The algorithm identifies match pairs based on measuring the distance (or likelihood) of two descriptor vectors. Therefore, it inevitably includes those false matches and regards them as positive matches. Those false positive matches will significantly affect the accuracy of the estimates and must be removed before passing them to the estimation algorithm. A commonly used algorithm is called RANSAC.

RANSAC stands for "RANdom SAmple Consensus". It is an iterative approach to estimate the parameters of a model from a set of data corrupted by number of outliers. The outliers, contrary to inliers, are the data points whose distribution cannot be explained by a given model. This approach is first proposed by [46] and has been widely adopted in computer vision community. The variants of RANSAC such as MSAC and MLESAC are also proposed in past years [47, 48, 49]. The presence of outliers can due to various reasons such as large sensor noise, erroneous measurements, or wrong hypothesis of interpreted data. This method is considered as a non-deterministic algorithm in the sense that the data points are concluded as inliers can only with a certain probability in each iteration. As more iterations are conducted, the probability of distinguishing inliers from outliers is increased. In general, RANSAC and its variants are two-step approaches in an iterative fashion.

- *Hypothesis* - Randomly select minimum number of samples from the data points and used the select samples to calculate the model parameters. The minimum number of samples varies depends on the number of unknown parameters. It is equal to the number of unknown parameters to be determined.

- *Testing* - In testing step, the algorithm uses the rest unselected data points and

check the consistency of those points against the model calculated in the hypothesis step. If a point matches the model, mark it as an inlier and record total number of inliers in this iteration. The set of inliers is defined as consensus set.

Repeat the two step process until the probability of finding a better consensus set is below a given threshold. The theoretical aspect of the algorithm will be briefly discussed in this section.

Given a model with h unknown parameters and noise free data sets ($\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$), the model is defined as

$$M(\boldsymbol{\theta}) = \{\mathbf{y} \in \mathbb{R}^n : f_M(\mathbf{y}; \boldsymbol{\theta}) = 0 \ \ \forall \mathbf{y}\} \tag{2.29}$$

With sufficient number of data points ($N \geq h$), the model parameters can be uniquely determined by the given data points. However, in reality, the data set is usually corrupted by unknown errors which result in the inequality of equation 2.29. The inequality is measured by the distance from data $\mathbf{y}$ to the model $M(\boldsymbol{\theta})$

$$\epsilon_M(\mathbf{y}, \boldsymbol{\theta}) = \min_{\mathbf{y}^* \in M(\boldsymbol{\theta})} \text{dist}(\mathbf{y}, \mathbf{y}^*) \tag{2.30}$$

where dist$(\cdot)$ is a distance function. We can defined the consensus set as

$$S(\boldsymbol{\theta}) = \{\mathbf{y} \in \text{Y} : \epsilon_M(\mathbf{y}, \boldsymbol{\theta}) \leq \delta\} \tag{2.31}$$

where $\text{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ is the noisy data set, and δ is a pre-defined threshold. $S(\boldsymbol{\theta})$ is essentially the set of inliers. Data points excludes from this set are considered as outliers and should be removed from the data set.

RANSAC relies on the iteration to remove outliers. But how many iterations are sufficient to separate the outliers from the data set? As mentioned before, the iteration stops when the probability of finding a better consensus set is lower than a pre-determined threshold. Assuming the data in Y have the same opportunity of being

selected, the probability of obtaining a minimum number of samples which only contains inliers can be computed

$$q = \frac{\mathrm{C}_h^{N_I}}{\mathrm{C}_h^N} = \frac{N_I!(N-h)!}{N!(N_I-h)!}$$
$$= \prod_{i=0}^{h-1} \frac{N_I - i}{N - i} \tag{2.32}$$

where N_I is the number of inliers and C is the h-combination from a set of N or N_I elements. If the number of data points and inliers are much larger than the number of unknown parameters N, $N_I >> h$, the probability can be approximated by

$$q = \prod_{i=0}^{h-1} \frac{N_I - i}{N - i} \approx (\frac{N_I}{N})^h \tag{2.33}$$

q is defined as the probability of only inliers are selected from the random sampling. The probability of at least one sample is outlier is $1 - q$. If n different minimum sample sets are drawn from the entire population, the probability of all the sets are contaminated by outliers is equal to $(1-q)^n$. This expression states that the probability will decrease if we can produce sufficiently large number ($n >> 0$) of minimum sample set. In other words, the probability of at least one set contains only inliers will increase. Therefore, we can determine the value of n so that the probability is smaller than a pre-defined threshold μ (i.e. $(1-q)^n \leq \mu$), so

$$n \geq \frac{\ln \mu}{\ln(1-q)} \tag{2.34}$$

the number of iteration can be selected as the smallest integer number larger than the lower bound $n_{iter} = \mathrm{ceil}(\frac{\ln \mu}{\ln(1-q)})$, where ceil is the ceiling function.

Equation 2.33 states the probability depends on the number of inliers. Unfortunately, the number of inliers is usually unknown *a-priori*. To deal with this problem, $\hat{N}_I$ is defined and used to replace N_I. $\hat{N}_I$ represents the current largest number of inlier detected. For the case inliers are the majorities in the data set, $\hat{N}_I$ is normally

less or equal to N_I. Evidently, $q(\hat{N}_I) \leq q(N_I)$ for $\hat{N}_I$ ¡ N_I. We can further derive $(1 - q(\hat{N}_I))^n > (1 - q(N_I))^n$. Instead of using N_I which is unavailable for calculating the number of iteration n_{iter}, the alternative approach is to find a more conservative threshold

$$\hat{n}_{iter} = \text{ceil}(\frac{\ln \mu}{\ln(1 - q(\hat{N}_I)}) \tag{2.35}$$

The alternative is based on the assumption of $\hat{N}_I \leq N_I$ if the data set is not severely corrupted by large noise. For the data set with large noise, the calculated number of iteration might be too optimistic. More advance approaches [47, 48] are proposed to further increase the robustness.

RANSAC provides a robust approach for outlier removal. However, this approach still has some disadvantages. A major disadvantage is that this approach is considerably time consuming due to its iterative nature. Furthermore, the time required for each calculation is not bounded and determined *a priori* which makes it difficult for real-time applications.

Chapter 3

Tightly Coupled INS/VISNAV Integration System

The information from the INS and VISNAV needs to be fused together in order to realize the benefits from the individual systems. Two integration approache(named tightly and loosely approaches) have been proposed. The terms "tight" and "loose" are analogs to the integration architectures of INS/GNSS. The tight architecture refers to a method of using lower level of information for sensor integration compared with the loose architecture. Both architectures use an EKF and/or its variant as the sensor integration method. However, the way of formulating the EKF is different which results in different characteristics for the two approaches. The tightly integrated approach is introduced in this chapter and loosely approach will be given in the next chapter.

3.1 Integration Architecture

The tight INS/VISNAV integration refers to a mathematical algorithm that fuses information from INS and camera at the camera's pixel measurement level to generate

an optimal estimate of the navigation state vector (position, velocity and attitude) and the gyro and accelerometer biases. The block diagram of tight integration architecture is illustrated in Figure 3.1. The pixel measurements are fused with IMU measurement using an EKF or its variant the IEKF. The INS solution is used to linearize the non-linear camera measurement model. Then, the state vector estimate is solved using the feature points' pixel coordinate measurements iteratively.

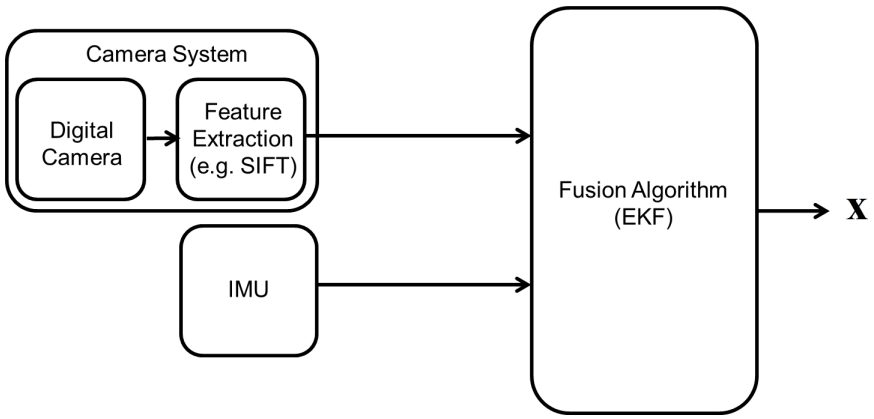

Figure 3.1: Functional block diagram of tight integration

The tight integration uses pixel coordinates for sensor fusion. The relationship between pixel coordinates and states of interest is modeled by the pinhole camera projection model. The equations relating states of interests and the pixel coordinates are nonlinear. In practice, we linearize the pinhole camera projection model and use the linearized equation as measurement update equation to integrate with the linearized time update equation we presented previously under the IEKF framework. The reason of adopting the IEKF instead of EKF is because the measurement model is linearized from a set of nonlinear equation, it often requires multiple iterations before the solution (or state) converge. Traditional EKF only updates the state one time during each measurement update. The state correction calculated and added to the *a priori* estimate may not close to the true solution. On the other hand, IEKF iteratively updates

the state until convergence is achieved or additional iteration do not improve the solution. This iterative update, therefore, can make up for some of the error caused by the linearization. The IEKF approach for tight INS/VISNAV integration will also be introduced in this chapter.

3.2 Linearization of Pinhole Projection Model

Recall that the pixel coordinates and states of interests are related by equation 2.2 and 2.11. One way to solve these two equations is by linearization. For clarity and without a loss of generality, in the derivation that follows, it is assumed that the body frame is the same as the camera frame and, thus, superscript b and c are interchangeable while $\mathbf{C}_b^c$ is equal to the identity. Given this assumption about the b and c frame, we first perturb on Equations 2.2 and 2.11 with respect to an initial guess of position and attitude to obtain the linearized pinhole camera model. This linear model can then be solved for the position and attitude perturbations using linear techniques. The resulting position and attitude perturbation can then be used to correct the initial guess of attitude and position about where the equation was linearized. In what follows, we show the details of how this is done as it is the key to understanding the stability of the tight integration approach.

Let us first define $\delta\boldsymbol{\xi}_i = [\delta u_i \quad \delta v_i]^{\mathrm{T}}$ to be the perturbation of i^{th} pixel coordinates. The perturbation of the camera position and attitude are denoted as $\delta\mathbf{p}$ and $\delta\boldsymbol{\alpha}^n = [\delta\boldsymbol{\alpha}_{b_x} \quad \delta\boldsymbol{\alpha}_{b_y} \quad \delta\boldsymbol{\alpha}_{b_z}]^{\mathrm{T}}$, respectively. Note, $\delta\boldsymbol{\alpha}^n$ is the small rotation along three body axes. It is not the perturbation of the Euler angles. The linearization is presented as a two step process. First, we relate the perturbation of pixel coordinate $\delta\boldsymbol{\xi}_i$ to the perturbation of line of sight vector $\mathbf{b}_i^b$. This relationship can be written as follows

$$\begin{bmatrix} \delta u_i \\ \delta v_i \end{bmatrix} = \begin{bmatrix} \frac{f_x}{b^b_{z,i}} & 0 & -\frac{f_x b^b_{x,i}}{(b^b_{z,i})^2} \\ 0 & \frac{f_y}{b^b_{z,i}} & -\frac{f_y b^b_{y,i}}{(b^b_{z,i})^2} \end{bmatrix} \begin{bmatrix} \delta b^b_{x,i} \\ \delta b^b_{y,i} \\ \delta b^b_{z,i} \end{bmatrix}$$
$$= \mathbf{H}_{1,i} \delta \mathbf{b}^b_i \tag{3.1}$$

Next, we relate $\delta \mathbf{b}^b_i$ to perturbation of the states of interest $\delta \mathbf{p}^n$ and $\delta \boldsymbol{\alpha}^n$. This can be written as follows

$$\begin{bmatrix} \delta b^b_{x,i} \\ \delta b^b_{y,i} \\ \delta b^b_{z,i} \end{bmatrix} = \begin{bmatrix} -\mathbf{C}^b_n & \mathbf{C}^b_n [\mathbf{l}^n_i - \mathbf{p}^n]^\times \end{bmatrix} \begin{bmatrix} \delta \mathbf{p}^n \\ \delta \boldsymbol{\alpha}^n \end{bmatrix}$$
$$= \mathbf{H}_{2,i} \begin{bmatrix} \delta \mathbf{p}^n \\ \delta \boldsymbol{\alpha}^n \end{bmatrix} \tag{3.2}$$

where the notation $[\bullet]^\times$ used to denote the skew-symmetric matrix representation of a vector. By substituting equation 3.2 into 3.1 and defining $\mathbf{H}_i = \mathbf{H}_{1,i}\mathbf{H}_{2,i}$, the linearized pinhole camera projection model is written as:

$$\delta \boldsymbol{\xi}_i = \mathbf{H}_i \begin{bmatrix} \delta \mathbf{p}^n \\ \delta \boldsymbol{\alpha}^n \end{bmatrix} \tag{3.3}$$

This equation will be used as the measurement update equation in the tight integration approach.

3.3 Measurement Update Equation

The tight integration measurement update utilizes the linearized pinhole camera model that has been derived in the previous section. It can also be written in the standard

form

$$\delta \mathbf{y}_t = \mathbf{H}_t \delta \mathbf{x} + \boldsymbol{\nu}_t \tag{3.4}$$

where the subscript t indicates tight integration. We define the state variable ($\delta \mathbf{x}$) which is the perturbation of the state vector $\mathbf{x}$ defined in Equation 2.10.

$$\delta \mathbf{x} = \begin{bmatrix} (\delta \mathbf{p}^n)^{\mathrm{T}} & (\delta \mathbf{v}^n)^{\mathrm{T}} & (\delta \boldsymbol{\alpha}^n)^{\mathrm{T}} & (\delta \mathbf{b}_a)^{\mathrm{T}} & \delta \mathbf{b}_g)^{\mathrm{T}} \end{bmatrix} \tag{3.5}$$

The dimension of the measurement vector $\delta \mathbf{y}_t$ varies depending on the number of feature points being extracted. Thus, $\delta \mathbf{y}_t$ can be written as:

$$\delta \mathbf{y}_t = \begin{bmatrix} \delta \boldsymbol{\xi}_1^{\mathrm{T}} & \delta \boldsymbol{\xi}_2^{\mathrm{T}} & \cdots & \delta \boldsymbol{\xi}_N^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{3.6}$$

where $\delta \mathbf{y}_t \in \mathbb{R}^{2N \times 1}$ and N is the total number of feature points being observed. For each of the pixel observation, the measurement matrix for i^{th} pixel measurement is given as:

$$\mathbf{H}_{t,i} = \begin{bmatrix} \frac{f_x}{b^b_{z,i}} & 0 & -\frac{f_x b^b_{x,i}}{(b^b_{z,i})^2} \\ 0 & \frac{f_y}{b^b_{z,i}} & -\frac{f_y b^b_{y,i}}{(b^b_{z,i})^2} \end{bmatrix} \begin{bmatrix} -\mathbf{C}^b_n & \mathbf{Z}_3 & \mathbf{C}^b_n [\mathbf{l}^n_i - \mathbf{p}^n]^{\times} & \mathbf{Z}_3 & \mathbf{Z}_3 \end{bmatrix} \tag{3.7}$$

The measurement matrix $\mathbf{H}_t$ can be constructed by stacking for $i = 1, 2, \ldots, N$

$$\mathbf{H}_t = \begin{bmatrix} \mathbf{H}_{t,1}^{\mathrm{T}} & \mathbf{H}_{t,2}^{\mathrm{T}} & \cdots & \mathbf{H}_{t,N}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{3.8}$$

The noise vector $\boldsymbol{\nu}$ is the pixel error of N feature points. The covariance matrix of the

noise $\boldsymbol{\nu}_t$ is defined as

$$\mathbf{R}_{\boldsymbol{\nu}_t} = \mathrm{E}\{\boldsymbol{\nu}_t \boldsymbol{\nu}_t^{\mathrm{T}}\} = \begin{bmatrix} \mathbf{R}_{\boldsymbol{\xi}_1} & \mathbf{R}_{\boldsymbol{\xi}_1\boldsymbol{\xi}_2} & \cdots & \mathbf{R}_{\boldsymbol{\xi}_1\boldsymbol{\xi}_N} \\ \mathbf{R}_{\boldsymbol{\xi}_2\boldsymbol{\xi}_1} & \mathbf{R}_{\boldsymbol{\xi}_2} & \cdots & \mathbf{R}_{\boldsymbol{\xi}_2\boldsymbol{\xi}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{\boldsymbol{\xi}_N\boldsymbol{\xi}_1} & \mathbf{R}_{\boldsymbol{\xi}_N\boldsymbol{\xi}_2} & \cdots & \mathbf{R}_{\boldsymbol{\xi}_N} \end{bmatrix} \tag{3.9}$$

where $\mathbf{R}_{\boldsymbol{\nu}_t} \in \mathbb{R}^{2N \times 2N}$. In practice, we assume pixel errors are uncorrelated across feature points (i.e., the cross correlation $\mathbf{R}_{\boldsymbol{\xi}_i\boldsymbol{\xi}_j} = 0$ for $i, j = 1, 2, \ldots N, i \neq j$).

3.4 Time Update Equation

A fully linearized time update equation can be found in section 2.1.3 or [4]. However, when consumer grade inertial sensors are used, the sensor noise is sufficiently large to be the primary contributor to navigation errors masking the contribution of many other terms in the equation. Those terms can be dropped without losing the accuracy. Furthermore, for the short range applications, the flat earth assumption provides fairly accurate approximation. The latitude, longitude and altitude representation of position can be simplified by a local x, y, z coordinate frame. The linearized equation has the following form:

$$\delta\dot{\mathbf{x}} = \mathbf{F}\delta\mathbf{x} + \mathbf{G}\mathbf{n} \tag{3.10}$$

where the state transition matrix $\mathbf{F}$

$$\mathbf{F} = \begin{bmatrix} \mathbf{Z}_3 & \mathbf{I}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 \\ \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{F}_{23} & \mathbf{F}_{24} & \mathbf{Z}_3 \\ \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{F}_{35} \\ \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & -\mathbf{I}_3/\tau_a & \mathbf{Z}_3 \\ \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & -\mathbf{I}_3/\tau_g \end{bmatrix} \tag{3.11}$$

the matrix $\mathbf{G}$

$$\mathbf{G} = \begin{bmatrix} \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 \\ \mathbf{C}_n^b & \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 \\ \mathbf{Z}_3 & -\mathbf{C}_n^b & \mathbf{Z}_3 & \mathbf{Z}_3 \\ \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{I}_3 & \mathbf{Z}_3 \\ \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{I}_3 \end{bmatrix} \tag{3.12}$$

where the $\mathbf{Z}_3$ and $\mathbf{I}_3$ refer to 3×3 zero and identity matrices, respectively. $\mathbf{F}_{23} = -[\mathbf{C}_b^n \mathbf{f}_b]^\times$ and $\mathbf{F}_{24} = \mathbf{F}_{35} = \mathbf{C}_b^n$. The $[\bullet]^\times$ is the skew symmetric matrix representation of a vector. $\mathbf{f}_b$ is the specific force measurement from accelerometer. τ_a and τ_g are the correlation time of accelerometer and gyroscope output errors, respectively. The process noise vector $\mathbf{n}$ is

$$\mathbf{n} = \begin{bmatrix} (\mathbf{w}_a)^{\mathrm{T}} & (\mathbf{w}_g)^{\mathrm{T}} & (\boldsymbol{\mu}_a)^{\mathrm{T}} & (\boldsymbol{\mu}_g)^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{3.13}$$

where $\mathbf{w}$ represents the white noise on the IMU output. The variable $\boldsymbol{\mu}$ is also a white noise process which drives the correlated or the Markov bias model of the IMU. The subscript of a and g refer to accelerometer and gyroscope, respectively. This is the IMU output error model described in section 2.1.4.

3.5 EKF For Tight INS/VISNAV Integration

The fusion of VISNAV with INS essentially follows the standard EKF formulation. First the Kalman gain is computed when the measurement update occurs at $t = t_k$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_t^{\mathrm{T}} (\mathbf{H}_t \mathbf{P}_k^- \mathbf{H}_t^{\mathrm{T}} + \mathbf{R}_{\boldsymbol{\nu}_t})^{-1} \tag{3.14}$$

The matrix $\mathbf{P}_k^-$ is the *apriori* state vector covariance at $t = t_k$ *before* incorporating the measurement update. The *aposterori* state covariance *after* measurement update is given

$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{H}_t \mathbf{P}_k^- \tag{3.15}$$

and the state vector update is determined by

$$\delta \mathbf{x}_k^+ = \mathbf{K}_k \delta \mathbf{y}_t \tag{3.16}$$

The superscript $(+)$ and $(-)$ represents quantities before and after a measurement updated, respectively. The updated estimate of the state vector $\mathbf{x}_k^+$ is computed by the following except the attitude states ($\boldsymbol{\alpha}^n$)

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \delta \mathbf{x}_k^+ \tag{3.17}$$

The updated attitude angles $(\delta \boldsymbol{\alpha}^n)^+$ are used to construct a rotation matrix defined as $\mathbf{C}_{n'}^n$. The updated rotation matrix is computed by

$$(\mathbf{C}_n^b)^+ = \mathbf{C}_{n'}^n (\mathbf{C}_n^b)^- \tag{3.18}$$

The updated roll, pitch and yaw can be computed by the following equation as shown

in

$$\phi^+ = \tan^{-1}\left(\frac{c_{23}^+}{c_{33}^+}\right) \tag{3.19a}$$

$$\theta^+ = \sin^{-1}\left(c_{13}^+\right) \tag{3.19b}$$

$$\psi^+ = \tan^{-1}\left(\frac{c_{12}^+}{c_{11}^+}\right) \tag{3.19c}$$

where c_{ij}^+ is the i^{th} row j^{th} column element of $(\mathbf{C}_n^b)^+$

3.6 IEKF For Tight INS/VISNAV Integration

IEKF is a variant of EKF and consists two steps: Time update and measurement update. The principle of IEKF is to linearize the nonlinear time and measurement update equations around an optimal estimate of the state, then iteratively solve the nonlinear equations using the linearized approximation. The idea behind the IEFK is this: Since the measurement update equation is derived from a nonlinear equation, the state requires multiple iterative updates before it converges to its final value. Thus, the only difference between the IEKF and EKF is that in an IEKF Equation 3.16 is solved iteratively until convergence. More specifically, during the measurement update at $t = t_k$, the following steps are performed

1. Initialize the IEKF estimate by

$$\hat{\mathbf{x}}_{k,0}^+ = \hat{\mathbf{x}}_k^- \tag{3.20}$$

$$\mathbf{P}_{k,0}^+ = \mathbf{P}_k^- \tag{3.21}$$

2. For $j = 0, 1, \ldots, N_{iter}$, compute the following equations where N_{iter} is number of

iteration

$$\mathbf{H}_{k,j} = \mathbf{H}_t\big|_{\hat{\mathbf{x}}^+_{k,j}} \tag{3.22}$$

$$\mathbf{K}_{k,j} = \mathbf{P}^-_k \mathbf{H}^{\mathrm{T}}_{k,j} (\mathbf{H}_{k,j} \mathbf{P}^-_k \mathbf{H}^{\mathrm{T}}_{k,j} + \mathbf{R}_{\boldsymbol{\nu}_l})^{-1} \tag{3.23}$$

$$\mathbf{P}^+_{k,j+1} = \mathbf{P}^-_k - \mathbf{K}_{k,j} \mathbf{H}_{k,j} \mathbf{P}^-_k \tag{3.24}$$

$$\delta\mathbf{x}^+_{k,i+1} = \mathbf{K}_{k,j} [\mathbf{y}_k - h(\hat{\mathbf{x}}^+_{k,j}) + H_{k,j} \delta\mathbf{x}^+_{k,j}] \tag{3.25}$$

$$\hat{\mathbf{x}}^+_{k,j+1} = \hat{\mathbf{x}}^-_k + \delta\mathbf{x}^+_{k,j+1} \tag{3.26}$$

The state estimate is iteratively updated using equation 3.17, 4.124 and 3.19 until it converges; N_{iter} is the number of iteration when $\delta\mathbf{x}$ is smaller than a pre-defined tolerance.

3. The final state and covariance estimate are given

$$\hat{\mathbf{x}}^+_k = \hat{\mathbf{x}}^-_{k,N_{iter}+1} \tag{3.27}$$

$$\mathbf{P}^+_k = \mathbf{P}^-_{k,N_{iter}+1} \tag{3.28}$$

To show that IEKF indeed has smaller linearization errors than the EKF, Figure 3.2 graphically depicts the difference between the EKF and IEKF using the yaw attitude state of an INS/VISNAV filter. The figure shows the results of 1000 different measurement updates. The EKF solves Equation 3.16 once while the IEKF solves it iteratively multiple times. In the case of the IEKF, after 10 iterations the state has converged close to one yaw value. In the case of the EKF, the estimates have a rather large spread about some central yaw value.

3.7 Simulation Results of Tight INS/VISNAV Integration

In this simulation, a trajectory is simulated in which a UAV is orbiting an area at an altitude between 300 to 400 meters above ground. The measurements of a low cost

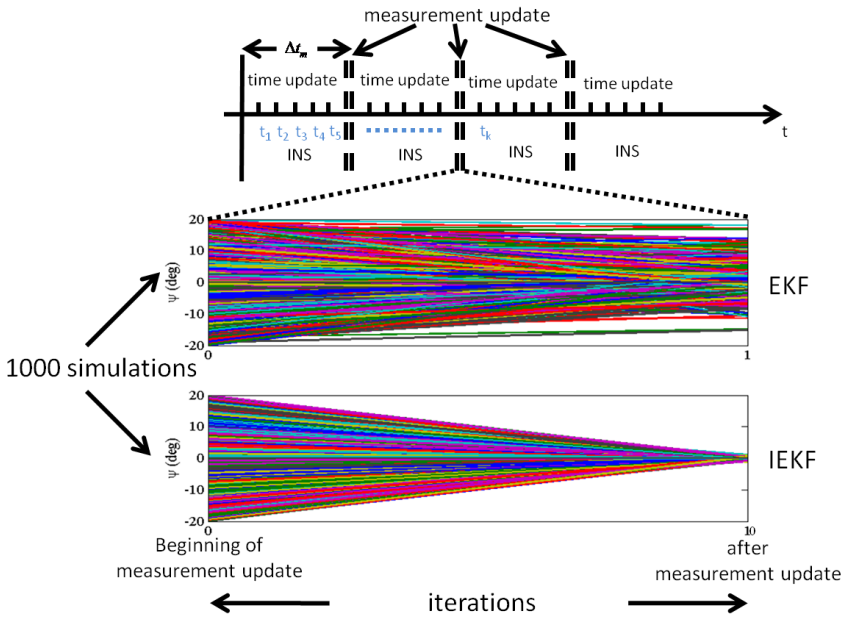

Figure 3.2: EKF versus IEKF

consumer grade IMU and a digital camera are also simulated. The simulation length is 2 minutes and the trajectory is depicted in Figure 3.3 where the green dots are the landmarks and blue line is the simulated trajectory. The parameters used to simulate IMU and camera measurements are tabulated in Table 3.1. The simulation results of position estimates are shown in Figure 3.4. The reference and estimated trajectories are shown on the left hand side. The estimation error is depicted on the right hand side.

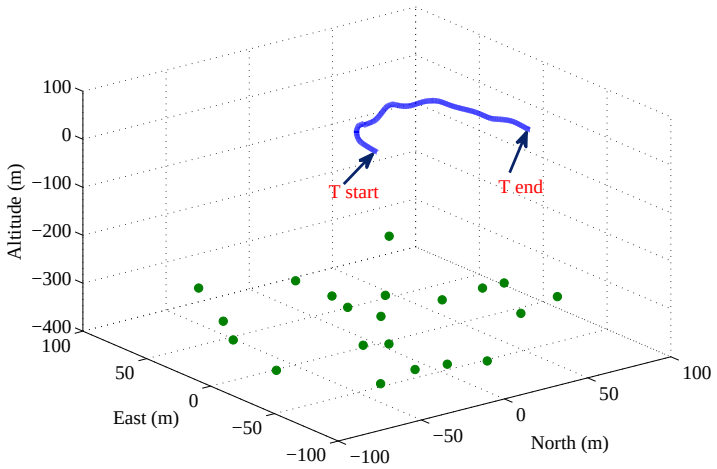

Figure 3.3: Simulated Trajectory and Landmarks

Table 3.1: IMU and Camera Parameters for Simulation

Sensor	Parameters	Value
Gyro	τ_g $\sigma_{\mu_{1,g}}$ σ_{w_g}	300 sec 0.1 deg 0.3 deg
Accel	τ_a $\sigma_{\mu_{1,a}}$ σ_{w_a}	100 sec $1.2 \times 10^{-3} g$ $9 \times 10^{-3} g$
Camera	Horizontal Resolution Vertical Resolution f_x f_y σ_u σ_v	1920 px 1080 px 2137 px 2133 px 3 px 3 px

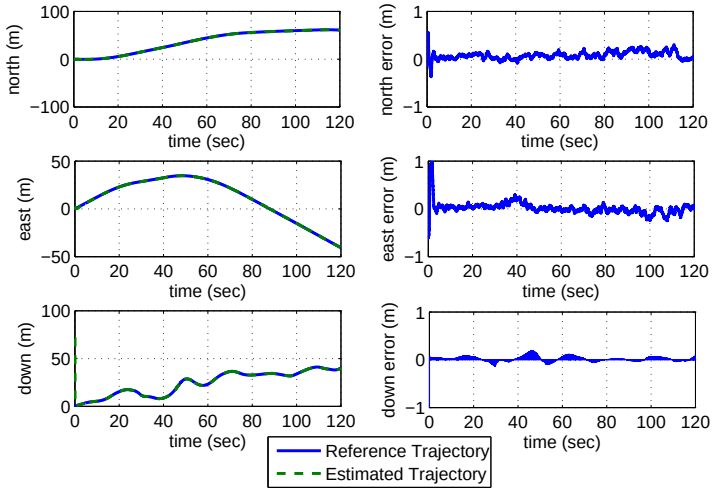

Figure 3.4: Simulation Result of Tight Integration

3.8 Multiple Minima Problem

When using an EKF or IEKF in the tight integration approach, the INS position and attitude solutions are used to linearize the measurement equation. Unfortunately, this measurement equation has multiple local minima or potential solutions which are close to each other [23, 50]. Thus, error in the INS solution used to linearize the measurement equation can cause the EKF to converge to a wrong solution or, even worse, to diverge. The problem cannot be solved by using an IEKF because repeated iterations do not guarantee convergence. In some instances, the IEKF shows limit cycle behavior where for certain landmark geometries, the filter would oscillate about a minimum and never converges. This is particularly true in environments where landmarks being used for camera navigation solution are in close proximity to the user.

This phenomenon can be better understood if we examine the problem graphically.

Figure 3.5 illustrates the existence of multiple local minima for different landmark-camera distances. Three different landmark-camera distances are shown. Each image is taken at a certain camera's pose and the pixel position of the landmarks are measured and, thus, known. For a given pose, we reproject the landmarks onto the camera image plane. The sum of the squares of the distance difference between the reprojected and measured pixel coordinates is the cost function J [22]. Mathematically, this is given as:

$$J = \sum_{i=1}^{N} \{(u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2\} \tag{3.29}$$

where N is the total number of Landmarks. $(\hat{u}_i, \hat{v}_i)$ and (u_i, v_i) are the reprojected and measured pixel location of i^{th} landmark, respectively. In normal situation, only one set of camera's pose (ignore the set with negative depth) would produce the same result as the reference image and this pose would minimize J. However, as shown in Figure 3.5(a), multiple local minima can exist and they are closer to each other if camera-landmark distance is smaller. This means that if the error of the INS solution used to linearize the measurement equation is large (e.g., prior 2 in Figure 3.5(a)), there is high potential that the solution would converge to an erroneous local minimum. When the camera-landmark distance is larger (as shown in Figure 3.5(b) and (c)), so is the distance between local minima and the solution is less sensitive to the INS error. This is why the tight integration architecture is not robust when using low cost inertial sensors. The next chapters discuss two solutions to this problem.

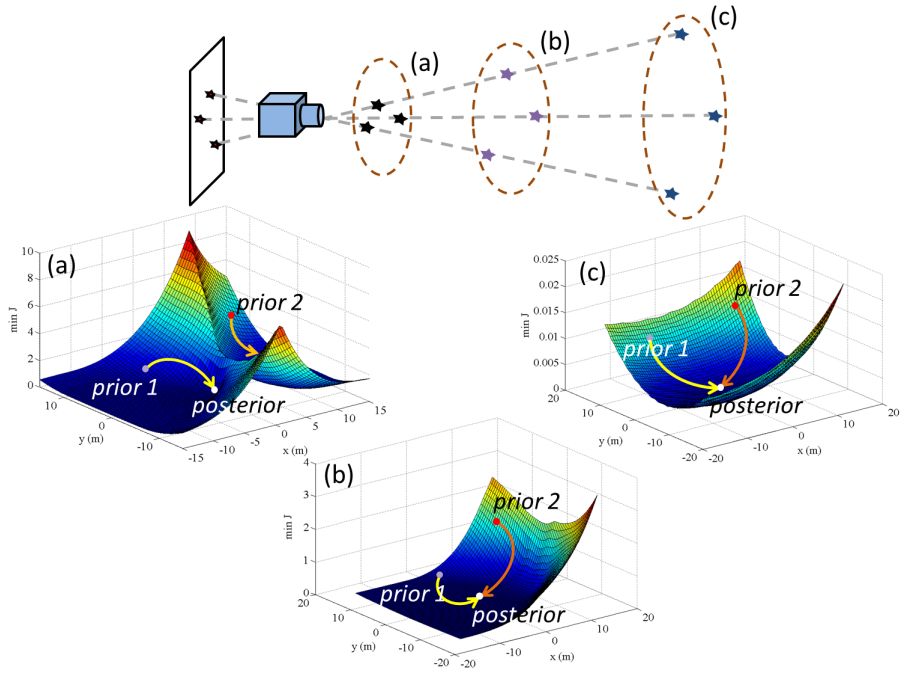

Figure 3.5: Multiple Local Minimum of Tight Integration

Chapter 4

Loosely Coupled INS/VISNAV Integration System

In this chapter, a solution to the local minimum problem of tight architecture is discussed. This alternative is named loosely coupled INS/VISNAV integration. In the loosely integrated INS/VISNAV fusion, both VISNAV and INS can independently generate the navigation state vector. The VISNAV system can operate as a standalone navigation system. The navigation solution from VISNAV is calculated using Perspective-n-Point (PnP) algorithm. Three analytical PnP algorithms will be introduced in this chapter. These solutions from VISNAV and INS are linearly integrated using an EKF. Here, the integration occurs at the position and attitude level. The solution from the camera is used to arrest the drift of the INS solution but no INS information is used by the camera at all. The block diagram of loose integration is shown in Figure 4.1.

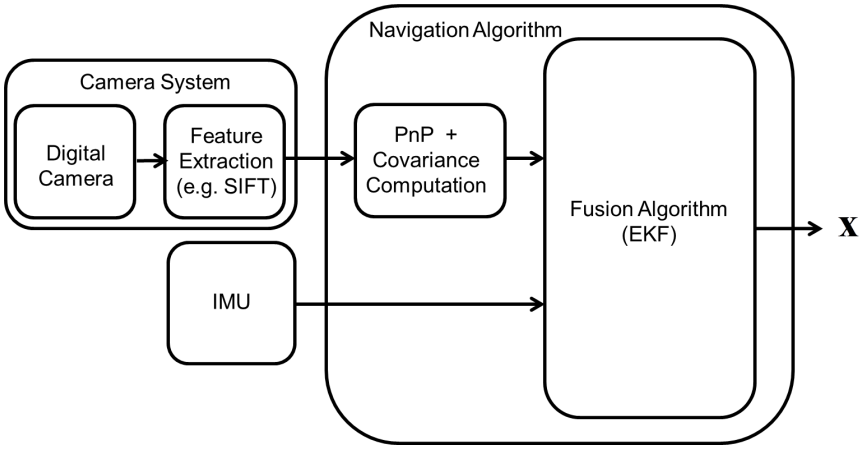

Figure 4.1: Functional block diagram of loose integration

4.1 Perspective-n-Point Problem

The loose integration relies on the fact that VISNAV can independently generate a position and attitude solution. Using VISNAV alone to determine position and attitude is known as the Perspective-n-Point (PnP) problem. The scope of PnP is to determine the position and attitude of a calibrated camera given a set of n three dimensional landmarks at known locations and their corresponding projections on the camera image plane. It is a minimization problem which can be modeled mathematically as

$$\{\mathbf{C}_n^b, \mathbf{p}^n\} = \text{argmin} J \tag{4.1}$$
$$\text{subject to } (\mathbf{C}_n^b)^{\mathrm{T}}\mathbf{C}_n^b = \mathbf{I}_3, \quad \det(\mathbf{C}_n^b) = 1$$

where the cost function J is the squared sum of measurement error:

$$J = \sum_{i=1}^{N} \|\delta u_i + \delta v_i\|^2 \tag{4.2}$$

where $\delta u_i = u_i - \hat{u}_i$ and $\delta v_i = v_i - \hat{v}_i$ are the differences between measured pixel locations and the computed pixel location parameterized by $\mathbf{C}_n^b$ and $\mathbf{p}^n$. This is a well-known problem which has received much attention due to its wide application

on robotics and computer vision [41]. This problem is also known as space resection in the photogrammetry [51]. Since each landmark projected on a two dimensional image plane can generate two measurements (u and v), it only requires 3 landmarks to uniquely determine the camera's position and attitude (6-DOF). This is known as P3P problem [45, 46, 52] and large number of approaches have been proposed since its earliest discussions back in 1841 [45, 53]. In reality, since the pixel projections are corrupted by noise, P3P solutions are more sensitive to the influence of noise. The n-point geometry where $n > 3$ provides redundancy which can greatly improve the solution's robustness.

The PnP solutions can be categorized into iterative and non-iterative approaches. Iterative approaches are generally less sensitive to pixel noises. However, the selection of initial conditions and convergence issue are the common challenges with the iterative approaches. In addition, iterative approaches are more computational intensive and can impose bigger challenge for real-time applications. This is especially true if RANSAC is applied for outlier removal or the VISNAV is deployed on small platforms such as small UAVs where the computational power is limited. On the other hand, even through non-iterative approaches are more sensitive to large pixel noises compare to iterative methods, they are more a practical choice in many applications since they demand much less computing resources than the iterative approaches. In the following discussion, three non-iterative approaches are discussed. The methods discussed are those which are considered to be ideal candidates for loose integration with INS. The methods discussed are: DLT, EPnP and DLS algorithms. Their performances and the challenges of integrating the algorithms into loosely coupled architecture will also be discussed.

4.1.1 Direct Linear Transformation (DLT) Method

The first PnP algorithm that we introduce for estimating the position and attitude is called the *Direct Linear Transformation* (DLT) [41]. Recall that the landmark projection is governed by Equation 2.16 Once the entries of the matrix $\mathbf{H}$ are estimated, the direction cosine matrix $\mathbf{C}_n^b$ and camera position vector $\mathbf{p}^n$ can be extracted. To do this in DLT, $\mathbf{H}$ is "vectorized" by defining

$$\boldsymbol{\eta} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{21} & \cdots & h_{34} \end{bmatrix}^T \tag{4.3}$$

where h_{ij} is the element in the i^{th} row and j^{th} column of $\mathbf{H}$. Since $\mathbf{e}_i$ and $\mathbf{H}\boldsymbol{\lambda}_i$ are colinear vectors, their cross product is zero. Using Equation 2.16, this yields the following linear equation in $\boldsymbol{\eta}$:

$$\mathbf{0} = \begin{bmatrix} \mathbf{Z}_{1\times 4} & -\boldsymbol{\lambda}_i^T & v_i\boldsymbol{\lambda}_i^T \\ \boldsymbol{\lambda}_i^T & \mathbf{Z}_{1\times 4} & -u_i\boldsymbol{\lambda}_i^T \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{34} \end{bmatrix} = \mathbf{B}_i\boldsymbol{\eta} \tag{4.4}$$

Given N landmarks and their associated pixel coordinates, N of the $\mathbf{B}_i$ matrices are stacked to construct the following linear equation in $\boldsymbol{\eta}$:

$$\mathbf{0} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \mathbf{B}_3 \\ \vdots \\ \mathbf{B}_N \end{bmatrix} \boldsymbol{\eta} = \mathbf{B}\boldsymbol{\eta} \tag{4.5}$$

The matrix $\mathbf{B} \in \mathbb{R}^{2N\times 12}$ where N is the number of feature points. If the information used to construct $\mathbf{B}$ (i.e. landmark position coordinates and their associated pixel coordinates on the image plane) is error-free, then $\text{rank}(\mathbf{B}) < 12$ and $\boldsymbol{\eta}$ would lie in the

null space of $\mathbf{B}$. However, in the presence of landmark and pixel errors one can only form an approximation to $\mathbf{B}$ which will be full rank. This full rank version of $\mathbf{B}$ formed using error-corrupted landmark and pixel coordinates is denoted as $\widetilde{\mathbf{B}}$. It is related to $\mathbf{B}$ by:

$$\widetilde{\mathbf{B}}\boldsymbol{\eta} = (\mathbf{B} + \Delta\mathbf{B})\,\boldsymbol{\eta} \neq \mathbf{0} \tag{4.6}$$

If the vector $\hat{\boldsymbol{\eta}}$ is desired to be the optimal estimate of $\boldsymbol{\eta}$ in the least squares sense, one can calculate $\hat{\boldsymbol{\eta}}$ (where "$\wedge$" indicates an estimate) using total least squares (TLS) [54, 55] to minimize the following cost function :

$$\min_{\boldsymbol{\eta},\,\Delta\mathbf{B}} \|(\mathbf{B} + \Delta\mathbf{B})\,\boldsymbol{\eta}\|_F \tag{4.7}$$

The total least squares solution to this minimization problem involves performing a singular value decomposition (SVD) of $\widetilde{\mathbf{B}} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^T$ where:

$$\begin{aligned}\widetilde{\mathbf{B}} &= \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^T \\ &= \left[\begin{array}{c|c|c|c} \widetilde{\mathbf{u}}_1 & \widetilde{\mathbf{u}}_2 & \cdots & \widetilde{\mathbf{u}}_{12} \end{array}\right] \begin{bmatrix} \widetilde{\sigma}_1 & 0 & \cdots & 0 \\ 0 & \widetilde{\sigma}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tilde{\sigma}_{12} \end{bmatrix} \left[\begin{array}{c|c|c|c} \widetilde{\mathbf{v}}_1 & \widetilde{\mathbf{v}}_2 & \cdots & \widetilde{\mathbf{v}}_{12} \end{array}\right]^T \end{aligned} \tag{4.8}$$

The solution to the minimization problem posed in Equation 4.7 is

$$\widehat{\boldsymbol{\eta}} = \alpha\widetilde{\mathbf{v}}_{12} \tag{4.9}$$

where α is a scaling constant and $\widetilde{\mathbf{v}}_{12}$ is the column of $\widetilde{\mathbf{V}}$ associated with the smallest singular value of $\widetilde{\mathbf{B}}$. The scaling constant is needed because $\widetilde{\mathbf{v}}_{12}$ generated by the SVD is only a unit vector pointing in the same direction as $\hat{\boldsymbol{\eta}}$. For ease of notation, let us denote this unit vector $\widetilde{\mathbf{v}}_{12} = \hat{\bar{\boldsymbol{\eta}}}$. The scaling constant α can be easily estimated using the camera calibration parameters $\mathbf{M}$ as follows:

$$\widehat{\mathbf{C}}_n^b = \mathbf{M}^{-1}\left(\alpha\widehat{\overline{\mathbf{H}}}_1\right) \tag{4.10}$$

where

$$\widehat{\overline{\mathbf{H}}}_1 = \begin{bmatrix} \widehat{\overline{\eta}}_1 & \widehat{\overline{\eta}}_2 & \widehat{\overline{\eta}}_3 \\ \widehat{\overline{\eta}}_5 & \widehat{\overline{\eta}}_6 & \widehat{\overline{\eta}}_7 \\ \widehat{\overline{\eta}}_9 & \widehat{\overline{\eta}}_{10} & \widehat{\overline{\eta}}_{11} \end{bmatrix} \tag{4.11}$$

Since $\widehat{\mathbf{C}}_n^b$ has to be a proper transformation matrix, the following constraint must be satisfied:

$$\det\left(\widehat{\mathbf{C}}_n^b\right) = \det\left(\alpha \mathbf{M}^{-1} \widehat{\overline{\mathbf{H}}}_1\right) = \alpha^3 \det\left(\mathbf{M}^{-1} \widehat{\overline{\mathbf{H}}}_1\right) = 1 \tag{4.12}$$

which leads to:

$$\alpha = \frac{1}{\left(\det\left(\widehat{\overline{\mathbf{C}}}_n^b\right)\right)^{\frac{1}{3}}} \tag{4.13}$$

Thus, $\widehat{\boldsymbol{\eta}} = \alpha \widehat{\overline{\boldsymbol{\eta}}}$ and $\widehat{\boldsymbol{\eta}}$ is used to form $\widehat{\mathbf{H}} = \left[\begin{array}{c|c} \widehat{\mathbf{H}}_1 & \widehat{\mathbf{H}}_2 \end{array}\right]$ from which it follows that the navigation states of interest are given by:

$$\widehat{\mathbf{C}}_n^b = \alpha \mathbf{M}^{-1} \widehat{\overline{\mathbf{H}}}_1 \tag{4.14}$$

$$\widehat{\mathbf{p}}^n = \widehat{\overline{\mathbf{H}}}_1^{-1} \widehat{\overline{\mathbf{H}}}_2 \tag{4.15}$$

Numerical errors in computing the $\widehat{\mathbf{C}}_n^b$ matrix may cause it to become non-orthogonal. Failure to compensate for this can cause large attitude errors. In order to ensure that the $\widehat{\mathbf{C}}_n^b$ matrix is orthogonal, the SVD of the $\widehat{\mathbf{C}}_n^b$ is taken as follows:

$$\widehat{\mathbf{C}}_n^b = \widehat{\mathbf{U}} \widehat{\boldsymbol{\Sigma}} \widehat{\mathbf{V}}^T \tag{4.16}$$

The SVD is then used to determine the orthogonal matrix $\breve{\mathbf{C}}_n^b$ according to:

$$\breve{\mathbf{C}}_n^b = \widehat{\mathbf{U}} \widehat{\mathbf{V}}^T \tag{4.17}$$

Normalizing Pixel and Landmark Coordinates

Although the algorithm presented above appears sound, if it is put into practice without adjustments it will produce poor estimates for the camera's position and attitude. This occurs because some of the equations above contain ill-conditioned matrices, particularly the matrices $\mathbf{B}_i$ and $\mathbf{B}$ in Equations 4.4 and 4.5. The matrices are ill-conditioned because of the relative magnitudes of the elements of $\boldsymbol{\lambda}_i$ and $\mathbf{e}_i$. For example, if Earth Centered Earth Fixed (ECEF) coordinates are being used to describe the physical location of landmarks, then the first three elements of $\boldsymbol{\lambda}_i$ can be on the order of 10^6 m while the fourth element is unity. Similarly, if high resolution cameras (1280×1024 pixels) are being used, the first three elements of $\boldsymbol{e}_i$ are the order of 10^3 pixels while the fourth element is unity. This makes the $\mathbf{B}$ and $\Delta\mathbf{B}$ matrices ill-conditioned.

The remedy for this is to scale $\mathbf{e}_i$ and $\boldsymbol{\lambda_i}$ before the DLT is performed. If the scaled versions of these vectors is denoted as $\mathbf{e}'_i$ and $\boldsymbol{\lambda}'_i$, then the relation between scaled and original vector is given by:

$$\mathbf{e}'_i = \mathbf{T}_e \mathbf{e}_i \tag{4.18}$$

$$\boldsymbol{\lambda}'_i = \mathbf{T}_\lambda \boldsymbol{\lambda}_i \tag{4.19}$$

There several options for scaling that can be used, but in the work described here the following scaling matrices are used:

$$\begin{bmatrix} \lambda'_{x,i} \\ \lambda'_{y,i} \\ \lambda'_{z,i} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{s_x} & 0 & 0 & -\frac{\sqrt{3}}{s_x} m_x \\ 0 & \frac{\sqrt{3}}{s_y} & 0 & -\frac{\sqrt{3}}{s_y} m_y \\ 0 & 0 & \frac{\sqrt{3}}{s_z} & -\frac{\sqrt{3}}{s_z} m_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_{x,i} \\ \lambda_{y,i} \\ \lambda_{z,i} \\ 1 \end{bmatrix} \tag{4.20}$$

where m_x, m_y, and m_z are the mean value of all $\boldsymbol{\lambda}_i$ in x, y and z direction, respectively. And s_x, s_y, s_z are the associated standard deviation. The concept behind this transformation is to move the origin to the mean of all $\boldsymbol{\lambda}_i$ and set the average distance from

feature points to the origin equal to $\sqrt{3}$ for all three directions. So that all four components for i^{th} feature points have the same order of magnitude. For the measurement $\mathbf{e}_i$, the normalization is similar to what was discussed for $\boldsymbol{\lambda}_i$. That is,

$$\begin{bmatrix} u_i' \\ v_i' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{s_u} & 0 & -\frac{\sqrt{2}}{s_u}m_u \\ 0 & \frac{\sqrt{2}}{s_v} & -\frac{\sqrt{2}}{s_v}m_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \tag{4.21}$$

where m_u, m_v, s_u, s_v are the mean and standard deviation of all intercepts in horizontal and vertical direction, respectively. With normalization, the relationship between pixels and landmark coordinates is recast as:

$$\mathbf{e}_i' = \mathbf{T}_e \mathbf{H} \mathbf{T}_\lambda^{-1} \boldsymbol{\lambda}_i' = \mathbf{H}' \boldsymbol{\lambda}_i' \tag{4.22}$$

where $\mathbf{H}' = \mathbf{T}_e \mathbf{H} \mathbf{T}_\lambda^{-1}$. Thus, instead of using the DLT to solve for $\mathbf{H}$, the pixel and landmarks are normalized, and the DLT is used to solve for $\mathbf{H}'$. Then $\mathbf{H}$ is denormalized from $\mathbf{H}'$ according to

$$\mathbf{H} = \mathbf{T}_e^{-1} \mathbf{H}' \mathbf{T}_\lambda \tag{4.23}$$

Finally $\mathbf{H}$ is used to compute $\hat{\mathbf{C}}_n^b$ and $\hat{\mathbf{p}}^n$, starting with Equation 4.10. It is important to note that normalizing is not an optional procedure; failure to normalize the pixel and landmark coordinates will lead to poor results. In the covariance estimation that follows, normalization will be used again to mitigate the effects of ill-conditioned matrices. When using normalized coordinates for pixels and landmarks, the noise levels for pixel registration and landmark surveying also change due to the normalization. They are assumed to be Gaussian and will remain Gaussian but their covariance will change. For pixel registration this is calculated as:

$$\mathcal{E}\left\{ \begin{bmatrix} \delta u_i' \\ \delta v_i' \end{bmatrix} \begin{bmatrix} \delta u_i' & \delta v_i' \end{bmatrix} \right\} = \begin{bmatrix} \sigma_{v_i'}^2 & \sigma_{u_i'v_i'} \\ \sigma_{u_i'v_i'} & \sigma_{u_i'}^2 \end{bmatrix} \tag{4.24}$$

where $\sigma^2_{v'_i}$, $\sigma_{u'_i v'_i}$, and $\sigma^2_{u'_i}$ are calculated from

$$\mathbf{R}_{e'} = \begin{bmatrix} \sigma^2_{v'_i} & \sigma_{u'_i v'_i} & 0 \\ \sigma_{u'_i v'_i} & \sigma^2_{u'_i} & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{T}_e \begin{bmatrix} \sigma^2_{v_i} & \sigma_{u_i v_i} & 0 \\ \sigma_{u_i v_i} & \sigma^2_{u_i} & 0 \\ 0 & 0 & 0 \end{bmatrix} (\mathbf{T}_e)^T \tag{4.25}$$

Using the normalized pixel and landmark coordinates and their normalized covariances to calculate the covariance of the projective transform will result in the covariance being calculated for $\hat{\boldsymbol{\eta}}'$ and not $\hat{\boldsymbol{\eta}}$. One can write a relationship between $\hat{\boldsymbol{\eta}}'$ and $\hat{\boldsymbol{\eta}}$ of the form $\hat{\boldsymbol{\eta}}' = \mathbf{T}_{\hat{\eta}} \hat{\boldsymbol{\eta}}$, where $\mathbf{T}_{\hat{\eta}}$ is the Kronecker tensor product between $\mathbf{T}_e$ and $\left(\mathbf{T}_\lambda^{-1}\right)^T$ or $\mathbf{T}_{\hat{\eta}} = \mathbf{T}_e \otimes \left(\mathbf{T}_\lambda^{-1}\right)^T$. Therefore:

$$\mathbf{R}_{\hat{\eta}} = \mathbf{T}_\eta^{-1} \mathbf{R}_{\hat{\eta}'} \left(\mathbf{T}_\eta^{-1}\right)^T \tag{4.26}$$

With this relationship established, the covariance analysis will also utilize it to mitigate the effect of ill-conditioned matrices on calculating the measurement covariance.

4.1.2 Efficient PnP (EPnP) Method

Efficient PnP (EPnP) approach is a non-iterative PnP algorithm proposed by Lepetit, Moreno-Noguer and Fua in 2009 [56]. The authors suggest this algorithm is more efficient than other iterative and non-iterative algorithms but with a slight loss in accuracy compared with iterative approaches. Their approach has $O(n)$ complexity which makes it attractive for dealing with large number of landmarks ($N \gg 0$). Other algorithms can have exponential computational complexity with respect to n landmarks. In general, the solution is more robust to pixel noises if the number of selected landmarks increase. This can be beneficial in feature rich environments where noise effects can be suppressed and the computation time increases linearly. To achieve $O(n)$ complexity, the algorithm doesn't use all the feature points for calculation. Instead, it defines four virtual *control points* and expresses all the landmarks' location as weighted sum of those four control

points. In mathematical terms this is written as.

$$\boldsymbol{\lambda}_i^n = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^n \tag{4.27}$$

where

$$\sum_{j=1}^{4} \alpha_{ij} = 1 \tag{4.28}$$

$\mathbf{c}_j^n$ where $j = 1, \ldots, 4$ are the control points expressed in a inertial navigation frame (n-frame). Their coordinates are usually set at $\mathbf{c}_1^n = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$, $\mathbf{c}_2^n = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^{\mathrm{T}}$, $\mathbf{c}_3^n = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{c}_4^n = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$. However, their locations expressed in c-frame are unknown and are to be determined. $\boldsymbol{\lambda}_i^n$ is i^{th} landmark's location written in cartesian coordinate. The superscript n indicates the fact that the points are resolved in a navigation frame. α_{ij} are the homogeneous barycentric coordinates which can be uniquely and easily determined. To do this, we expressed the landmarks' location using the homogeneous coordinate and form a matrix $\mathbf{C}$ given by

$$\mathbf{C} = \begin{bmatrix} \mathbf{c}_1^n & \mathbf{c}_2^n & \mathbf{c}_3^n & \mathbf{c}_4^n \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.29}$$

Now α_i can be calculated as

$$\boldsymbol{\alpha}_i = \mathbf{C}^{-1} \boldsymbol{\lambda}_i^h \tag{4.30}$$

where the superscript h denotes a homogeneous coordinate expression. Note that the relation of Equation 4.27 is also applicable to points expressed in a camera frame (c-frame).

$$\boldsymbol{\lambda}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c \tag{4.31}$$

The projection of a 3-D point resolved in the camera frame (c-frame) on to a 2-D image

plane and written in homogeneous coordinates can be expressed

$$\mathbf{e}_i = s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{M}\boldsymbol{\lambda}_i^c \tag{4.32}$$

where $\mathbf{M}$ is the camera intrinsic matrix defined in Equation 2.17 and $\mathbf{e}_i = s_i \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^{\mathrm{T}}$ is i^{th} feature point projection on the image plane written in homogeneous coordinate. If we substitute Equation 4.31 and 2.17 into Equation 4.32, the equation can be written as

$$s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} c_{x,j}^c \\ c_{y,j}^c \\ c_{z,j}^c \end{bmatrix} \tag{4.33}$$

Note that the third row of Equation 4.33 implies that $s_i = \sum_{j=1}^{4} \alpha_{ij} c_{z,j}$. By substituting this expression into the first two rows, we can rewrite the equation above as the following two linear equations for each feature point:

$$\sum_{j=1}^{4} \alpha_{ij} f_x c_{x,j}^c + \alpha_{ij}(p_x - u_i) c_{z,j}^c = 0 \tag{4.34a}$$

$$\sum_{j=1}^{4} \alpha_{ij} f_y c_{y,j}^c + \alpha_{ij}(p_y - v_i) c_{z,j}^c = 0 \tag{4.34b}$$

If we put the four control points into a 12×1 vector $\mathbf{z}^c = \begin{bmatrix} \mathbf{c}_1^{c\mathrm{T}} & \mathbf{c}_2^{c\mathrm{T}} & \mathbf{c}_3^{c\mathrm{T}} & \mathbf{c}_4^{c\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ and rewrite Equation 4.34

$$\mathbf{0} = \begin{bmatrix} \alpha_{i1} f_x & 0 & \alpha_{i1}(p_x - u_i) & \alpha_{i2} f_x & \dots & \alpha_{i4}(p_x - u_i) \\ 0 & \alpha_{i1} f_y & \alpha_{i1}(p_y - v_i) & 0 & \dots & \alpha_{i4}(p_y - v_i) \end{bmatrix} \begin{bmatrix} c_{x,1}^{\mathrm{T}} \\ c_{y,1}^{\mathrm{T}} \\ c_{z,1}^{\mathrm{T}} \\ \vdots \\ c_{z,4}^{\mathrm{T}} \end{bmatrix} = \mathbf{A}_i \mathbf{z}^c \tag{4.35}$$

Given N landmarks and their associated pixel coordinates, N of the $\mathbf{A}_i$ matrices are stacked to construct the following linear equation in $\mathbf{z}^c$

$$\mathbf{0} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \vdots \\ \mathbf{A}_N \end{bmatrix} \mathbf{z}^c = \mathbf{A}\mathbf{z}^c \tag{4.36}$$

The matrix $\mathbf{A} \in \mathbb{R}^{2N \times 12}$ where N is the number of feature points. Similar to the arguments used in deriving Equation 4.9. If pixel projections (u_i and v_i) are noise free, the solution $\mathbf{z}$ would lie in the null space of $\mathbf{A}$ and can be expressed as

$$\mathbf{z} = \sum_{i=1}^{n} \beta_i \mathbf{v}_i \tag{4.37}$$

where $\mathbf{v}_i$ is the right singular vector of $\mathbf{A}$ corresponding to $n \leq 4$ zero singular values. In theory, the dimension of the null space is one ($n = 1$ meaning one zero singular value) for perspective projection and increases to four ($n = 4$) if the camera projection becomes orthographic [56]. This assumption is usually made if the distance to an object is far or the focal length of the lens is large. We normally consider the effective dimension to vary from 1 to 4 depending on the camera projection type. In the presence of landmarks and pixel errors, the noise corrupted matrix, denoted as $\tilde{\mathbf{A}}$, is generally full rank. Therefore, the solution no longer lies in the null space of $\mathbf{A}$. That is,

$$\tilde{\mathbf{A}}\mathbf{z}^c = (\mathbf{A} + \Delta\mathbf{A})\mathbf{z}^c \neq 0 \tag{4.38}$$

where $\Delta\mathbf{A}$ is the noise component of $\mathbf{A}$. Assuming vector $\hat{\mathbf{z}}^c$ is the optimal estimate of $\mathbf{z}^c$ in the least square sense, the solution for minimizing the following cost function based on Total Least Square (TLS) [54, 55]

$$\min_{\mathbf{z}^c,\, \Delta\mathbf{A}} \|(\mathbf{A} + \Delta\mathbf{A})\, \mathbf{z}^c\| \tag{4.39}$$

lies in the direction of the right singular vector of $\tilde{\mathbf{A}}$ which correspond to the minimum singular value. Recall that the effective dimension for the noise free matrix $\mathbf{A}$ can vary from 1 to 4 depending on the camera projection assumption. The optimal estimate $\hat{\mathbf{z}}^c$ is thus

$$\hat{\mathbf{z}}^c = \sum_{i=1}^{4} \beta_i \tilde{\mathbf{v}}_i \tag{4.40}$$

where $\tilde{\mathbf{v}}_i$ is the right singular vector of $\tilde{\mathbf{A}}$ associate with each of the four smallest singular values. The next step is to determine to unknown coefficients β_i. The coefficients are calculated based on the fact (constraint) that the distance between any two control points remains unchanged in different reference frames. The coefficients can be calculated based on the assumption of the effective dimension (n) of the null space of $\widetilde{\mathbf{A}}$

For case n = 1:

The estimate is written as $\hat{\mathbf{z}}^c = \beta\tilde{\mathbf{v}}_1$. If we define $\tilde{\mathbf{v}}_1^{[p]}$ as the sub-vector of $\tilde{\mathbf{v}}_1$ which correspond to the p^{th} control point $\mathbf{c}_p$. The distance constraint expression is given by

$$\left\|\beta\tilde{\mathbf{v}}^{[p]} - \beta\tilde{\mathbf{v}}^{[q]}\right\|^2 = \left\|\mathbf{c}_p^n - \mathbf{c}_q^n\right\|^2 = d_{pq}^2 \qquad \forall p,\ q = 1, \ldots 4 \tag{4.41}$$

where d_{pq} is the distance between control points p and q which is given. The subscript 1 in $\tilde{\mathbf{v}}_1$ represents the singular vector is associated with the smallest singular value of $\tilde{\mathbf{A}}$. The computation of β has a close-form solution:

$$\beta = \frac{\sum_{\{p,q\}\in[1:4]} d_{pq} \left\|\tilde{\mathbf{v}}^{[p]} - \tilde{\mathbf{v}}^{[q]}\right\|}{\sum_{\{p,q\}\in[1:4]} \left\|\tilde{\mathbf{v}}^{[p]} - \tilde{\mathbf{v}}^{[q]}\right\|^2} \tag{4.42}$$

For case n = 2:

For $n = 2$, the solution estimate is written as $\mathbf{z}^c = \beta_1\tilde{\mathbf{v}}_1 + \beta_2\tilde{\mathbf{v}}_2$, the distance expression is given as

$$\left\|(\beta_1\tilde{\mathbf{v}}_1^{[p]} + \beta_2\tilde{\mathbf{v}}_2^{[p]}) - (\beta_1\tilde{\mathbf{v}}_1^{[q]} + \beta_2\tilde{\mathbf{v}}_2^{[q]})\right\|^2 = \left\|\mathbf{c}_p^n - \mathbf{c}_q^n\right\|^2 = d_{pq}^2 \qquad \forall p,\ q = 1, \ldots 4 \tag{4.43}$$

β_1 and β_2 are in quadratic terms. To solve for β_1 and β_2, [56] uses a technique called "linearization" in cryptography. It converts a set of quadratic nonlinear equation into

a set of linear equation by defining a vector $\boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{22} \end{bmatrix}^{\mathrm{T}}$ where $\beta_{11} = \beta_1^2$, $\beta_{12} = \beta_1\beta_2$ and $\beta_{22} = \beta_2^2$. The original equation can be re-organized into the form:

$$\mathbf{L}\boldsymbol{\beta} = \mathbf{d} \tag{4.44}$$

where $\mathbf{L} \in \mathbb{R}^{6\times 3}$ formed by the entries of $\tilde{\mathbf{v}}_1$ and $\tilde{\mathbf{v}}_2$. $\mathbf{d} \in \mathbb{R}^6$ is constructed by the squared distance d_{pq}^2. Since this equation is over-determined, $\boldsymbol{\beta}$ can be calculated using pseudo inverse

$$\boldsymbol{\beta} = (\mathbf{L}^{\mathrm{T}}\mathbf{L})^{-1}\mathbf{L}^{\mathrm{T}}\mathbf{d} \tag{4.45}$$

β_1 and β_2 can be solved accordingly.

For case n = 3:

Similar to the case of $n = 2$, we can also define a vector $\boldsymbol{\beta}$ where the entries of $\boldsymbol{\beta}$ are the quadratic terms of $\beta_i,\ \ i = 1, 2, 3$. $\boldsymbol{\beta}$ is written as $\boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{22} & \beta_{23} & \beta_{33} \end{bmatrix}^{\mathrm{T}}$. The distance constraint can also be expressed as

$$\mathbf{L}\boldsymbol{\beta} = \mathbf{d} \tag{4.46}$$

where $\mathbf{L} \in \mathbb{R}^{6\times 6}$, and $\mathbf{d} \in \mathbb{R}^6$. Since $\mathbf{L}$ is a square matrix, the solution can be calculated by matrix inversion:

$$\boldsymbol{\beta} = \mathbf{L}^{-1}\mathbf{d} \tag{4.47}$$

For case n = 4:

In theory, for the case of $n = 4$, the equation is solvable since we only have four unknowns $\beta_i,\ \ i = 1, 2, 3, 4$ but six distance constraint equations. However, the linearization technique treats all $\beta_{ij} = \beta_i\beta_j,\ \ i = 1, 2, 3, 4$ as independent variables which will result in 10 unknowns with only six equations. The work in [56] adopted the relinearization technique proposed by [57]. This approach is a two steps linearization technique. As we will show later, the details of the solution are not relevant to the main focus of this thesis; namely the INS/VISNAV fusion. Thus, we will simply refer the interested readers to

[57] for a detail discussion.

After the calculation of $\boldsymbol{\beta}$, the estimated location of control points expressed in the c-frame ($\hat{\mathbf{z}}^c$) is determined by Equation 4.40. The next step is to compute the attitude and position. Given a set of control points expressed in in n-frame and c-frame $\left\{ \begin{array}{cccc} \mathbf{c}_1^c & \mathbf{c}_2^c & \mathbf{c}_3^c & \mathbf{c}_4^c \end{array} \right\}$ and $\left\{ \begin{array}{cccc} \mathbf{c}_1^n & \mathbf{c}_2^n & \mathbf{c}_3^n & \mathbf{c}_4^n \end{array} \right\}$, we first recover the feature points location by using Equation 4.27 and 4.31. Next, the centroid of the feature points are calculated:

$$\overline{\boldsymbol{\lambda}}^n = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\lambda}_i^n \tag{4.48a}$$

$$\overline{\boldsymbol{\lambda}}^c = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\lambda}_i^c \tag{4.48b}$$

This is used to construct directional vectors which originate from the centroid and point to each individual feature point:

$$\boldsymbol{\gamma}_i^n = \boldsymbol{\lambda}_i^n - \overline{\boldsymbol{\lambda}}^n \tag{4.49a}$$

$$\boldsymbol{\gamma}_i^c = \boldsymbol{\lambda}_i^c - \overline{\boldsymbol{\lambda}}^c \tag{4.49b}$$

Since the vectors expressed in n-frame can be transformed into c-frame by

$$\boldsymbol{\gamma}_i^c = \mathbf{C}_n^c \boldsymbol{\gamma}_i^n \tag{4.50}$$

the rotational matrix can be calculated if we multiply $(\boldsymbol{\gamma}_i^n)^{\mathrm{T}}$ at both sides:

$$\mathbf{C}_n^c = \boldsymbol{\gamma}_i^c (\boldsymbol{\gamma}_i^n)^{\mathrm{T}} \tag{4.51}$$

Since we have N pairs of $\boldsymbol{\gamma}_i^c$ and $\boldsymbol{\gamma}_i^n$, the optimal estimate of the rotation matrix can be calculated in the least square sense [58] as

$$\hat{\mathbf{C}}_n^c = \mathbf{U}\mathbf{V}^{\mathrm{T}} \tag{4.52}$$

where $\mathbf{U}$ and $\mathbf{V}$ are the matrices formed by the left and right singular vectors of $\mathbf{\Sigma} = \mathbf{USV}^{\mathrm{T}}$ obtained from a singular value decomposition. The matrix $\mathbf{\Sigma}$ is constructed as

$$\mathbf{\Sigma} = \sum_{i=1}^{N} \boldsymbol{\gamma}_i^c (\boldsymbol{\gamma}_i^n)^{\mathrm{T}} \tag{4.53}$$

The geometry relationship of the centroid can also be expressed using Equation 2.11:

$$\overline{\boldsymbol{\lambda}}^c = \mathbf{C}_n^c (\overline{\boldsymbol{\lambda}}^n - \mathbf{p}^n) \tag{4.54}$$

Therefore, the position estimate can be solved from:

$$\hat{\mathbf{p}}^n = \overline{\boldsymbol{\lambda}}^n - \mathbf{C}_c^n \overline{\boldsymbol{\lambda}}^c \tag{4.55}$$

4.1.3 Direct Least-Squares (DLS) Method

Direct Least Square approach is another non-iterative PnP algorithm proposed by Hesch and Roumeliotis in 2011 [1]. The authors suggest this method is scalable which is beneficial for large number of feature points. Its solution accuracy outperforms the other PnP methods in that it is close to the Maximum-Likelihood Estimator (MLE) solution of Equation 4.1. The work in [1] express the landmark using geometry different than the model used here and described in section 2.2.1. In order to be consistent with [1], we will adopt the formulation of [1] in this section. The geometry is shown in Figure 4.2. The rest of this thesis will follow the line-of-sight formulation as shown in Equation 2.11.

In [1] the image projection is as:

$$\mathbf{y}_i' = \overline{\mathbf{b}}_i^c + \boldsymbol{\nu}_i' \tag{4.56a}$$

$$\mathbf{b}_i^c = \mathbf{C}_n^c \boldsymbol{\lambda}_i^n + \mathbf{p}^c \tag{4.56b}$$

where $\mathbf{y}_i' \in \mathbb{R}^3$ is the measured line of sight vector of i^{th} landmark. It can be constructed by pixel measurement (u_i, v_i) and $\boldsymbol{\nu}_i'$ is the noise. $\overline{\mathbf{b}}_i^c$ is the normalized light-of-sight

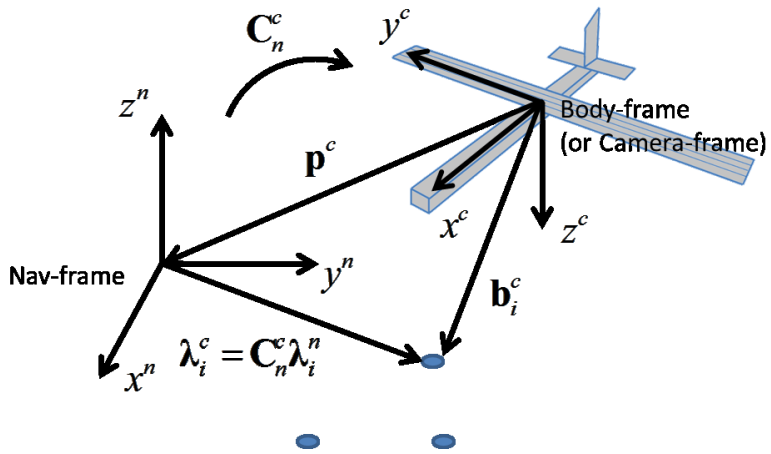

Figure 4.2: The Landmark Geometry Used by [1] and Direct Least Square (DLS) Approach

(LOS) vector ($\overline{\mathbf{b}}_i^c = \frac{\mathbf{b}_i^c}{\|\mathbf{b}_i^c\|}$) expressed in c-frame. Considering the noise-free case, the equations above can be written as

$$\alpha_i \overline{\mathbf{b}}_i^c = \mathbf{C}_n^c \boldsymbol{\lambda}_i^n + \mathbf{p}^c \tag{4.57}$$

The equations contains unknown quantities (α_i, $\mathbf{C}_n^c$, $\mathbf{p}^c$). By defining α_i and $\mathbf{p}^c$ as a vector, this equation can be written as:

$$\underbrace{\begin{bmatrix} \overline{\mathbf{b}}_1^c & & & -\mathbf{I} \\ & \ddots & & -\mathbf{I} \\ & & \overline{\mathbf{b}}_N^c & -\mathbf{I} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \\ \mathbf{p}^c \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} \mathbf{C}_n^c & & \\ & \ddots & \\ & & \mathbf{C}_n^c \end{bmatrix}}_{\mathbf{W}} \underbrace{\begin{bmatrix} \boldsymbol{\lambda}_1^n \\ \vdots \\ \boldsymbol{\lambda}_N^n \end{bmatrix}}_{\boldsymbol{\Lambda}} \tag{4.58}$$

where $\mathbf{A}$ and $\boldsymbol{\Lambda}$ are given or measured quantities. The vector and matrix $\mathbf{x}$ and $\mathbf{W}$ are the unknown quantities that must be determined. The vector $\mathbf{x}$ can be expressed as

weighted least square solution:

$$\begin{aligned}\mathbf{x} &= (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{W}\boldsymbol{\Lambda} \\ &= \begin{bmatrix} \mathbf{U} \\ \mathbf{V} \end{bmatrix} \mathbf{W}\boldsymbol{\Lambda}\end{aligned} \tag{4.59}$$

In [1] it is proved that $(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}$ can be partitioned into $\mathbf{U}$ and $\mathbf{V}$ such that α_i and $\mathbf{p}^c$ are function of $\mathbf{U}$ and $\mathbf{V}$, respectively. Furthermore, α_i and $\mathbf{p}^c$ can be written as

$$\alpha_i = \mathbf{u}_i^{\mathrm{T}}\mathbf{W}\boldsymbol{\Lambda} \qquad i = 1, 2, \ldots, N \tag{4.60}$$

$$\mathbf{p}^c = \mathbf{V}\mathbf{W}\boldsymbol{\Lambda} \tag{4.61}$$

where $\mathbf{u}_i^{\mathrm{T}}$ corresponds to the i-th row of the matrix $\mathbf{U}$. If we substitute Equation 4.60 and 4.61 back into Equation 4.57 and rewrite it:

$$\underbrace{\mathbf{u}_i^{\mathrm{T}}\mathbf{W}\boldsymbol{\Lambda}}_{\alpha_i}\overline{\mathbf{b}}_i^c = \mathbf{C}_n^c\boldsymbol{\lambda}_i^n + \underbrace{\mathbf{V}\mathbf{W}\boldsymbol{\Lambda}}_{\mathbf{p}^c} \tag{4.62}$$

The attitude matrix $\mathbf{C}_n^c$ can be parameterized by Cayley-Gibbs-Rodriguez (CGR) parameterizations $\mathbf{s} = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}^{\mathrm{T}}$. The rotational matrix is expressed using CGR parameter:

$$\mathbf{C}_n^c = \frac{\overline{\mathbf{C}}_n^c}{1 + \mathbf{s}^{\mathrm{T}}\mathbf{s}} \tag{4.63a}$$

$$\overline{\mathbf{C}}_n^c \triangleq ((1 - \mathbf{s}^{\mathrm{T}}\mathbf{s})\mathbf{I}_3 + 2[\mathbf{s}]^{\times} + 2\mathbf{s}\mathbf{s}^{\mathrm{T}} \tag{4.63b}$$

where $[\mathbf{s}]^{\times}$ is the skew-symmetric matrix parameterized by $\mathbf{s}$. $\mathbf{I}_3$ is the 3×3 identity matrix. This can be used to rewrite Equation 4.62 as

$$\mathbf{u}_i^{\mathrm{T}}\mathbf{W}(\mathbf{C}_{\mathbf{n}}^{\mathbf{c}}(\mathbf{s}))\boldsymbol{\Lambda}\overline{\mathbf{b}}_i^c = \mathbf{C}_n^c\boldsymbol{\lambda}_i^n + \mathbf{V}\mathbf{W}(\mathbf{C}_n^c(\mathbf{s}))\boldsymbol{\Lambda} \tag{4.64}$$

Since $\mathbf{C}_n^c$ is linearly appeared in all terms of the equation above, the denominator $1+\mathbf{s}^{\mathrm{T}}\mathbf{s}$ in $\mathbf{C}_n^c$ can be dropped leading to a further simplification

$$\mathbf{u}_i^{\mathrm{T}}\mathbf{W}(\overline{\mathbf{C}}_{\mathbf{n}}^{\mathbf{c}}(\mathbf{s}))\boldsymbol{\Lambda}\overline{\mathbf{b}}_i^c = \overline{\mathbf{C}}_n^c\boldsymbol{\lambda}_i^n + \mathbf{V}\mathbf{W}(\overline{\mathbf{C}}_n^c(\mathbf{s}))\boldsymbol{\Lambda} \tag{4.65}$$

The equation above is quadratic in $\mathbf{s}$. The equation discussed above is expressed in terms of noise free line of sight vector $\overline{\mathbf{b}}_i^c$. In practice, the measured vector $(\mathbf{y}'_i)$ is usually corrupted by noise as described in Equation 4.56a. Thus, the equation is modified as

$$\mathbf{u}_i^{\mathrm{T}}\mathbf{W}(\overline{\mathbf{C}}_{\mathbf{n}}^{\mathbf{c}}(\mathbf{s}))\mathbf{\Lambda}(\mathbf{y}'_i - \boldsymbol{\nu}'_i) = \overline{\mathbf{C}}_n^c\boldsymbol{\lambda}_i^n + \mathbf{V}\mathbf{W}(\overline{\mathbf{C}}_n^c(\mathbf{s}))\mathbf{\Lambda} \tag{4.66}$$

By defining $\underline{\boldsymbol{\nu}}'_i \triangleq \mathbf{u}_i^{\mathrm{T}}\mathbf{W}(\overline{\mathbf{C}}_{\mathbf{n}}^{\mathbf{c}}(\mathbf{s}))\boldsymbol{\nu}'_i$ we can rewrite the equation as:

$$\underline{\boldsymbol{\nu}}'_i = \mathbf{u}_i^{\mathrm{T}}\mathbf{W}(\overline{\mathbf{C}}_{\mathbf{n}}^{\mathbf{c}}(\mathbf{s}))\mathbf{\Lambda}\mathbf{y}'_i - \overline{\mathbf{C}}_n^c\boldsymbol{\lambda}_i^n - \mathbf{V}\mathbf{W}(\overline{\mathbf{C}}_n^c(\mathbf{s}))\mathbf{\Lambda} \tag{4.67}$$

The original PnP problem is the minimization problem as described in Equation 4.1. This problem is modified into another minimization problem

$$\{s_1,\ s_2,\ s_3\} = \text{argmin}\ J' \tag{4.68}$$

where the cost function J' is the squared sum of $\underline{\boldsymbol{\nu}}'_i$

$$J' = \sum_{i=1}^{N} \underline{\boldsymbol{\nu}}_i'^{\mathrm{T}}\underline{\boldsymbol{\nu}}'_i \tag{4.69}$$

the cost function J' is a fourth order polynomial parameterized by the s_1, s_2 and s_3. The minimum of the cost function can be solved for using the optimality conditions:

$$\bigtriangledown_{s_i} J' = 0, \quad i = 1,\ 2,\ 3 \tag{4.70}$$

where the $\bigtriangledown_{s_i}$ are partial derivative of J' with respect to s_i. In [1] Macaulay matrix is used to determine the roots of the polynomial given in Equation 4.70. Their goal is to calculate the multiplication matrix that they can directly solve for the system equations via eigen decomposition. The multiplication matrix is obtained by constructing a Macaulay resultant matrix. Readers can refer to [1] for detail information on this solution. In [1] it is noted that Equation 4.70 has total 27 solutions and their proposed approach can obtain all solutions analytically. However, some of the solutions might

be real or imaginary numbers correspond to minima, maxima or saddle points which should be excluded. In practice, only four real local minimum which are "in front of" the camera have been observed [1]. Only one of them can be the global minimum where can be easily determined by computing the cost function at each candidate solution. The one with smallest cost function will be regarded as the global minimum solution. The solution of Equation 4.70 yields s_1, s_2 and s_3 which are the estimates of the attitude parameter. They can be used to construct the rotational matrix (Equation 4.63a and 4.63b) or further derive the Euler angles. Finally, position estimate $\mathbf{p}^c$ can be obtained by Equation 4.61.

4.1.4 Performance Evaluation on DLT, EPnP and DLS Approaches

This section will compare the performance of three PnP algorithms discussed earlier in this chapter. They will be evaluated for the solution sensitivities against different pixel noise magnitudes with different landmarks in sight. We will simulate three different scenarios with 6, 12 and 24 landmarks observable. In addition, the IEKF based method discussed in chapter 3 will be included for comparison. As will be seen, when it converges, the IEKF based solution has comparable or better accuracy than the PnP methods. However, it might not converge to the correct solution as shown in Figure 4.3. Among the PnP methods it will be seen the DLT is the least accurate.

We will also inject three different level of noise to the pixel measurement where the pixel noises are assumed to be zero mean Gaussian random variables. The variance of injected noise are 10%, 5% and 1% of the camera's resolution. The camera is assumed to be placed at $\mathbf{p}^n = \begin{bmatrix} 33 & 12 & 14 \end{bmatrix}^{\mathrm{T}}$ in the n-frame with attitude (Euler angles in degrees) $\boldsymbol{\alpha} = \begin{bmatrix} 10 & 25 & 35 \end{bmatrix}^{\mathrm{T}}$ relative to n-frame. The camera's is assumed to be

calibrated with an intrinsic matrix:

$$\mathbf{M}_{Sim} = \begin{bmatrix} 2136.9 & 0 & 475.1 \\ 0 & 2133.2 & 560.3 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.71}$$

We plot a scatter plot of position errors of x and y after conducting 500 of Monte-Carlo simulation. The simulated landmarks position is tabulated in Table 4.1. Figure 4.3 shows the performance of PnP algorithms with 6 feature points observable. The DLT algorithm is very sensitive to large pixel noise. For smaller noise, its solution accuracy is improved but it is still not as good as the other two. EPnP ranks the second where the DLS has the best performance among all three.

Table 4.1: Location of 6 Simulated Landmarks

	Landmark					
	#1	#2	#3	#4	#5	#6
x	36.508	36.005	33.837	36.082	38.254	38.309
y	12.188	14.727	12.872	12.765	13.866	14.544
z	19.065	21.915	18.526	18.642	27.124	22.059

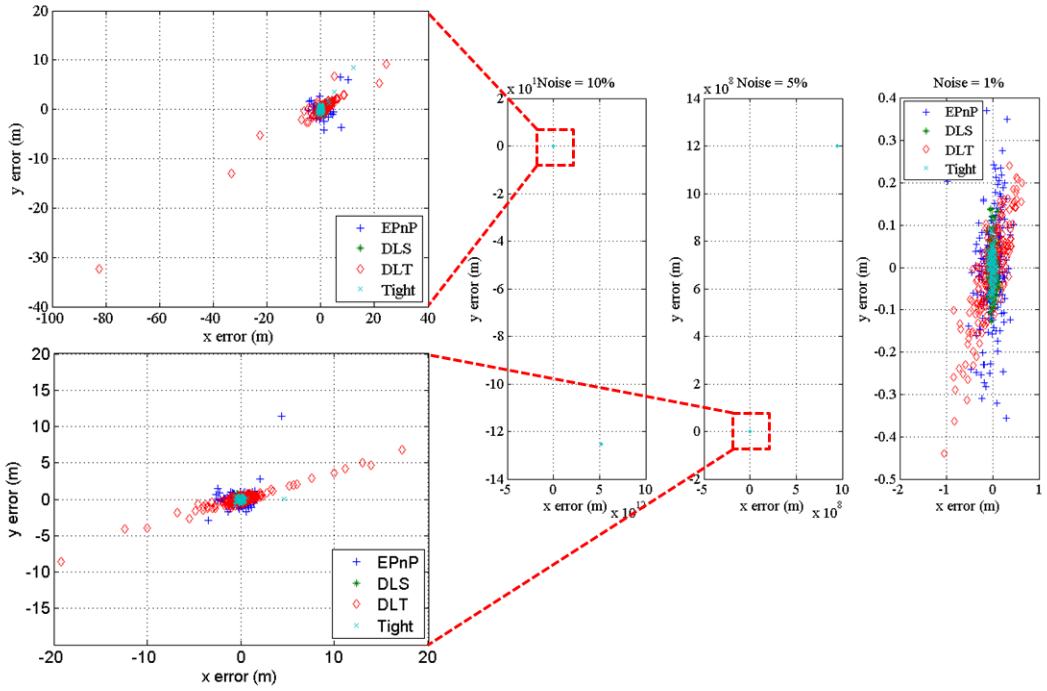

Figure 4.3: Comparison of PnP and Tight Algorithms (6 Landmarks)

Figure 4.4 shows the case with 12 landmarks observed. The location of the 12 landmarks are tabulated in Table 4.2. The performance of DLT is significantly improved in large noise case but still the other two algorithms can generate more accurate solutions than the DLT. For the case of smaller pixel noise, with more redundant measurement, the DLT can produce a comparable solution to EPnP and DLS.

Table 4.2: Location of 12 Simulated Landmarks

	Landmark					
	#1	#2	#3	#4	#5	#6
x	34.780	36.029	37.329	38.545	37.010	41.424
y	13.780	14.829	13.542	10.047	11.770	16.995
z	20.703	22.863	20.796	26.119	19.285	31.051
	#7	#8	#9	#10	#11	#12
x	36.752	36.573	37.749	35.540	35.272	35.964
y	10.608	12.106	12.387	11.614	11.806	14.389
z	21.118	19.031	24.282	17.531	21.296	26.136

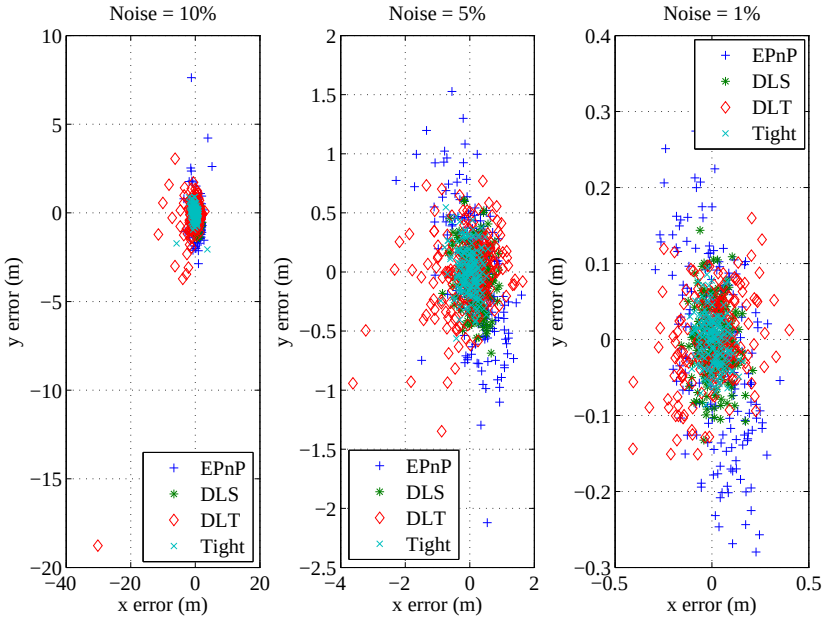

Figure 4.4: Comparison of PnP and Tight Algorithms (12 Landmarks)

We further increase the number of observable landmarks to 24 and exam the performance of these two algorithms . The simulated landmarks are tabulated in Table 4.3. The result is given in Figure 4.5. With higher number of redundancy, all three approaches provide more accurate estimates.

Table 4.3: Location of 24 Simulated Landmarks

	Landmark							
	#1	#2	#3	#4	#5	#6	#7	#8
x	39.151	45.222	39.222	37.660	37.779	39.943	35.363	34.500
y	11.745	11.791	10.870	14.044	12.531	17.848	12.195	13.512
z	21.557	28.150	33.098	26.363	27.332	32.256	18.783	22.333
	#9	#10	#11	#12	#13	#14	#15	#16
x	34.448	35.688	42.190	34.293	36.340	35.724	35.353	35.311
y	12.799	14.509	18.210	12.991	13.808	11.434	13.274	12.968
z	18.832	21.139	29.944	19.349	29.052	19.683	20.733	22.492
	#17	#18	#19	#20	#21	#22	#23	#24
x	35.025	36.659	40.632	33.738	33.959	34.713	36.258	33.937
y	11.901	12.176	10.407	12.266	12.145	11.549	12.242	13.000
z	16.386	19.237	25.241	15.897	16.369	17.380	19.090	19.004

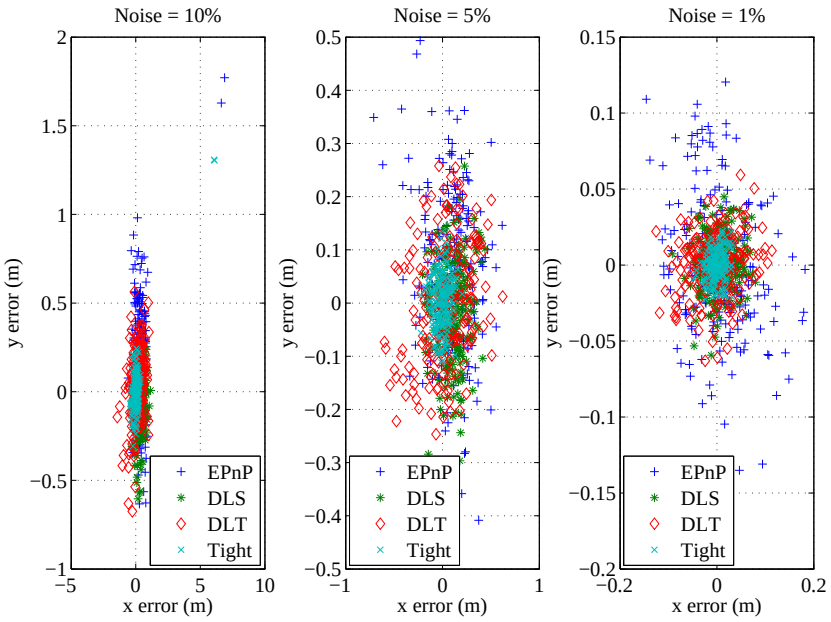

Figure 4.5: Comparison of PnP and Tight Algorithms (24 Landmarks)

From the simulation results, DLS outperforms EPnP and DLT in terms of accuracy. DLT provides a relatively less accurate estimate compared with the other two. This is

because, unlike DLS and EPnP, DLT formulates the PnP problem as an unconstrained optimization problem. Without imposing constraints on the possible solutions, the result can be greatly affected by measurement noise or outliers. On the other hand, constrained optimization (i.e. DLS) also has its own issue. The hard constraints imposed by the algorithm can prevent the existence of a solution. This is especially true if the measurement noise is too large or the there are many measurement outliers. For DLS approach, the robustness of generating a solution is not guaranteed. EPnP is an intermediate approach between constrained and unconstrained methods. As the result, the accuracy of EPnP is in the middle among the three and it does not have the robustness issue as DLS.

All three algorithms can provide position and attitude estimates based on a set of pre-determined landmarks and their pixel projection. In theory, they can be applied in the loosely coupled INS/VISNAV integration architecture. However, the loose architecture requires that the PnP algorithms provide a covariance estimate of the calculated solution. This is necessary since the position and attitude estimates generated by the PnP algorithm are measurements which will be fused with the INS. Unlike many cases that the measurement (or sensor) noise has fixed distribution which can be characterized off-line, the covariance of PnP estimates is related to the landmark geometry which changes constantly. Even if we characterize the distribution of pixel noise by laboratory experiments, the effect of pixel noise on position and attitude estimates is uncharacterized before the landmark geometry is known. Therefore, we need an algorithm which can provide a means for mapping the pixel error distribution to the solution (i.e. position and attitude error) distribution given the landmark geometry.

For EPnP and DLS methods, it is difficult to derive an analytical approach since they involve some mode selection operations. For example, EPnP determines its optimal solution by comparing the cost function of four cases $n = 1, \ldots, 4$. The optimal solution

is picked as the case with smallest cost. The four cases of $n = 1 \ldots 4$ in EPnP algorithms can be regarded as using four different models to fit the noisy measurement and the model which has the minimum residual is considered as the optimal choice in the least square sense. Then the solution derived from this optimal model will be used as the reported solution. Depending on the magnitude and distribution of the measurement noise, the reported solution can switch between these four models.

This switching effect can reduce the solution error but it will also alter the distribution of the reported solution. To illustrate the concept of mode selection of EPnP algorithms, a simplified one dimensional example is used to explain the idea. Assuming the noiseless measurement model has hemispheric form of a circle so measurement y is related to state x by $y = y_0 + \sqrt{1 - (x - x_0)^2}$ (we only consider the positive hemisphere here), where (x_0,y_0) is the center of the hemisphere. In this simulation, we define four hemisphere at four different locations (shown in Figure 4.6) as four different measurement models. Among the models, we assume model 2 is the correct representation of the physical measurement model and the true state $\overline{x} = 0.5$. These four models are created to analog to the four models (n = 1 (perspective) to 4 (orthographic)) in EPnP. Since the camera projection is perspective physically, $n = 1$ is the correct mathematical model to describe the projection. In this simplified example, we assume model 2 has the same role as $n = 1$ for EPnP.

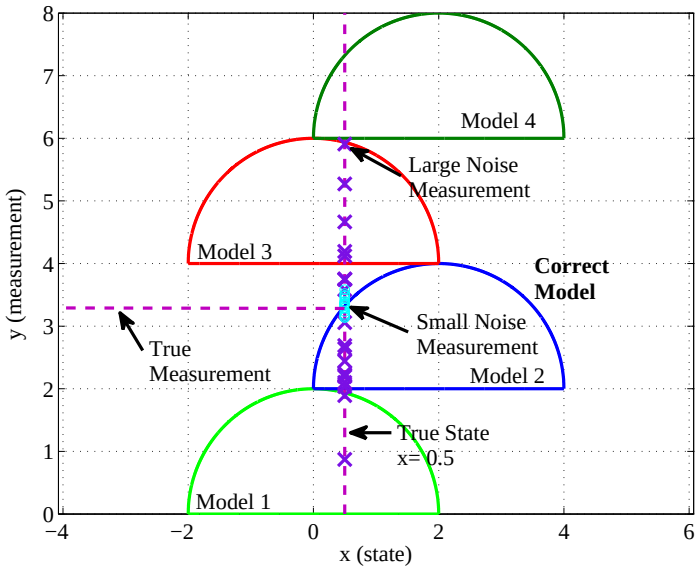

Figure 4.6: The Concept of Model Selection of State Estimate

For measurement corrupted by smaller noise (shown as cyan "o" in the figure), model 2 is still considered the "closest" representation of the measurement model by comparing the minimum residual with the other models. To visualize the idea, we run 2000 Monte Carlo simulations and plot the residual of each model. In each simulation, 20 measurements are generated and they are corrupted by zero mean Gaussian noise with standard deviation ($\sigma_y = 0.5$). The model with minimum residual will be regarded as the "closest" to the correct measurement model and its estimate will be reported as optimal state estimate. The upper plot of Figure 4.7 shows the model 2 has the minimum residual among all the models. Its estimate will be reported as optimal estimate as shown in the lower plot of Figure 4.7. The analytical state covariance estimation by linearizing the measurement model is a valid approximation due to the small measurement noise and no model switching occurred. The estimated covariance is $\hat{\sigma}_x = 0.0097$ where the sample

covariance of 2000 Monte Carlo simulations is $\breve{\sigma}_x = 0.0104$.

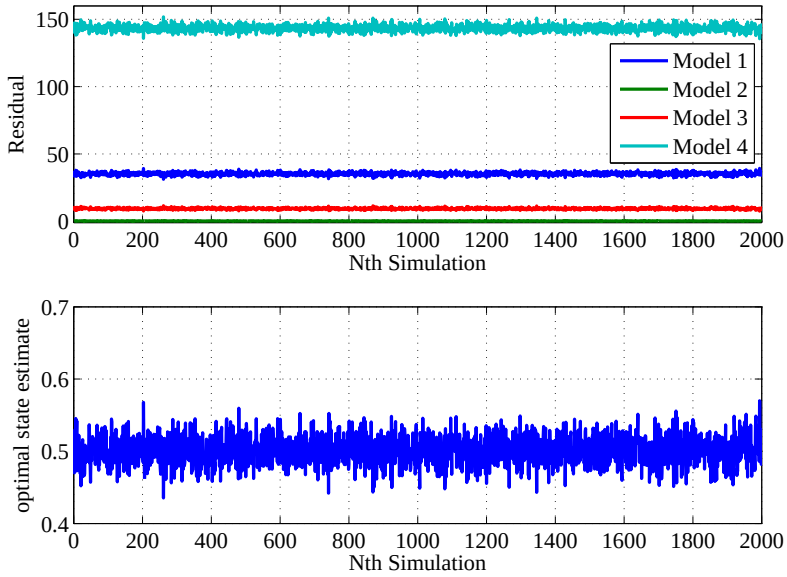

Figure 4.7: The Residual of the Models Corrupted by Gaussian Noise ($\sigma_y = 0.5$)

For large measurement noise (shown as purple "x" in Figure 4.6), we redo the Monte Carlo simulation with noise level increase to $\sigma_y = 2.5$. The residual of the four models and the optimal estimate is given in Figure 4.8. It can be seen from the figure, model 3 generates smaller residual than model 2 in some cases. Therefore, the algorithm will select the estimate from model 3 as reported solution. The sample covariance of 2000 Monte Carlo simulation is $\breve{\sigma}_x = 0.3842$. We also calculate the analytical covariance estimate based on the linearization approach at true state or the estimated state for particular simulation (N = 451, the state estimate $\hat{x} = 1.99$), the calculated covariance is tabulated in Table 4.4.

The analytical covariance estimates cannot be used to represent the sample covariance is due to large measurement noise which result model switching between different

Table 4.4: Calculated State Covariance at True State and Estimated State

at True State ($x = 0.5$)	at Estimated State ($x = 1.99$)
$\sigma_{x_1} = 4.6875$	$\sigma_{x_1} = 0.0034$
$\sigma_{x_2} = 0.2431$	$\sigma_{x_2} = 10716$
$\sigma_{x_3} = 4.6875$	$\sigma_{x_3} = 0.0034$
$\sigma_{x_4} = 0.2431$	$\sigma_{x_4} = 10716$

models. This model switching will affect the distribution of the state estimate.

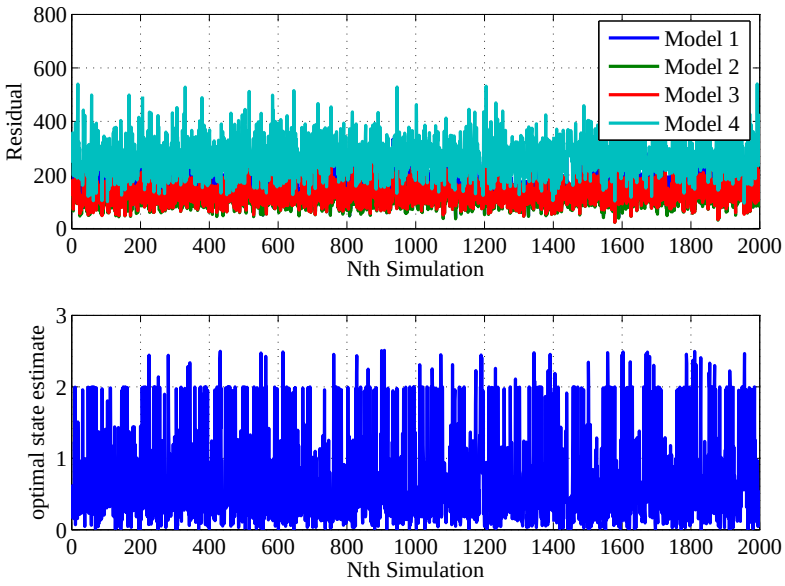

Figure 4.8: The Residual of the Models Corrupted by Gaussian Noise ($\sigma_y = 2.5$)

In the next section, we propose an analytical approach for estimating the measurement covariance based on DLT approach. The performance of DLT is not as good as the EPnP and DLS approaches in noisy environment which can greatly limit its application. Its performance in the case of rich feature points and small pixel noise is still acceptable. The existence of analytical covariance estimate is a plus for it to be used in loosely coupled integration.

4.1.5 DLT Based Measurement Covariance Analysis

Once the attitude and position solution have been calculated, the covariance estimation proceeds in three major steps. First, the normalized pixel and landmark survey errors are related to the covariance of $\hat{\overline{\boldsymbol{\eta}}}'$, where $\hat{\overline{\boldsymbol{\eta}}}'$ is the smallest singular value of $\tilde{\mathbf{B}}'$. This covariance is called $\mathbf{R}_{\hat{\overline{\eta}}'}$. Next α and $\mathbf{R}_{\hat{\overline{\eta}}'}$ are used to calculate the covariance of the elements of the $\mathbf{H}'$ matrix or $\mathbf{R}_{\hat{\eta}'}$. $\mathbf{T}_{\hat{\eta}}$ is used to denormalize $\mathbf{R}_{\hat{\eta}'}$, producing $\mathbf{R}_{\hat{\eta}}$. Finally, $\mathbf{R}_{\hat{\eta}}$ is used to calculate the attitude covariance $\mathbf{R}_a$ and the position covariance $\mathbf{R}_p$.

Calculating $\mathbf{R}_{\overline{\eta}}$ and $\mathbf{R}_{\eta}$

In the derivation that follows, the use of an *apostrophe* denotes a quantity that is using normalized pixel and landmark coordinates, and their associated, normalized noise statistics. The covariance matrix $\mathbf{R}_{\overline{\eta}'}$ is defined as:

$$\mathbf{R}_{\overline{\eta}'} = \mathcal{E}\left\{\delta\overline{\boldsymbol{\eta}}'\,\delta\overline{\boldsymbol{\eta}}'^{T}\right\} \tag{4.72}$$

where $\delta\overline{\boldsymbol{\eta}}' = \hat{\overline{\boldsymbol{\eta}}}' - \overline{\boldsymbol{\eta}}'$. This can be rearranged as $\hat{\overline{\boldsymbol{\eta}}}' = \overline{\boldsymbol{\eta}}' + \delta\overline{\boldsymbol{\eta}}'$. Recalling that $\hat{\overline{\boldsymbol{\eta}}}'$ is the unscaled solution to Equation 4.7 leads to the relationships:

$$\tilde{\mathbf{B}}'\hat{\overline{\boldsymbol{\eta}}}' = \mathbf{0} \tag{4.73}$$

$$\tilde{\mathbf{B}}'\hat{\overline{\boldsymbol{\eta}}}' = \left(\mathbf{B}' + \Delta\mathbf{B}'\right)\left(\overline{\boldsymbol{\eta}}' + \delta\overline{\boldsymbol{\eta}}'\right) \tag{4.74}$$

By dropping the higher order terms, and using the identity $\mathbf{B}'\overline{\boldsymbol{\eta}}' = \mathbf{0}$, these equations yield:

$$-\mathbf{B}'\delta\overline{\boldsymbol{\eta}}' = \Delta\mathbf{B}'\overline{\boldsymbol{\eta}}' \tag{4.75}$$

Because rank($\mathbf{B}'$) is 11, the Eckart-Young theorem [59] can be used to approximate B as:

$$\mathbf{B}' \approx \mathbf{U}_s\boldsymbol{\Sigma}_s\mathbf{V}_s^T \tag{4.76}$$

where $\mathbf{U}_s\mathbf{\Sigma}_s\mathbf{V}_s^T$ is the rank deficient matrix formed by retaining the first eleven singular values in $\tilde{\mathbf{\Sigma}}'$ and the first eleven columns of $\tilde{\mathbf{U}}'$ and $\tilde{\mathbf{V}}'$. Substituting this into Equation 4.75 and solving for $\delta\overline{\boldsymbol{\eta}}'$ gives:

$$\delta\overline{\boldsymbol{\eta}}' = \left(-\mathbf{V}_s\mathbf{\Sigma}_s^{-1}\mathbf{U}_s^T\right)\left(\Delta\mathbf{B}'\right)\overline{\boldsymbol{\eta}}' \tag{4.77}$$

$\Delta\mathbf{B}'$ can therefore be related directly to $\sigma_{u_i'}$, $\sigma_{v_i'}$ and $\sigma_{u_i'v_i'}$ by stacking $\Delta\mathbf{B}_i'$ for $i = 1, 2, \ldots, N$ where

$$\Delta\mathbf{B}_i' = \begin{bmatrix} \mathbf{Z}_{1\times4} & \mathbf{Z}_{1\times4} & \boldsymbol{\lambda}_i'^T\delta v_i' \\ \mathbf{Z}_{1\times4} & \mathbf{Z}_{1\times4} & -\boldsymbol{\lambda}_i'^T\delta u_i' \end{bmatrix} \tag{4.78}$$

Now Equation 4.77 is rearranged by converting $\Delta\mathbf{B}'$ into a vector $\delta\mathbf{b}'$

$$\delta\overline{\boldsymbol{\eta}}' = \mathbf{V}_s\mathbf{\Sigma}_s^{-1}\mathbf{U}_s^T\overline{\mathbf{V}}\,\delta\mathbf{b}' \tag{4.79}$$

where $\overline{\mathbf{V}} \in \mathbb{R}^{2N\times(2N\times12)}$ is $\tilde{\mathbf{v}}_{12}'$ written in matrix form as follows:

$$\overline{\mathbf{V}} = \begin{bmatrix} \tilde{\mathbf{v}}_{12}'^T & & & \\ & \tilde{\mathbf{v}}_{12}'^T & & \\ & & \ddots & \\ & & & \tilde{\mathbf{v}}_{12}'^T \end{bmatrix} \tag{4.80}$$

and $\delta\mathbf{b}' \in \mathbb{R}^{(2N\times12)\times1}$ is the vectorized form of $\Delta\mathbf{B}'$

$$\delta\mathbf{b}' = \begin{bmatrix} \mathbf{Z}_{4\times1} \\ \mathbf{Z}_{4\times1} \\ \boldsymbol{\lambda}_1'\delta v_1' \\ \mathbf{Z}_{4\times1} \\ \mathbf{Z}_{4\times1} \\ -\boldsymbol{\lambda}_1'\delta u_1' \\ \vdots \end{bmatrix} \tag{4.81}$$

The above equations can now be used to write:

$$\begin{aligned}\mathbf{R}_{\overline{\boldsymbol{\eta}}'} &= \mathbf{V}_s \boldsymbol{\Sigma}_s^{-1} \mathbf{U}_s^T \overline{\mathbf{V}} \mathbf{R}_{\delta \mathbf{b}'} \overline{\mathbf{V}}^T \mathbf{U}_s \boldsymbol{\Sigma}_s^{-T} \mathbf{V}_s^T \\ &= \boldsymbol{\Gamma} \mathbf{R}_{\delta \mathbf{b}'} \boldsymbol{\Gamma}\end{aligned} \tag{4.82}$$

where $\mathbf{R}_{\delta \mathbf{b}'} = \mathrm{E}\{\delta \mathbf{b}' \, \delta \mathbf{b}'^T\}$. Note that $\mathbf{R}_{\overline{\boldsymbol{\eta}}'}$ could also be calculated as:

$$\mathbf{R}_{\overline{\boldsymbol{\eta}}'} = \mathbf{J}_{\mathbf{v}^T} \mathbf{R}_{\delta \mathbf{b}'} \mathbf{J}_{\mathbf{v}^T}^T \tag{4.83}$$

where $\mathbf{J}_{\mathbf{v}^T}$ is the Jacobian of $\mathbf{v}^T$, which can be calculated using the derivation in [60]. However, calculating $\mathbf{J}_{\mathbf{v}^T}$ is an $\mathbf{O}\left(N^3\right)$ operation, while the method presented in this work takes advantage of sparseness to be $\mathbf{O}\left(N\right)$, where N is the number of landmarks. The entries in $\mathbf{R}_{\delta \mathbf{b}'}$ can be calculated by taking the partial derivatives of $\delta \mathbf{b}'$ with respect to each u_i', and v_i'. Using Equation 2.20 leads to defining $\mathbf{R}_{\delta \mathbf{b}'}$ as a large, sparse, block-diagonal matrix where each diagonal entry is a partitioned 24×24 matrix $\mathbf{R}_{\delta \mathbf{b}_i'}$ of the following form:

$$\mathbf{R}_{\delta b_i} = \begin{bmatrix} & \mathbf{Z}_{8\times4} & & \mathbf{Z}_{8\times4} \\ & \sigma_{v_i}^2 \boldsymbol{\lambda}_i \boldsymbol{\lambda}_i^{\mathrm{T}} & & \sigma_{u_i}\sigma_{v_i} \boldsymbol{\lambda}_i \boldsymbol{\lambda}_i^{\mathrm{T}} \\ \mathbf{Z}_{24\times8} & & \mathbf{Z}_{24\times8} & \\ & \mathbf{Z}_{8\times4} & & \mathbf{Z}_{8\times4} \\ & \sigma_{u_i}\sigma_{v_i} \boldsymbol{\lambda}_i \boldsymbol{\lambda}_i^{\mathrm{T}} & & \sigma_{u_i}^2 \boldsymbol{\lambda}_i \boldsymbol{\lambda}_i^{\mathrm{T}} \end{bmatrix} \tag{4.84}$$

At this point, having calculated the normalized, unscaled covariance $\mathbf{R}_{\overline{\boldsymbol{\eta}}'}$, the covariance can be denormalized. Using Equation 4.26, the denormalized, unscaled covariance $\mathbf{R}_{\overline{\boldsymbol{\eta}}}$ is calculated:

$$\mathbf{R}_{\overline{\boldsymbol{\eta}}} = \mathbf{T}_\eta^{-1} \mathbf{R}_{\overline{\boldsymbol{\eta}}'} \left(\mathbf{T}_\eta^{-1}\right)^T \tag{4.85}$$

Now the unscaled covariance $\mathbf{R}_{\overline{\boldsymbol{\eta}}}$ can be used to calculate the scaled covariance $\mathbf{R}_{\boldsymbol{\eta}}$. To do this, it is noted that

$$\mathbf{R}_{\boldsymbol{\eta}} = \mathcal{E}\left\{\delta \boldsymbol{\eta} \, \delta \boldsymbol{\eta}^T\right\} \tag{4.86}$$

where $\delta\boldsymbol{\eta}$ is defined as:

$$\delta\boldsymbol{\eta} = \delta\left(\alpha\overline{\boldsymbol{\eta}}\right) = \overline{\boldsymbol{\eta}}\frac{\partial\alpha}{\partial\overline{\boldsymbol{\eta}}}\delta\overline{\boldsymbol{\eta}} + \frac{\partial\overline{\boldsymbol{\eta}}}{\partial\overline{\boldsymbol{\eta}}}\delta\overline{\boldsymbol{\eta}}\alpha = \left[\overline{\boldsymbol{\eta}}\frac{\partial\alpha}{\partial\overline{\boldsymbol{\eta}}} + \alpha\mathbf{I}_{1\times 2}\right]\delta\overline{\boldsymbol{\eta}} \tag{4.87}$$

An explicit value for $\frac{\partial\alpha}{\partial\overline{\boldsymbol{\eta}}}$ must be derived. Applying the chain rule to Equation 4.13 yields:

$$\frac{\partial\alpha}{\partial\overline{\boldsymbol{\eta}}} = \frac{-\left[\det\left(\hat{\overline{\mathbf{H}}}_1\right)\right]^{-\frac{4}{3}}}{3\left[\det\left(\mathbf{M}^{-1}\right)\right]^{\frac{1}{3}}}\frac{\partial}{\partial\overline{\boldsymbol{\eta}}}\left(\det\left(\hat{\overline{\mathbf{H}}}_1\right)\right) \tag{4.88}$$

where $\frac{\partial}{\partial\overline{\boldsymbol{\eta}}}\left(\det\left(\hat{\overline{\mathbf{H}}}_1\right)\right)$ is:

$$\frac{\partial}{\partial\overline{\boldsymbol{\eta}}}\left(\det\left(\hat{\overline{\mathbf{H}}}_1\right)\right) = \begin{bmatrix} d_{11} & -d_{12} & d_{13} & 0 & -d_{21} & d_{22} & -d_{23} & 0 & d_{31} & -d_{32} & d_{33} & 0 \end{bmatrix} \tag{4.89}$$

and $d_{i,j}$ is the unsigned cofactor of the i^{th}, j^{th} entry in $\hat{\overline{\mathbf{H}}}_1$. Substituting into Equation 4.87 gives

$$\begin{aligned}\delta\boldsymbol{\eta} &= \left[\overline{\boldsymbol{\eta}}\frac{-\left[\det\left(\hat{\overline{\mathbf{H}}}_1\right)\right]^{-\frac{4}{3}}}{3\left[\det\left(\mathbf{M}^{-1}\right)\right]^{\frac{1}{3}}}\begin{bmatrix} d_{11} & -d_{12} & d_{13} & 0 & \dots & d_{33} & 0 \end{bmatrix} + \alpha\mathbf{I}_{12}\right]\delta\overline{\boldsymbol{\eta}} \\ &= \mathbf{G}\delta\overline{\boldsymbol{\eta}}\end{aligned} \tag{4.90}$$

and, therefore,

$$\begin{aligned}\mathbf{R}_{\boldsymbol{\eta}} &= \mathbf{G}^T\mathbf{R}_{\overline{\boldsymbol{\eta}}}\mathbf{G} \\ &= \mathbf{G}^T\mathbf{T}_{\eta}^{-1}\boldsymbol{\Gamma}\mathbf{R}_{\delta\mathbf{b}'}\boldsymbol{\Gamma}^T\left(\mathbf{T}_{\eta}^{-1}\right)^T\mathbf{G}\end{aligned} \tag{4.91}$$

This completes the derivation of the expression for $\mathbf{P}_{\boldsymbol{\eta}}$.

Relating $\mathbf{R}_{\boldsymbol{\eta}}$ to Attitude Errors

Now the covariance for $\boldsymbol{\eta}$ must be related to the covariance of the attitude. Euler angles are used to represent the attitude state. This results in a three step process to relate

$\mathbf{R}_{\boldsymbol{\eta}}$ to Euler angle errors. First $\mathbf{R}_{\boldsymbol{\eta}}$ is used to derive the covariance for the elements of $\hat{\mathbf{C}}_n^b$ or $\mathbf{R}_{\hat{c}}$. Next, the covariance of the orthogonalized $\breve{\mathbf{C}}_n^b$ or $\mathbf{R}_{\breve{c}}$ is calculated. Then $\mathbf{R}_{\breve{c}}$ is used to determine the covariance for the attitude errors.

Relating $\mathbf{R}_{\boldsymbol{\eta}}$ to $\mathbf{R}_{\hat{c}}$ is straight forward. This is done by noting that:

$$\mathbf{H}_1 = \mathbf{M}\hat{\mathbf{C}}_n^b = \begin{bmatrix} \eta_1 & \eta_2 & \eta_3 \\ \eta_5 & \eta_6 & \eta_7 \\ \eta_9 & \eta_{10} & \eta_{11} \end{bmatrix} \tag{4.92}$$

$$\hat{\mathbf{C}}_n^b = \mathbf{M}^{-1}\mathbf{H}_1 \tag{4.93}$$

Next the above equation is "vectorized" by arranging the elements of $\hat{\mathbf{C}}_n^b$ into a 9×1 vector.

$$\mathbf{c} \equiv \begin{bmatrix} c_{11} \\ c_{12} \\ c_{13} \\ c_{21} \\ \vdots \\ c_{33} \end{bmatrix} = \mathbf{S}_1\boldsymbol{\eta} \tag{4.94}$$

The notation c_{ij} is used to indicate the element of $\hat{\mathbf{C}}_n^b$ found in the i^{th} row and j^{th} column. The matrix $\mathbf{S}_1 = \mathbf{S}_1(\mathbf{M})$ is a shaping matrix which depends on the camera calibration matrix $\mathbf{M}$. The subscript "1" is used because this is the first of two such shaping matrices that will be used. Now assuming $\mathbf{M}$ is known without error, one can write:

$$\delta\mathbf{c} = \mathbf{S}_1\delta\boldsymbol{\eta} \tag{4.95}$$

from which follows

$$\mathbf{R}_{\hat{c}} = \mathcal{E}\left\{\delta\mathbf{c}\,\delta\mathbf{c}^T\right\} \tag{4.96}$$

$$= \mathcal{E}\left\{\mathbf{S}_1\delta\boldsymbol{\eta}\,\delta\boldsymbol{\eta}^T\mathbf{S}_1^T\right\} \tag{4.97}$$

$$= \mathbf{S}_1\mathbf{R}_{\boldsymbol{\eta}}\mathbf{S}_1^T \tag{4.98}$$

Next the covariance of the orthogonal $\breve{\mathbf{C}}_n^b$ matrix must be calculated. From examining Equation 4.17, it is clear that the covariance $\mathbf{R}_{\breve{c}}$ is calculated as:

$$\begin{aligned}\mathbf{R}_{\breve{c}} &= \left[\frac{\partial\hat{\mathbf{U}}}{\partial\hat{\mathbf{C}}_n^b}\hat{\mathbf{V}}^T + \hat{\mathbf{U}}\frac{\partial\hat{\mathbf{V}}^T}{\partial\hat{\mathbf{C}}_n^b}\right]\mathbf{R}_{\hat{c}}\left[\frac{\partial\hat{\mathbf{U}}}{\partial\hat{\mathbf{C}}_n^b}\hat{\mathbf{V}}^T + \hat{\mathbf{U}}\frac{\partial\hat{\mathbf{V}}^T}{\partial\hat{\mathbf{C}}_n^b}\right]^T \\ &= \mathbf{J}_{\mathbf{uv}}\mathbf{R}_{\hat{c}}\mathbf{J}_{\mathbf{uv}}^T\end{aligned} \tag{4.99}$$

where $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}^T$ come from the SVD of $\hat{\mathbf{C}}_n^b$. An elegant method of calculating $\frac{\partial\hat{\mathbf{U}}}{\partial\hat{\mathbf{C}}_n^b}$ and $\frac{\partial\hat{\mathbf{V}}^T}{\partial\hat{\mathbf{C}}_n^b}$ is presented in [60].

With $\mathbf{R}_{\breve{c}}$ calculated, it must now be related to Euler angle errors. For the 3-2-1 Euler angle sequence $\mathbf{C}_n^b$ is given by:

$$\breve{\mathbf{C}}_n^b = \begin{bmatrix} c_{\breve{\theta}}c_{\breve{\psi}} & c_{\breve{\theta}}s_{\breve{\psi}} & -s_{\breve{\theta}} \\ s_{\breve{\phi}}s_{\breve{\theta}}c_{\breve{\psi}} - c_{\breve{\phi}}s_{\breve{\psi}} & s_{\breve{\phi}}s_{\breve{\theta}}s_{\breve{\psi}} - c_{\breve{\phi}}c_{\breve{\psi}} & s_{\breve{\phi}}c_{\breve{\theta}} \\ c_{\breve{\phi}}s_{\breve{\theta}}c_{\breve{\psi}} + s_{\breve{\phi}}s_{\breve{\psi}} & c_{\breve{\phi}}s_{\breve{\theta}}s_{\breve{\psi}} - s_{\breve{\phi}}c_{\breve{\psi}} & c_{\breve{\phi}}c_{\breve{\theta}} \end{bmatrix} \tag{4.100}$$

where $c_{(\bullet)}$ and $s_{(\bullet)}$ are used as short-hand for cos and sin of $(\bullet)$. From this it can be seen that the Euler angles are related to the elements of $\breve{\mathbf{C}}_n^b$ by the following non-linear relations:

$$\breve{\phi} = \tan^{-1}\left(\frac{c_{23}}{c_{33}}\right) \tag{4.101}$$

$$\breve{\theta} = -\sin^{-1}c_{13} \tag{4.102}$$

$$\breve{\psi} = \tan^{-1}\left(\frac{c_{12}}{c_{11}}\right) \tag{4.103}$$

A perturbation of these relations leads to:

$$\begin{aligned}
\delta\alpha_N \equiv \breve{\phi} - \phi &\approx \left(\frac{c_{33}}{c_{23}^2 + c_{33}^2}\right)\delta c_{23} - \left(\frac{c_{23}}{c_{23}^2 + c_{33}^2}\right)\delta c_{33} \\
\delta\alpha_E \equiv \breve{\theta} - \theta &\approx \left(-\frac{1}{\sqrt{1 - c_{13}^2}}\right)\delta c_{13} \\
\delta\alpha_N \equiv \breve{\psi} - \psi &\approx \left(\frac{c_{11}}{c_{12}^2 + c_{11}^2}\right)\delta c_{12} - \left(\frac{c_{12}}{c_{12}^2 + c_{11}^2}\right)\delta c_{11}
\end{aligned} \tag{4.104}$$

Note that $\delta\boldsymbol{\alpha}$ is not a standard vector in the sense that addition of attitude error vectors does not yield another attitude error vector. This is just a notational convenience. Using this definition of $\boldsymbol{\alpha}$ Equation 4.104 is written as

$$\delta\boldsymbol{\alpha} = \mathbf{S}_2\delta\mathbf{c} \tag{4.105}$$

where $\mathbf{S}_2 = \mathbf{S}_2\left(\mathbf{C}_n^b\right)$ is another shaping matrix. From this it follows that the Euler angle covariance matrix $\mathbf{R}_{\boldsymbol{\alpha}}$ is given by:

$$\begin{aligned}
\mathbf{R}_{\boldsymbol{\alpha}} &= \mathbf{S}_2\mathbf{R}_{\breve{\mathbf{c}}}\mathbf{S}_2^T \\
&= \mathbf{S}_2\mathbf{J}_{\mathbf{uv}}\mathbf{S}_1\mathbf{R}_{\boldsymbol{\eta}}\mathbf{S}_1^T\mathbf{J}_{\mathbf{uv}}^T\mathbf{S}_2^T \\
&= \mathbf{S}\mathbf{R}_{\boldsymbol{\eta}}\mathbf{S}^T
\end{aligned} \tag{4.106}$$

where the overall shaping matrix $\mathbf{S}$ is defined as $\mathbf{S} = \mathbf{S}_2\mathbf{J}_{\mathbf{uv}}\mathbf{S}_1$.

Relating $\mathbf{R}_{\boldsymbol{\eta}}$ to Position Errors

To relate $\mathbf{R}_{\boldsymbol{\eta}}$ to the position solution covariance $\mathbf{R}_{\mathbf{p}}$, define the following three auxiliary variables

$$\mathbf{E} \equiv -\left(\mathbf{M}\mathbf{C}_n^b\right)^{-1} = -\mathbf{H}_1^{-1} \tag{4.107}$$

$$\boldsymbol{\eta}_1 \equiv \begin{bmatrix} \eta_1 & \eta_2 & \eta_3 & \eta_5 & \eta_6 & \eta_7 & \eta_9 & \eta_{10} & \eta_{11} \end{bmatrix}^T \tag{4.108}$$

$$\boldsymbol{\eta}_2 \equiv \begin{bmatrix} \eta_4 & \eta_8 & \eta_{12} \end{bmatrix}^T \tag{4.109}$$

and then note that

$$\mathbf{p}^n = \mathbf{E}\boldsymbol{\eta}_2 = \mathbf{F}\mathbf{e} \tag{4.110}$$

The second equality in the equation above is just a rewrite of the first where the matrix $\mathbf{E}$ has been written as a vector $\mathbf{e}$ while $\boldsymbol{\eta}_2$ has been rearranged into a the matrix $\mathbf{F}$. A perturbation of the equation above leads to:

$$\delta\mathbf{p}^n = \mathbf{E}\delta\boldsymbol{\eta}_2 + \mathbf{F}\delta\mathbf{e} \tag{4.111}$$

Since by definition

$$\mathbf{R}_\mathbf{p} = \mathcal{E}\left\{\delta\mathbf{p}^n \ (\delta\mathbf{p}^n)^T\right\} \tag{4.112}$$

it follows that:

$$\begin{aligned}\mathbf{R}_\mathbf{p} = \mathbf{E}\mathcal{E}\left\{\delta\boldsymbol{\eta}_2 \ (\delta\boldsymbol{\eta}_2)^T\right\}\mathbf{E}^T + \mathbf{F}\mathcal{E}\left\{\delta\mathbf{e} \ (\delta\mathbf{e})^T\right\}\mathbf{F}^T \\ + \mathbf{E}\mathcal{E}\left\{\delta\boldsymbol{\eta}_2 \, \delta\mathbf{e}^T\right\}\mathbf{F}^T + \mathbf{F}\mathcal{E}\left\{\delta\mathbf{e} \ (\delta\boldsymbol{\eta}_2)^T\right\}\mathbf{E}^T\end{aligned} \tag{4.113}$$

The first term on the left is the easiest to deal with. The expected value in this term has already been computed and is just a 3×3 subset of $\mathbf{R}_{\boldsymbol{\eta}}$. The remaining terms can also be dealt with easily once it is noted that the perturbation in the second term of Equation 4.113 is a function of the elements of the $\boldsymbol{\eta}$ vector.Note that $\mathbf{e} = \mathbf{e}(\boldsymbol{\eta}_1)$. Thus, by the rules of perturbations:

$$\delta\mathbf{e} = \frac{\partial\mathbf{e}}{\partial\boldsymbol{\eta}_1}\delta\boldsymbol{\eta}_1 = \mathbf{J}_e\boldsymbol{\eta}_1 \tag{4.114}$$

The symbol $\mathbf{J}_e$ is the Jacobian of the vector $\mathbf{e}$ which can be used to write:

$$\mathbf{F}\mathcal{E}\left\{\delta\mathbf{e} \ (\delta\mathbf{e})^T\right\}\mathbf{F}^T = \mathbf{F}\mathbf{J}_e\mathcal{E}\left\{\delta\boldsymbol{\eta}_1 \ (\delta\boldsymbol{\eta}_1)^T\right\}\mathbf{J}_e^T\mathbf{F}^T \tag{4.115}$$

Once again, $\mathcal{E}\left\{\delta\boldsymbol{\eta}_1 \ (\delta\boldsymbol{\eta}_1)^T\right\}$ is a subset of the covariance matrix $\mathbf{R}_\eta$ that was computed earlier. Using a similar argument for each of the other remaining terms, Equation 4.113

can be rewritten as:

$$\mathbf{R_p} = \mathbf{E}\mathcal{E}\left\{\delta\boldsymbol{\eta}_2\ (\delta\boldsymbol{\eta}_2)^T\right\}\mathbf{E}^T + \mathbf{F}\mathbf{J}_e\mathcal{E}\left\{\delta\boldsymbol{\eta}_1\ (\delta\boldsymbol{\eta}_1)^T\right\}(\mathbf{J}_e\mathbf{F})^T + 2\mathcal{E}\left\{(\mathbf{E}\delta\boldsymbol{\eta}_2)\ (\mathbf{F}\mathbf{J}_e\delta\boldsymbol{\eta}_1)^T\right\} \tag{4.116}$$

where all matrices of the form $\mathcal{E}\{\bullet\}$ in the first two terms contain elements which were computed earlier in $\mathbf{R}_{\boldsymbol{\eta}}$. The last term contains elements which are linear combinations of the elements in $\mathbf{R}_{\boldsymbol{\eta}}$. This is reflected by the expectation operator being placed outside of the entire product of matrices with vectors $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$.

By giving the statistical characteristic of the pixel errors, the analysis derived above can quantify the impact on the position and attitude estimates computed by the DLT approach.

Validating the DLT Covariance Estimate

To verify the correctness of the analytical covariance estimate, the output of the covariance analysis are compared against Monte Carlo simulations. An example is generated to visualize the comparison. The simulation assumes 20 feature points in view and their geometry is arranged as shown in Figure 4.9. The pixel measurements are corrupted by Gaussian random noise with standard deviation $\sigma_u = \sigma_v = 3$ pixels. The result is shown in Figure 4.10.

This figure shows the distribution of y and z position error estimates. The green "+" are DLT position estimates corrupted by pixel noise with 20 pixel of the standard deviation. The blue dash ellipse is the result of DLT covariance estimate and the red ellipse is the sample covariance based on five hundred times Monte Carlo simulations. The comparison of estimated versus statistical covariances for position and attitude is tabulated in Table 4.5.

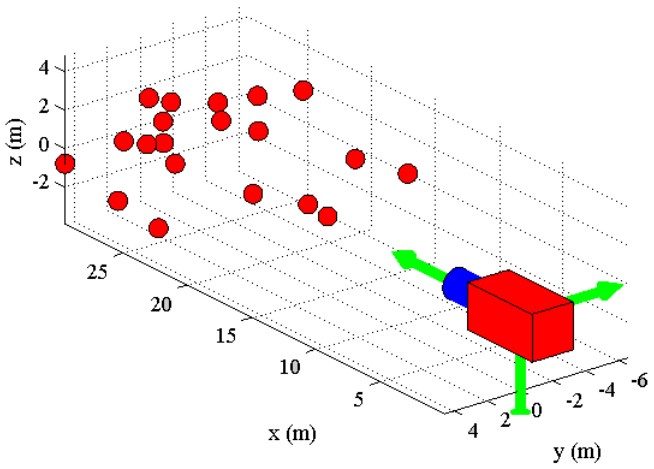

Figure 4.9: Simulation Scenario of DLT Covariance Estimate

4.2 Filter Formulation

The loosely integrated INS/VISNAV architecture is implemented using an EKF framework. Thus, the perturbation of the state vector expressed in equation 3.5 is also used for tight integration. Furthermore, the time update equation for tight integration is essentially the same for the loose integration. Therefore, we ignore the details of time update equation in this section since they were given in section 3.4.

4.2.1 Measurement Update Equation

The measurement update for loose integration is straightforward and linear since the state vector is directly measurable and provided by VISNAV algorithm (e.g. DLT). The measurement update equation is given as

$$\delta \mathbf{y}_l = \mathbf{H}_l \delta \mathbf{x} + \boldsymbol{\nu}_l \tag{4.117}$$

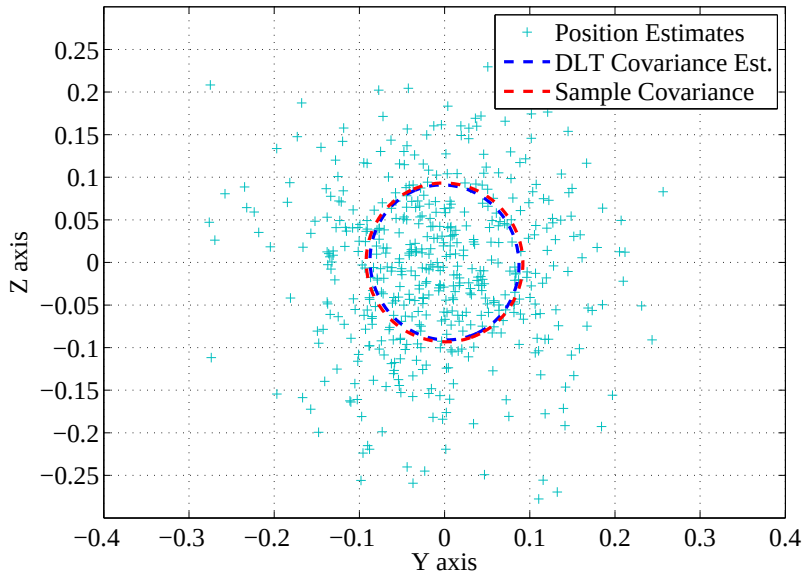

Figure 4.10: DLT Covariance Estimate v.s. Sample Covariance

where the subscript l indicates loose integration. $\delta\mathbf{y}_l = [(\delta\mathbf{p}^n)^{\mathrm{T}} \quad (\delta\boldsymbol{\alpha}^n)^{\mathrm{T}}]^{\mathrm{T}}$ and the measurement matrix $\mathbf{H}_l$ is given by

$$\mathbf{H}_l = \begin{bmatrix} \mathbf{I}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 \\ \mathbf{Z}_3 & \mathbf{Z}_3 & \mathbf{I}_3 & \mathbf{Z}_3 & \mathbf{Z}_3 \end{bmatrix} \tag{4.118}$$

The measurement noise $\boldsymbol{\nu}_l$ represents the position and attitude errors. The covariance matrix of $\boldsymbol{\nu}_l$ is defined as

$$\mathbf{R}_{\boldsymbol{\nu}_l} = \mathrm{E}\{\boldsymbol{\nu}_l \boldsymbol{\nu}_l^{\mathrm{T}}\} = \begin{bmatrix} \mathbf{R}_{\mathbf{p}} & \mathbf{R}_{\boldsymbol{\alpha}\mathbf{p}} \\ \mathbf{R}_{\boldsymbol{\alpha}\mathbf{p}} & \mathbf{R}_{\boldsymbol{\alpha}} \end{bmatrix} \tag{4.119}$$

where $\mathbf{R}_{\mathbf{p}}$ and $\mathbf{R}_{\boldsymbol{\alpha}}$ are the covariance of position and attitude measurement, respectively

Table 4.5: Comparison of Statistical and Estimated Covariance

Position Covariance (m^2)					
Sample Covariance			DLT Covariance		
0.3859	0.0034	0.0126	0.3844	0.0038	0.0124
0.0034	0.0078	-0.0002	0.0038	0.0076	-0.0003
0.0126	-0.0002	0.0083	0.0124	-0.0003	0.0083
Attitude Covariance (deg^2)					
Sample Covariance			DLT Covariance		
0.7343	-0.0104	0.0002	0.7438	-0.0098	0.0008
-0.0104	0.0211	0.0020	-0.0098	0.0207	0.0020
0.0002	0.0020	0.0017	0.0008	0.0020	0.0017

and $\mathbf{R}_{\mathbf{p}\boldsymbol{\alpha}}$ is their correlation. The computation of the position and attitude covariance and is given in the previous section. For the time being, we assume the cross correlation $\mathbf{R}_{\boldsymbol{\alpha}\mathbf{p}}$ is zero.

4.2.2 Incorporating Measurement Updates

Similar to the EKF formulation for tight architecture, the first step when the measurement update occurs is to calculate the Kalman gain at $t = t_k$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_l^{\mathrm{T}} (\mathbf{H}_l \mathbf{P}_k^- \mathbf{H}_t^{\mathrm{T}} + \mathbf{R}_{\boldsymbol{\nu}_l})^{-1} \tag{4.120}$$

where the subscript l indicates the loose measurement model and the associate measurement covariance are used in the calculation. The state vector covariance is then updated by

$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{H}_l \mathbf{P}_k^- \tag{4.121}$$

Then the state vector correction is determined by

$$\delta \mathbf{x}_k^+ = \mathbf{K}_k \delta \mathbf{y}_l \tag{4.122}$$

The updated estimate $\mathbf{x}_k^+$ is calculated as follow except the attitude states ($\boldsymbol{\alpha}^n$)

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \delta\mathbf{x}_k^+ \tag{4.123}$$

The attitude angles correction $(\delta\boldsymbol{\alpha}^n)^+$ in updated states are used to construct a rotational matrix defined as $\mathbf{C}_{n'}^n$. The updated rotational matrix is computed by

$$(\mathbf{C}_n^b)^+ = \mathbf{C}_{n'}^n(\mathbf{C}_n^b)^- \tag{4.124}$$

Finally, the three updated Euler angles (roll, pitch yaw) are calculated as the following

$$\phi^+ = \tan^{-1}\left(\frac{c_{23}^+}{c_{33}^+}\right) \tag{4.125a}$$

$$\theta^+ = \sin^{-1}\left(c_{13}^+\right) \tag{4.125b}$$

$$\psi^+ = \tan^{-1}\left(\frac{c_{12}^+}{c_{11}^+}\right) \tag{4.125c}$$

c_{ij}^+ is‘ the element of i^{th} row and j^{th} column in $(\mathbf{C}_n^b)^+$.

4.3 Filter Validation Through Simulation

The loose integration described above is validated using flight data from a simulated trajectory is depicted in Figure 4.11. The altitude of the trajectory varied from 5 meters to 30 meters above ground level. Red dots in Figure 4.11 are the position markers that show the UAV's location every 10 seconds and green dots represent pre-surveyed landmarks. The entire simulation run lasts for about 2 minutes. The loose integration result is compared against tight integration for evaluating the estimation accuracy and the robustness.

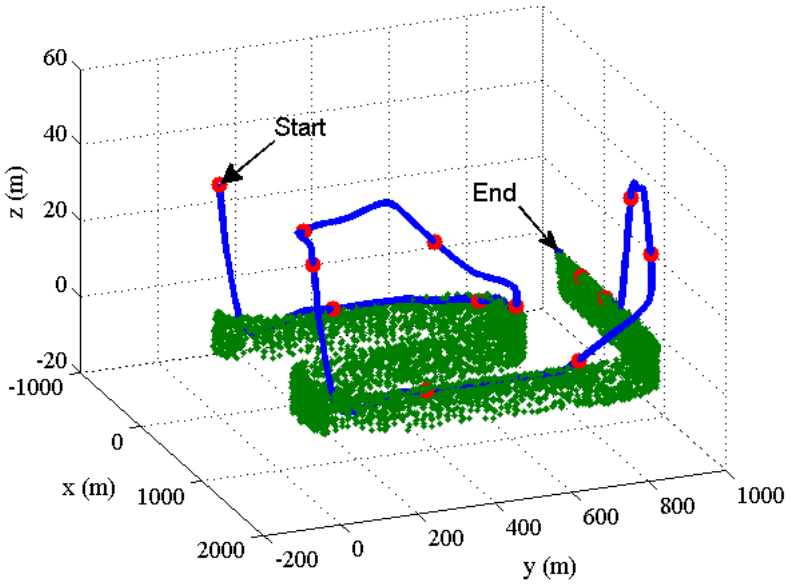

Figure 4.11: Simulated trajectory and feature points

4.3.1 Sensor Models

As described in section 2.1.4, the noise corrupted IMU measurement can be modeled as

$$\mathbf{f} = \mathbf{f}_{\text{true}} + \mathbf{b}_{0,a} + \mathbf{b}_{1,a} + \mathbf{w}_a \tag{4.126}$$

$$\boldsymbol{\omega} = \boldsymbol{\omega}_{\text{true}} + \mathbf{b}_{0,g} + \mathbf{b}_{1,g} + \mathbf{w}_g \tag{4.127}$$

where $\mathbf{f}$ and $\boldsymbol{\omega}$ are the noise corrupted accelerometer and gyro outputs, respectively. The noise terms include wide band $\mathbf{w}$, turn on bias $\mathbf{b}_0$ and the time varying bias $\mathbf{b}_1$. The time varying bias is modeled as a first-order Gauss Markov process

$$\dot{\mathbf{b}}_1 = -\frac{1}{\tau}\mathbf{b}_1 + \boldsymbol{\mu}_1 \tag{4.128}$$

where τ is the correlation time constant and $\mathbf{n}_1$ represents the driving Gaussian white noise. The statistics of the IMU output error used in the simulation are shown in Table 4.6.

The camera is modeled by the pin-hole camera model described in section 2.2.1. The parameters that describe the model are the horizontal and vertical focal length (f_x and f_y). The pixel measurement is assumed to be corrupted by white jitter noise that has standard deviation σ_u and σ_v in the horizontal and vertical direction respectively. The simulated values are shown in Table 4.6.

Table 4.6: IMU and Camera Parameters for Simulation

Sensor	Parameters	Value
Gyro	τ_g $\sigma_{\mu_{1,g}}$ σ_{w_g}	300 sec 0.1 deg 0.3 deg
Accel	τ_a $\sigma_{\mu_{1,a}}$ σ_{w_a}	100 sec $1.2 \times 10^{-3} g$ $9 \times 10^{-3} g$
Camera	Horizontal Resolution Vertical Resolution f_x f_y σ_u σ_v	4608 px 3456 px 3230 px 3235 px 6.36 px 6.36 px

4.3.2 Simulation Result

Figure 4.12 plots the position error obtained from running the loose integration filter at 2 Hz and 0.5 Hz respectively. Figure 4.13 shows the same plot for the tight integration running in the same simulation scenario. At 2 Hz, tight integration outperforms loose in terms of estimation accuracy due to its optimal formulation nature. However, if the measurement update rate is reduced to 0.5 Hz, the loosely integrated architecture still results in position error that is less than 3 m even though measurements come in at a slower rate, but the performance of the tight integration filter is significantly degraded.

From the figure, the tight position error is 100 times larger than the result of the loose integration filter. This shows the instability of the tight integration filter described in section 3.8 when the measurement update becomes less frequent. The concept of local minimum inclusion is illustrated in Figure 4.14. If the measurement update rate is lower (e.g. every two seconds as shown in the figure), the covariance (depicted as a uncertainty ellipse in the figure) of the state estimate solely calculated by INS will be larger than the covariance (shown as a gray ellipse in the figure) at higher measurement update rate. The probability of including a erroneous local minimum into the uncertainty ellipse will increase. As the result the chance of converging into the erroneous local minimum will also increases which will cause inaccurate estimate and further lead to instability. We also note that in both architectures the position error is larger in the time intervals 35 - 60 seconds and 85 - 105 seconds. These intervals correspond to the portions of the flight where the UAV was flying higher above the ground. In these maneuvers, the pixel measurement errors will translate into larger position error cause less accurate position estimate.

4.4 Filter Validation Through Experiment

The loosely coupled architecture is also validated through experiment. The experiment was conducted in the Interactive Guidance and Control Lab (IGCL) at Aerospace Engineering and Mechanics, University of Minnesota-Twin Cities. The lab is equipped with a Vicon motion tracking system which can be used to track the position and attitude of a object with centimeter and sub-degree accuracies (shown in Figure 4.15). This tracking system provide reference trajectory and attitude which will serve as true references. An AR.Drone with onboard camera and IMU (shown in Figure 4.16) is used to generate the 6-DOF trajectory.

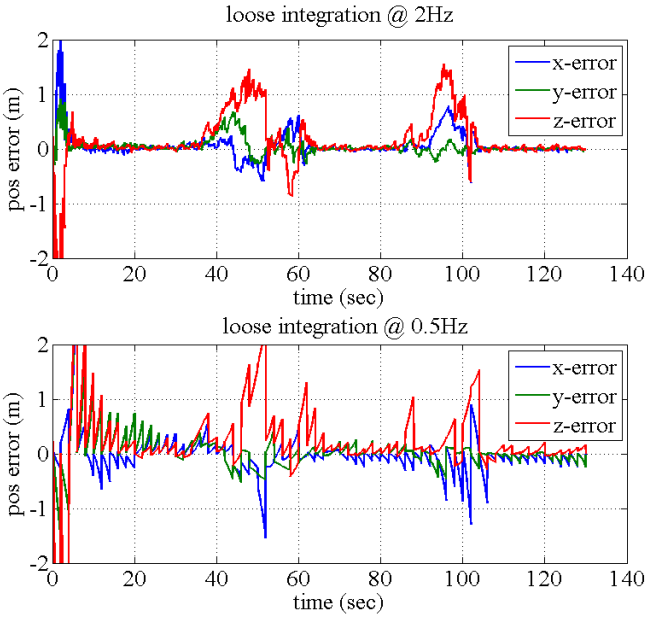

Figure 4.12: Position Error of Loose Integration

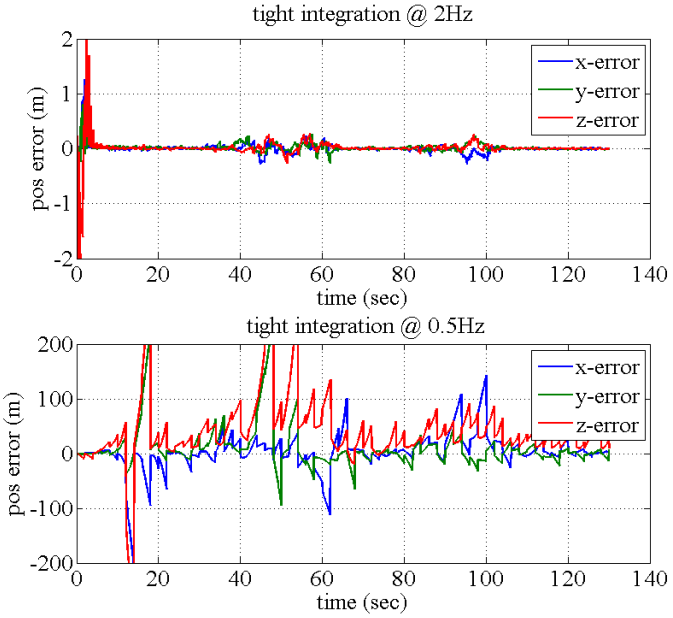

Figure 4.13: Position Error of Tight Integration

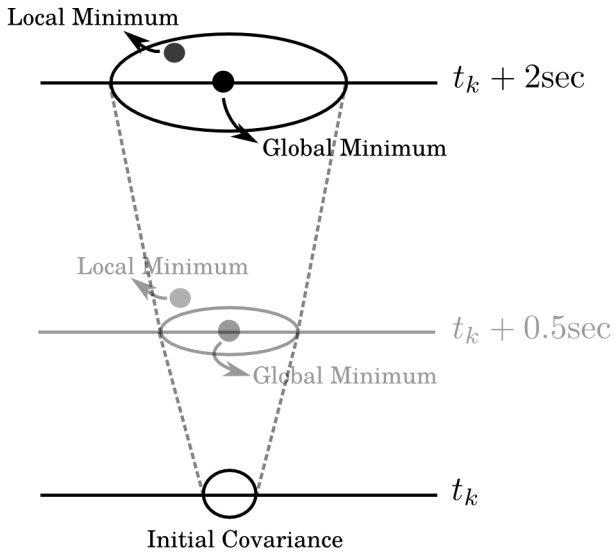

Figure 4.14: Inclusion of Local Minima at Low Measurement Update Rate

Figure 4.15: Interactive Guidance and Control Lab (Courtesy of Jon Andersh)

Figure 4.16: Photo of AR Drone (Courtesy of Jon Andersh)

First step of this experiment is to generate a reference image with geo-tagged feature points. Given two or more images taken from a calibrated camera at known location and attitude, the 3D positions of matched feature pairs can be calculated by triangulation techniques [61, 62]. In this experiment, two images are taken from calibrated SONY DSC-TX100V as shown in Figure 4.17(a) and 4.17(b). The calibrated camera has resolution of 3000×2250 pixels and its intrinsic matrix is given

$$\mathbf{M}_{ref\ cam} = \begin{bmatrix} 2102.25 & 0 & 1476.28 \\ 0 & 2101.84 & 1127.20 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.129}$$

The position and attitude of the camera are captured by the Vicon system. SIFT algorithm scans through images and search for matched pairs. Noted the image distortion has been compensated prior to the triangulation. The pixel locations of the matched pairs along with the camera's position, attitude and calibration parameters are used to solve for the locations of identified feature points. Figure 4.18 shows the triangulation result of the two view geometry. The blue dots represents the locations of identified feature points. The positions and poses of the camera are also shown in the figure.

(a) First View (b) Second View

Figure 4.17: Reference Images For Triangulation

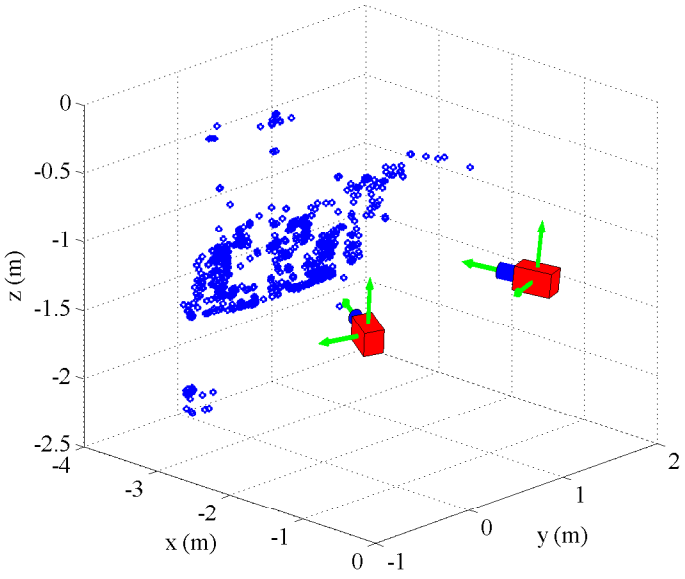

Figure 4.18: Image Triangulation Result

The trajectory is flown by a quadrotor equipped with a low cost consumer grade IMU and a forward looking camera with resolution (320×240). The camera is calibrated

and the intrinsic matrix is given

$$\mathbf{M}_{AR\ drone\ cam} = \begin{bmatrix} 221.50 & 0 & 154.52 \\ 0 & 223.78 & 115.36 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.130}$$

The image distortion is also compensated in image pre-processing. The trajectory and attitude of the quadrotor is also captured by the Vicon tracking system and will be used as reference for evaluating the accuracy of the filter result. The onboard IMU can generate measurements at 100 Hz and the camera produces image stream up to 15 frame per second (FPS). The onboard data is transmitted through wireless connection to a data acquisition computer where the information is time-tagged and saved for post-processing. The SIFT algorithm will first compare the streaming images against one of the reference images (Figure 4.17(a) and 4.17(b)) to identify possible matches. RANSAC algorithm is applied to remove faulty matches from the correct ones. The pixel locations of correct matches and their associate absolute location are integrated with IMU in loosely coupled architecture for generating filter estimates. The process of the experiment is shown in Figure 4.19.

The estimate results from a 2 minutes test are shown in Figure 4.20 and 4.21. Figure 4.20 shows the position error and the attitude error is given in Figure 4.21 as the measurement comes in at 4 FPS. The position accuracy can achieve approximately 20 centimeters whereas the attitude accuracy is roughly 5 degrees. From the figures, the pixel noises seem to have larger influence on the attitude accuracy than the position accuracy. This impact depends on the mean distance from the feature points to the camera. For feature points closer to the camera, the pixel errors will have larger influence on attitude than position. As the distance grows, the impact on attitude will become smaller, but the impact on position will grow. This conclusion can be easily presented

by performing the sensitivity analysis of position and attitude of the camera to camera-feature distance. Also noted the position and attitude estimates are biased. The bias is due to calibration error. The error can come from sources such as camera calibration errors from the reference camera or the onboard camera or the residual errors of the image distortion. The calibration of the onboard camera installation angle and position can also contribute to the biased error. The experiment is also tested on the tightly coupled approach. However, since the identified features are very close to the camera, tight architecture fails to generate stable solution due to the explanation discussed in section 3.8. The presentation of the testing results are skipped in this section.

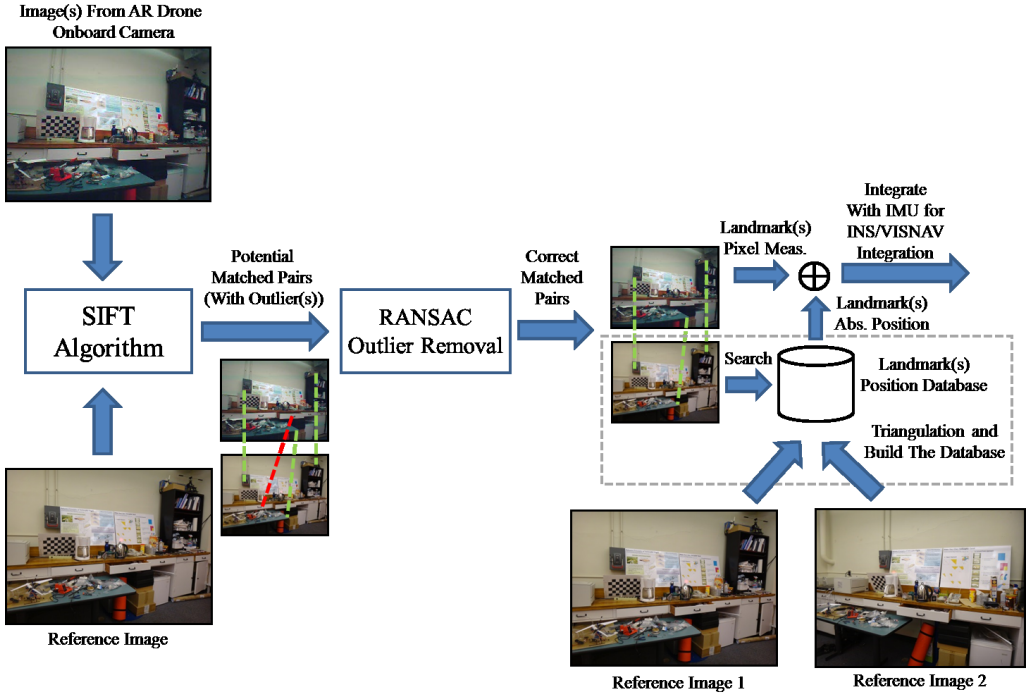

Figure 4.19: The Process of the Experiment

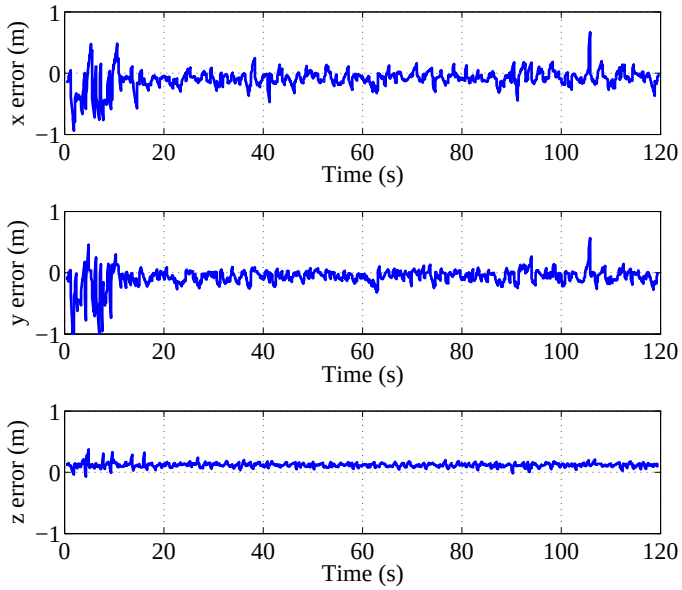

Figure 4.20: Position Error of Loose Experimental Results

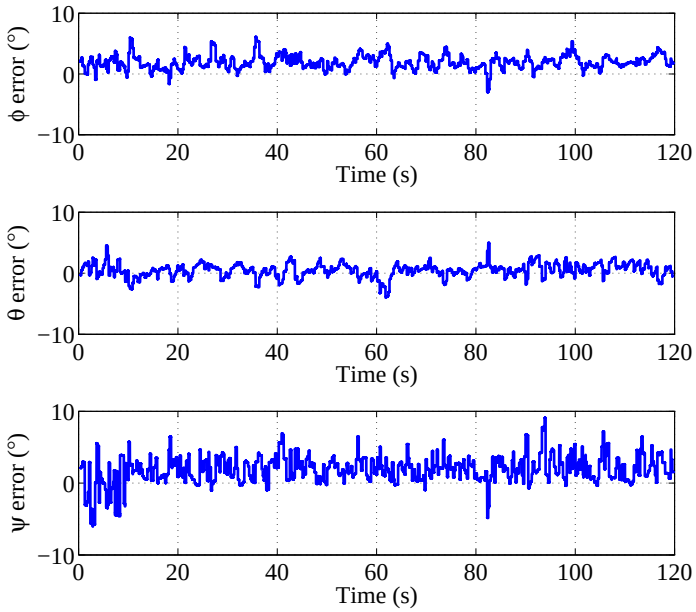

Figure 4.21: Attitude Error of Loose Experimental Results

Chapter 5

Multiple Hypothesis Filter Fusion

The Multiple Hypothesis Filter (MHT) method is a Bayesian framework that performs multiple hypothesis testing on a bank of Kalman filters that run in parallel. In our case the bank will consist of two filters (e.g., loose and tight integration architectures). Each filter has a different mathematical model and the MHT technique optimally weighs the solution of each filter to generate the final estimate. The weights are constantly updated at every measurement epoch by computing the likelihood of each filter output. The model with higher likelihood will be assigned a larger weight and thus contributes more to the final solution. In the INS/Camera problem, the switching between loose and tight integration architectures is done by assigning different weights to each model. This kind of switching mechanism is known as"soft-switching". In contrast, "hard-switching" is the case where only one architecture will be selected at a time. The advantage of the multiple hypothesis filter is that it can be built on top of the existing Kalman filters and therefore preserves the original structure of each filter. This greatly reduces the filter design effort.

5.1 Dual Hypothesis Filter

The INS/Camera fusion algorithm discussed here is a dual-hypothesis filter (DHT). This is another way of saying that there are two sub-filters representing two hypotheses. The filters run side-by-side and the navigation solution reported by the algorithm is based on the optimal blending of the two hypotheses. The first hypothesis is that the linearized pin-hole camera model (i.e. the tight integration measurement equation) is convergent. The second hypothesis is the complement of the first; that is the linearized pin-hole camera model is divergent. Under the second hypothesis a sub-optimal, two-step estimator is used to form the navigation solution. This estimator is what we call the loose integration strategy.

5.1.1 Time and Measurement Update Models

The time and measurement update equations, respectively, for the two sub-filters associated with the two hypotheses have the following form:

$$\mathbf{x}_k = \boldsymbol{\mathcal{F}}(m_{k-1,}, \mathbf{x}_{k-1}) + \mathbf{w}(m_{k-1}, k-1) \tag{5.1}$$

However, since the measurement models are different (i.e. tight vs loose) then,

$$\mathbf{y}_k = \boldsymbol{\mathcal{H}}(m_k, \mathbf{x}_k) + \mathbf{v}(m_k, k) \tag{5.2}$$

where $\mathbf{x}_k$ is the navigation state vector (position, velocity, attitude and IMU sensor biases); $\mathbf{y}_k$ is the measurement vector; $\mathbf{w}_k = \mathbf{w}(m_k, k)$ is the process noise representing uncertainties resulting from IMU output errors; and $\mathbf{v}_k = \mathbf{v}(m_k, k)$ is the measurement noise. The variable m_k is the *hypothesis index* and indicates which hypothesis is in effect. When $m_k = 1$ the first hypothesis is in effect and when $m_k = 2$ the second hypothesis is in effect. Under the first hypothesis, the measurement vector $\mathbf{y}_k$ consists of raw pixel measurements from the camera and under the second hypothesis $\mathbf{y}_k$ consists of the position and attitude of the user estimated using camera measurements.

The function $\boldsymbol{\mathcal{F}}$ in the time update equation is nothing more than the INS mechanization equations and is identical for both hypotheses. Therefore,

$$\boldsymbol{\mathcal{F}}(1, \mathbf{x}_{k-1}) = \boldsymbol{\mathcal{F}}(2, \mathbf{x}_{k-1}) \tag{5.3}$$

$$\boldsymbol{\mathcal{H}}(1, \mathbf{x}_{k-1}) \neq \boldsymbol{\mathcal{H}}(2, \mathbf{x}_{k-1}) \tag{5.4}$$

The hypothesis index m_k is a discrete random variable with probability mass function (pmf) μ_k. It is used as weight when combining solutions from the two sub-filters. Its value is not constant and needs to be calculated at every measurement epoch. In the sections that follow, the recursion that calculates the weight μ_k and the equations that combine the sub-filter solutions using μ_k are derived.

5.1.2 Filter Timeline

To understand the dual hypothesis filter, it is helpful to construct a timeline that depicts how the filter processes information from the sub-filters. Figure 5.1 shows a timeline of a dual hypothesis filter. Filter i on the left and filter j on the right are the tight and loose integration filters respectively. Each of these filters is run more or less the way described in Chapter 3 and Chapter 4. However, there are two key differences. The first difference has to do with how the time update equations are implemented and the second one is how the reported navigation state vector output is computed. We will "walk" through one cycle of the filter to highlight these differences.

One cycle of the filter starting from the completion of the measurement update at t_{k-1} to the next measurement update at t_k is depicted in Figure 5.1. One complete filter cycle consists of the following steps in sequence:

1. ***Time Update/Model Interaction from*** $t = t_{k-1}$ ***to*** $t = t_k$. In this step, the state vectors of both hypotheses are propagated forward in time using the INS equations. After performing this update, the two hypotheses interact so that a

blended estimate that takes into account the estimate from both hypothesis. It also involves estimates exchange between both hypothesis The details of the model interaction process are discussed when we derive the equations below. However, an intuitive feel for why this blending is necessary can be obtained by noting the following. There is a possibility that after the measurement update at t_{k-1}, one of the filters (more than likely the tight filter) has diverged. Without the interaction step, a divergent state estimate would be propagated in the time update equations which would exacerbate the situation. The interaction step is a way of performing a "soft" reset of the states of the two filters using information exchange between them. Note that for this problem the interaction occurs after performing the time update. For linear time update process, this step can be executed in any order.

2. ***Reporting of the a priori estimate at*** $t = t_k$. In this step, the state estimates after INS propagation and model interaction for hypotheses i and j are combined optimally to generate the final reported navigation state vector estimate. As will be shown in the next section, the combination is nothing more that a weighted summing of the solutions from the i^{th} and j^{th} filters.

3. ***Measurement Update at*** $t = t_k$. The *a priori* state estimates of each individual hypothesis are fused with camera measurements at their respective level. Tight integration architecture would fuse the camera pixel measurements while loose integration architecture would fuse the position and attitude estimates from the camera. The model weights of each hypothesis/filter are also updated.

4. ***Reporting of the*** **a posteriori** ***estimate at*** $t = t_k$. In this step, the posterior state estimates from the i^{th} and j^{th} hypotheses are combined optimally to generate the final reported navigation state vector estimates. After this step, we are ready for the next time update/model interaction.

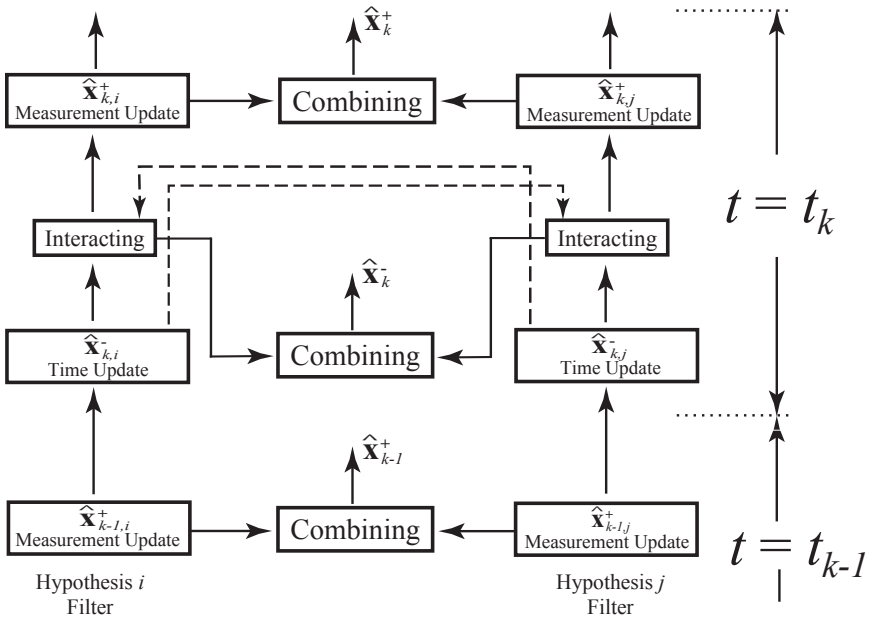

Figure 5.1: Timeline of the dual hypothesis filter.

5.1.3 Filter Variables Definition

The equations associated with each of the steps in the filter cycle described above will be derived in the next section. Before we proceed with the derivation, however, we will define the variables that will be used.

We use the notation $\hat{\mathbf{x}}_{k,i}$ where $i = 1, 2$ for the states of the the two sub-filters respectively. The variable $\hat{\mathbf{x}}_k$ is used to denote the reported states (i.e. the output of the navigation algorithm). It is a function of the states from the two hypotheses/filters. We will use the notation $\mathbf{Y}_k$ to indicate a set consisting of all measurements up to time step t_k. That is

$$\mathbf{Y}_k = \{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \ldots, \mathbf{y}_k\} \tag{5.5}$$

The superscripts "$-$" and "$+$" will be used to indicate *a priori* (before measurement update) and *a posteriori* (after measurement update) estimates of states and weights.

More precisely, $\hat{\mathbf{x}}_k^-$ and $\hat{\mathbf{x}}_k^+$ are defined as follows:

$$\hat{\mathbf{x}}_k^- \triangleq E\{\mathbf{x}_k|\mathbf{Y}_{k-1}\} = \int_{-\infty}^{\infty} \mathbf{x}_k \, p(\mathbf{x}_k|\mathbf{Y}_{k-1}) d\mathbf{x}_k \tag{5.6}$$

$$\hat{\mathbf{x}}_k^+ \triangleq E\{\mathbf{x}_k|\mathbf{Y}_k\} = \int_{-\infty}^{\infty} \mathbf{x}_k \, p(\mathbf{x}_k|\mathbf{Y}_k) d\mathbf{x}_k \tag{5.7}$$

where $p(\mathbf{x}_k|\mathbf{Y}_{k-1})$ and $p(\mathbf{x}_k|\mathbf{Y}_k)$ are the probability density functions of $\mathbf{x}_k$ conditioned on $\mathbf{Y}_{k-1}$ and $\mathbf{Y}_k$, respectively. For each sub-filter state, $\hat{\mathbf{x}}_{k,i}^-$ and $\hat{\mathbf{x}}_{k,i}^+$ are defined as:

$$\begin{aligned} \hat{\mathbf{x}}_{k,i}^- &\triangleq E\{\mathbf{x}_k|m_k = i, \mathbf{Y}_{k-1}\} \\ &= \int_{-\infty}^{\infty} \mathbf{x}_k \, p(\mathbf{x}_k|m_k = i, \mathbf{Y}_{k-1}) d\mathbf{x}_k \end{aligned} \tag{5.8}$$

$$\begin{aligned} \hat{\mathbf{x}}_{k,i}^+ &\triangleq E\{\mathbf{x}_k|m_k = i, \mathbf{Y}_k\} \\ &= \int_{-\infty}^{\infty} \mathbf{x}_k \, p(\mathbf{x}_k|m_k = i, \mathbf{Y}_k) d\mathbf{x}_k \end{aligned} \tag{5.9}$$

As noted earlier, the hypothesis index m_k is a discrete random variable whose probability mass function or weight needs to be estimated recursively. In deriving the equations for its recursion we will adopt the following notations:

$$\mu_{k,i}^- = P\{m_k = i|\mathbf{Y}_{k-1}\} \tag{5.10}$$

$$\mu_{k,i}^+ = P\{m_k = i|\mathbf{Y}_k\} \tag{5.11}$$

where $i \in \{1, 2\}$. Stated in words, $\mu_{k,i}^-$ is the probability that $m_k = i$ conditioned on all measurements up to time step t_{k-1} while $\mu_{k,i}^+$ is the probability that $m_k = i$ conditioned on all measurements up to and including time step t_k.

5.2 Filter Derivation

In this section, the equations for the dual hypothesis filter are derived. While, the inspiration for this filter is the Interactive Multiple Model (IMM) filter described in [63],

there are some important differences which are unique to camera-aided inertial navigation problem. Therefore, we will derive the measurement and time update equations in sequence while keeping in mind the specifics of the problem we are dealing with here. We will derive the relevant equations from Bayesian point of view [64].

5.2.1 Measurement Update Equations

To develop a mathematical relationship between the *a posteriori* optimal system state estimate $\hat{\mathbf{x}}_k^+$ and the $r = 2$ distinct estimates $\hat{\mathbf{x}}_{k,i}^+$ (corresponding to each of the two hypothesis sub-filters) we first use the law of total probability to write [65]:

$$p(\mathbf{x}_k|\mathbf{Y}_k) = \sum_{i=1}^{r} p(\mathbf{x}_k|m_k = i, \mathbf{Y}_k) P\left\{m_k = i|\mathbf{Y}_k\right\} \tag{5.12}$$

Substituting this into Equation (5.7) gives:

$$\hat{\mathbf{x}}_k^+ = \int_{-\infty}^{\infty} \mathbf{x}_k \Big[\sum_{i=1}^{r} p(\mathbf{x}_k|m_k = i, \mathbf{Y}_k) \times P\left\{m_k = i|\mathbf{Y}_k\right\} \Big] d\mathbf{x}_k \tag{5.13}$$

Taking the integral sign into the summation and noting that $P\left\{m_k = i|\mathbf{Y}_k\right\} = \mu_{k,i}^+$ gives:

$$\hat{\mathbf{x}}_k^+ = \sum_{i=1}^{r} \left[\int_{-\infty}^{\infty} \mathbf{x}_k p(\mathbf{x}_k|m_k = i, \mathbf{Y}_k) d\mathbf{x}_k \right] \mu_{k,i}^+ \tag{5.14}$$

$$= \sum_{i=1}^{r} \mu_{k,i}^+ \hat{\mathbf{x}}_{k,i}^+ \tag{5.15}$$

where $r = 2$ and $\hat{\mathbf{x}}_{k,i}^+$ is the *a posteriori* state estimate (i.e., post-measurement update) of the i^{th} filter. The algorithms for calculating $\hat{\mathbf{x}}_{k,i}^+$ have been discussed in section 3.1 and section 4.2, we will not repeat the discussion in this section.

To derive an expression for the a posteriori covariance we first note that Equation (5.12) can be generalized as:

$$p(\mathbf{x}) = \sum_{i=1}^{r} w_i p_i(\mathbf{x}) \tag{5.16}$$

where w_i is the weighting factor which subject to

$$\sum_{i=1}^{r} w_i = 1 \tag{5.17}$$

and $p_i(\mathbf{x})$ is the pdf of i^{th} hypothesis. For any function $g(\cdot)$, the expectation of $g(\cdot)$ is

$$\begin{aligned} E[g(\mathbf{x})] &= \int_{-\infty}^{\infty} g(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \\ &= \int_{-\infty}^{\infty} g(\mathbf{x}) \sum_{i=1}^{r} w_i p_i(\mathbf{x}) d\mathbf{x} \\ &= \sum_{i=1}^{r} w_i \int_{-\infty}^{\infty} p_i(\mathbf{x}) g(\mathbf{x}) d\mathbf{x} \\ &= \sum_{i=1}^{r} w_i E[g(\mathbf{x}_i)] \end{aligned} \tag{5.18}$$

Now let us define the function $g(\mathbf{x})$ as:

$$g(\mathbf{x}) = (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^T \tag{5.19}$$

where $\overline{\mathbf{x}} = E[\mathbf{x}]$ is the expected value of $\mathbf{x}$. By applying the relation as shown in Equation (5.18), the covariance of $\mathbf{x}$

$$\begin{aligned} \mathbf{P} &\triangleq E[(\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^T] \\ &= \sum_{i=1}^{r} w_i E[(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T] \\ &= \sum_{i=1}^{r} w_i E[(\mathbf{x}_i - \overline{\mathbf{x}}_i + \overline{\mathbf{x}}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}}_i + \overline{\mathbf{x}}_i - \overline{\mathbf{x}})^T] \end{aligned} \tag{5.20}$$

By grouping and collecting terms, the equation can be rearranged as:

$$\begin{aligned}\mathbf{P} = &\sum_{i=1}^{r} w_i E[(\mathbf{x}_i - \overline{\mathbf{x}}_i)(\mathbf{x}_i - \overline{\mathbf{x}}_i)^T] + \\ &\sum_{i=1}^{r} w_i E[(\mathbf{x}_i - \overline{\mathbf{x}}_i)(\overline{\mathbf{x}}_i - \overline{\mathbf{x}})^T] + \\ &\sum_{i=1}^{r} w_i E[(\overline{\mathbf{x}}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}}_i)^T] + \\ &\sum_{i=1}^{r} w_i E[(\overline{\mathbf{x}}_i - \overline{\mathbf{x}})(\overline{\mathbf{x}}_i - \overline{\mathbf{x}})^T] \end{aligned} \tag{5.21}$$

Note that the first term is the covariance of i^{th} model and denoted as $\mathbf{P}_i$. The second and third terms are zero which can be easily demonstrated by expanding each terms. The expectation operator can be dropped in the fourth term since the quantities are not random variables but estimates of them. Therefore, the covariance of $\mathbf{x}$ can be computed:

$$\mathbf{P} = \sum_{i=1}^{r} w_i \{\mathbf{P}_i + (\overline{\mathbf{x}}_i - \overline{\mathbf{x}})(\overline{\mathbf{x}}_i - \overline{\mathbf{x}})^T\} \tag{5.22}$$

This result is used for covariance estimation. If we substitute $w_i = \mu_{k,i}^+$, $\overline{\mathbf{x}}_i = \hat{\mathbf{x}}_{k,i}^+$ and $\overline{\mathbf{x}} = \hat{\mathbf{x}}_k^+$, then we has the following expression or the covariance of the state estimate:

$$\hat{\mathbf{P}}_k^+ = \sum_{i=1}^{r} \mu_{k,i}^+ \{\hat{\mathbf{P}}_{k,i}^+ + [\hat{\mathbf{x}}_{k,i}^+ - \hat{\mathbf{x}}_k^+][\hat{\mathbf{x}}_{k,i}^+ - \hat{\mathbf{x}}_k^+]^T\} \tag{5.23}$$

To derive an expression for the weight $\mu_{k,i}^+$ we start by noting that

$$\mu_{k,i}^+ = P\{m_k = i | \mathbf{Y}_k\} = P\{m_k = i | \mathbf{y}_k, \mathbf{Y}_{k-1}\} \tag{5.24}$$

Even though $\mathbf{y}_k$ is a continuous random variable, the limiting arguments given in [65] can be used to show that Bayes' rule for the mixture of continuous and discrete random

variables can be written as:

$$\begin{aligned}\mu_{k,i}^{+} &= \frac{p(\mathbf{y}_k|m_k=i,\mathbf{Y}_{k-1})P\left\{m_k=i|\mathbf{Y}_{k-1}\right\}}{\sum_{j=1}^{r}\left[p(\mathbf{y}_k|m_k=j,\mathbf{Y}_{k-1})P\left\{m_k=j|\mathbf{Y}_{k-1}\right\}\right]} \\ &= \frac{p(\mathbf{y}_k|m_k=i,\mathbf{Y}_{k-1})\mu_{k,i}^{-}}{\sum_{j=1}^{r}\left[p(\mathbf{y}_k|m_k=j,\mathbf{Y}_{k-1})\mu_{k,i}^{-}\right]}\end{aligned} \tag{5.25}$$

The first term in the numerator of the expression above is the measurement likelihood function. It can be determined from the measurement model (Equation (5.2)) if we know the statistical properties of the measurement noise $\mathbf{v}_k$. For the work reported here we make the following assumptions:

$$\mathbf{v}_k \sim N\left(0,\mathbf{R}\right) \tag{5.26}$$

$$E\left\{\mathbf{v}_k\mathbf{v}_k^T\right\} = \begin{cases}\mathbf{R} & j\neq k \\ 0 & j=k\end{cases} \tag{5.27}$$

The first of the assumptions above is that the measurement noise is a zero-mean, Gaussian vector with covariance $\mathbf{R}$ while the second assumption states that it is uncorrelated in time. The assumption that the measurement noise is independent in time allows us to write:

$$p(\mathbf{y}_k|m_k=i,\mathbf{Y}_{k-1}) = p(\mathbf{y}_k|m_k=i) \tag{5.28}$$

Since the probability density function of the measurement noise $p(\mathbf{v}_k)$ is given by

$$p(\mathbf{v}_k) = \frac{1}{\sqrt{2^N\pi^N|\mathbf{R}|}}e^{-\frac{1}{2}\left[\mathbf{v}_k^T\mathbf{R}^{-1}\mathbf{v}_k\right]} \tag{5.29}$$

where N is total number of landmarks in view, we can invert the measurement equation $\mathbf{v}_k = \mathbf{y}_k - \mathbf{h}(m_k,\mathbf{x}_k)$ to get:

$$p(\mathbf{y}_k|m_k=i) = \frac{e^{-\frac{1}{2}\left[(\mathbf{y}_k-\mathbf{h}(i,\mathbf{x}_k))^T\mathbf{R}^{-1}(\mathbf{y}_k-\mathbf{h}(i,\mathbf{x}_k))\right]}}{\sqrt{2^N\pi^N|\mathbf{R}|}} \tag{5.30}$$

Note that $\mathbf{x}_k$ is a known parameter in the probability density function given above and thus:

$$p(\mathbf{y}_k|m_k = i) = p(\mathbf{y}_k|\mathbf{x}_{k,i}) \sim N\left(\mathbf{h}(i, \mathbf{x}_k), \mathbf{R}\right) \tag{5.31}$$

For ease of notation we will define:

$$\Lambda_{k,i} \triangleq p(\mathbf{y}_k|\mathbf{x}_{k,i}) \tag{5.32}$$

Since at the measurement update $\mathbf{x}_{k,i} = \hat{\mathbf{x}}^+_{k,i}$, a numerical value for $\Lambda_{k,i}$ can be easily computed. Putting all of this together, we can write the following expression for $\mu^+_{k,i}$:

$$\mu^+_{k,i} = \frac{\Lambda_{k,i}\mu^-_{k,i}}{\sum_{j=1}^{r} \Lambda_{k,j}\mu^-_{k,j}} \tag{5.33}$$

This is the measurement update equation for the hypothesis index weight.

5.2.2 Time Update Equations

The derivation of the time update equations is mathematically straightforward. However, an understanding of the physical meaning of the resulting equations is not intuitive. We will first go through the mechanics of deriving the equations. Then at the end of this section we will attempt to give a physical interpretation of what the time update equations accomplish.

Our goal is to re-write Equation (5.8) in a form that is suitable for a recursive estimation algorithm. To do this we will first use Bayes' rule to rewrite $p(\mathbf{x}_k|m_k = i, \mathbf{Y}_{k-1})$ as follows:

$$p(\mathbf{x}_k|m_k = i, \mathbf{Y}_{k-1}) = \frac{p(\mathbf{x}_k, m_k = i|\mathbf{Y}_{k-1})}{P\left\{m_k = i|\mathbf{Y}_{k-1}\right\}} \tag{5.34}$$

To simplify the denominator of the expression above, we note that the hypothesis index m_k can be viewed as a Markov random process with two states. The system can be

graphically depicted as shown in Figure 5.2 where π_{ij} is a measure of the probability or likelihood that a value of hypothesis index equal to j at time step t_{k-1} will transition to a value of i at time step t_k.

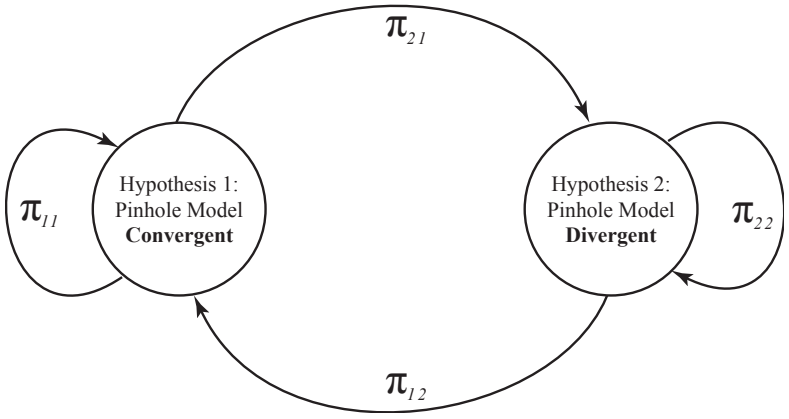

Figure 5.2: The two hypotheses as a Markov process

This is a parameter which is used to specify the filter bandwidth. Filter with large π_{ij} where $i \neq j$ has higher bandwidth meaning it is more likely to switch between two models. However, the price we pay for this is that the filter output is noisier. From properties of Markov random processes and the Chapman-Kolmogorov equation we can write [66]:

$$\begin{aligned} P\left\{m_k = i|\mathbf{Y}_{k-1}\right\} &= \sum_{j=1}^{r} \pi_{ij} P\left\{m_{k-1} = j|\mathbf{Y}_{k-1}\right\} \\ &= \sum_{j=1}^{r} \pi_{ij} \mu_{k-1,j}^{+} \end{aligned} \tag{5.35}$$

or using our compact notation

$$\mu_{k,i}^{-} = \sum_{j=1}^{r} \pi_{ij} \mu_{k-1,j}^{+} \tag{5.36}$$

This is the time update equation for the hypothesis index probability mass function or what we simply refer to as the model weights. To get to the time update equation for

the state we need to further simplify the numerator of Equation (5.34). Once again using Bayes's rule we can write:

$$\begin{aligned} p(\mathbf{x}_k, m_k = i|\mathbf{Y}_{k-1}) &= \sum_{j=1}^{r} p(\mathbf{x}_k, m_k = i|m_{k-1} = j, \mathbf{Y}_{k-1}) P\left\{m_{k-1} = j|\mathbf{Y}_{k-1}\right\} \\ &= \sum_{j=1}^{r} p(\mathbf{x}_k, |m_{k-1} = j, \mathbf{Y}_{k-1}) \pi_{ij} \mu^{+}_{k-1,j} \end{aligned} \tag{5.37}$$

where in the last step we have made use of the fact that at time step t_k, Equation (5.1) is not a function of m_k. Substituting all of this back into Equation (5.34) gives:

$$p(\mathbf{x}_k|m_k = i, \mathbf{Y}_{k-1}) = \frac{\sum\limits_{j=1}^{r} p(\mathbf{x}_k, |m_{k-1} = j, \mathbf{Y}_{k-1}) \pi_{ij} \mu^{+}_{k-1,j}}{\mu^{-}_{k,i}} \tag{5.38}$$

Substituting Equation (5.38) into Equation (5.8) leads to:

$$\begin{aligned} &\hat{\mathbf{x}}^{-}_{k,i} \triangleq E\{\mathbf{x}_k|m_k = i, \mathbf{Y}_{k-1}\} \\ &= \int_{-\infty}^{\infty} \mathbf{x}_k \left[\frac{\sum\limits_{j=1}^{r} p(\mathbf{x}_k, |m_{k-1} = j, \mathbf{Y}_{k-1}) \pi_{ij} \mu^{+}_{k-1,j}}{\mu^{-}_{k,i}} \right] d\mathbf{x}_k \\ &= \frac{\sum\limits_{j=1}^{r} \left[\int_{-\infty}^{\infty} \mathbf{x}_k p(\mathbf{x}_k, |m_{k-1} = j, \mathbf{Y}_{k-1}) d\mathbf{x}_k \right] \pi_{ij} \mu^{+}_{k-1,j}}{\mu^{-}_{k,i}} \end{aligned} \tag{5.39}$$

To simplify the expression above further, note that per our time update model (Equation (5.1)) the process noise $\mathbf{w}_k$ is assumed to be additive. Furthermore, if we assume that that the process noise $\mathbf{w}_k$ is a zero-mean, Gaussian white sequence [67] given by

$$\mathbf{w}_k \sim N(0, \mathbf{Q}) \tag{5.40}$$

$$E\left\{\mathbf{w}_k \mathbf{w}_k^T\right\} = \begin{cases} \mathbf{Q} & j \neq k \\ 0 & j = k \end{cases} \tag{5.41}$$

then using an argument similar to the one used in deriving Equation (5.30), we can show the value of the term in brackets in Equation (5.39) is nothing more than $\boldsymbol{\mathcal{F}}(\hat{\mathbf{x}}^+_{k-1,i})$. This means that Equation (5.39) can be written as:

$$\hat{\mathbf{x}}^-_{k,i} = \frac{\sum_{j=1}^{r} \boldsymbol{\mathcal{F}}(\hat{\mathbf{x}}^+_{k-1,i})\pi_{ij}\mu^+_{k-1,j}}{\mu^-_{k,i}} \tag{5.42}$$

Finally, using an argument similar to the one used in deriving the covariance for the measurement update, the covariance estimate for time update can be computed:

$$\hat{\mathbf{P}}^-_{k,i} = \sum_{i=1}^{r} \frac{\pi_{ij}\mu^+_{k-1,j}}{\mu^-_{k,i}} \{\hat{\mathbf{P}}^+_{k-1,i} + [\hat{\mathbf{x}}^-_{k,i} - \hat{\mathbf{x}}^-_{k,j}][\hat{\mathbf{x}}^-_{k,i} - \hat{\mathbf{x}}^-_{k,j}]^T\} \tag{5.43}$$

This completes the recursion formulas for the time update equation.

5.3 Experimental Results

The dual hypothesis filter is validated by post-processing collected in a flight test where we use a real IMU combined with the simulated feature points. This is because the feature points information is not available in the environment where the flight test is conducted. The reference trajectory and IMU measurement is provided by [68]. It is flown by an AscTech Pelican quadrotor helicopter with an InterSense NavChip IMU onboard (shown in Figure 5.3) in Electrical Engineering/Computer Science Building at the University of Minnesota as shown in Figure 5.4. A pre-calibrated camera with resolution 640x480 is assumed to be mounted on the airframe with "forward-looking" orientation. The camera intrinsic matrix is

$$\mathbf{M}_{Pelican\ cam} = \begin{bmatrix} 480.07 & 0 & 346.68 \\ 0 & 480.10 & 249.00 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.44}$$

The pre-surveyed landmarks are simulated and plotted as green dots in Figure 5.4 and the red dots are the quadrotor's position every 10 seconds. The vehicle circled the indoor hallway clockwise and returned to the initial position in approximately 2.5 minutes. The total distance traveled is approximately 150 meters. The image update rates are assumed to be 2 and 0.5 FPS. The landmark projections on the pixel frame are corrupted by Gaussian noise with $\sigma = 1.4$ pixels for both horizontal and vertical directions.

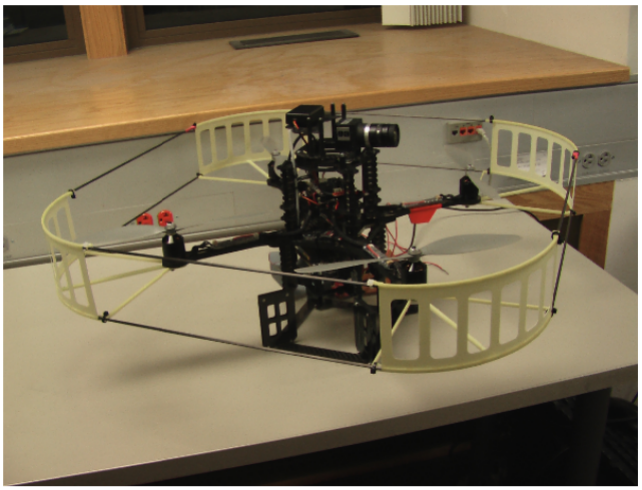

Figure 5.3: AscTech Pelican Quadrotor Testbed (Courtesy of Dimitrios Kottas)

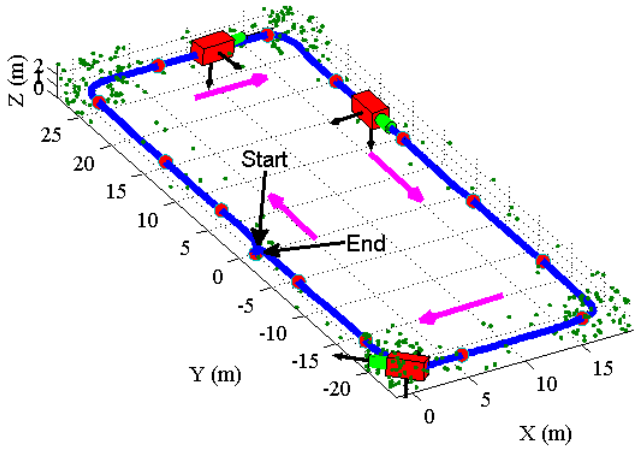

Figure 5.4: Real Trajectory and Simulated Feature Points

We first test the standalone loose and tight integration filter independently at the two measurement update rates to obtain a baseline result for comparison later. The results are plotted in terms of position error as shown in Figure 5.5 and 5.6, respectively. At a 2 Hz measurement update rate, both loose and tight architectures generate a stable solution. The tight integration outperforms the loose architecture in terms of estimation accuracy due to its optimal fusion strategy.

At 0.5 Hz, the loosely integrated architecture still generates stable solution even though the estimation accuracy is significantly decreased whereas the tight integration produces divergent solution which is completely unusable.

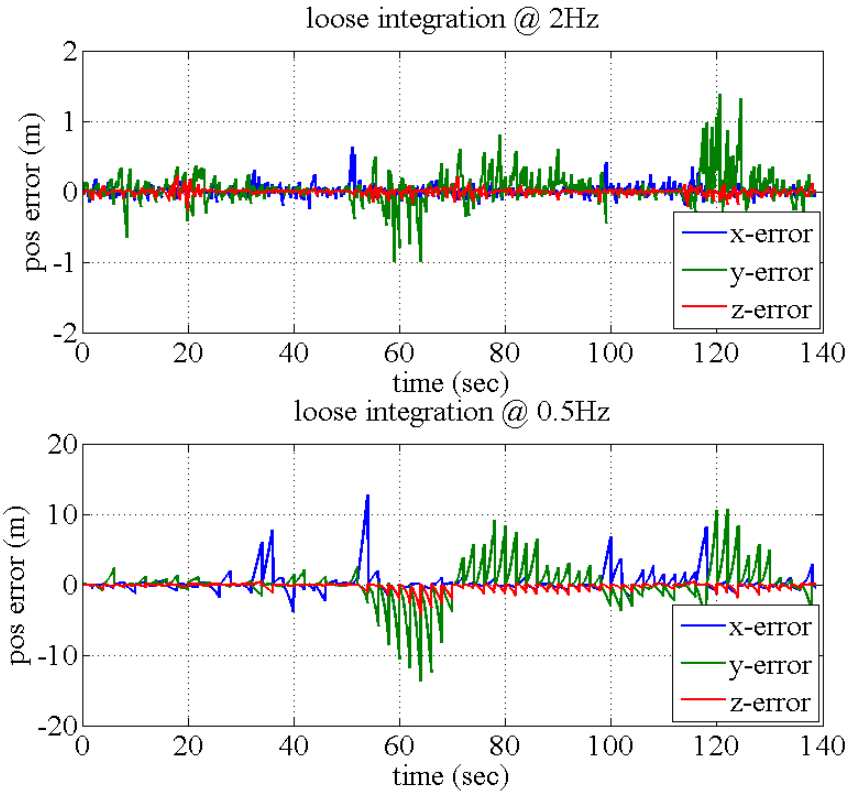

Figure 5.5: Position Error of Loose Integration

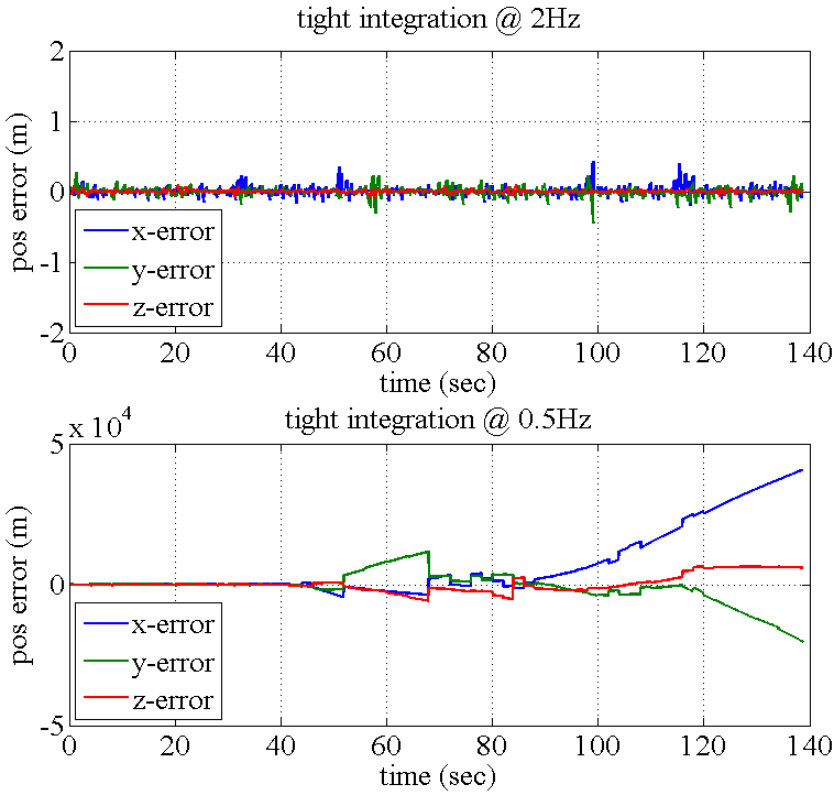

Figure 5.6: Position Error of Tight Integration

The results of the dual hypothesis filter running at 2 Hz and 0.5 Hz are shown in Figure 5.7 and 5.8, respectively. The position error is plotted in the upper figure and model weight assigned to loose and tight architectures are shown in the lower plot. Figure 5.7 depicted the position error at the 2 Hz measurement update. As expected, the dual hypothesis filter put higher weight on the tight architecture since tight architecture is stable and generates more accurate solution. Therefore, the blended solution is closer to the result of standalone tight integration at 2 Hz.

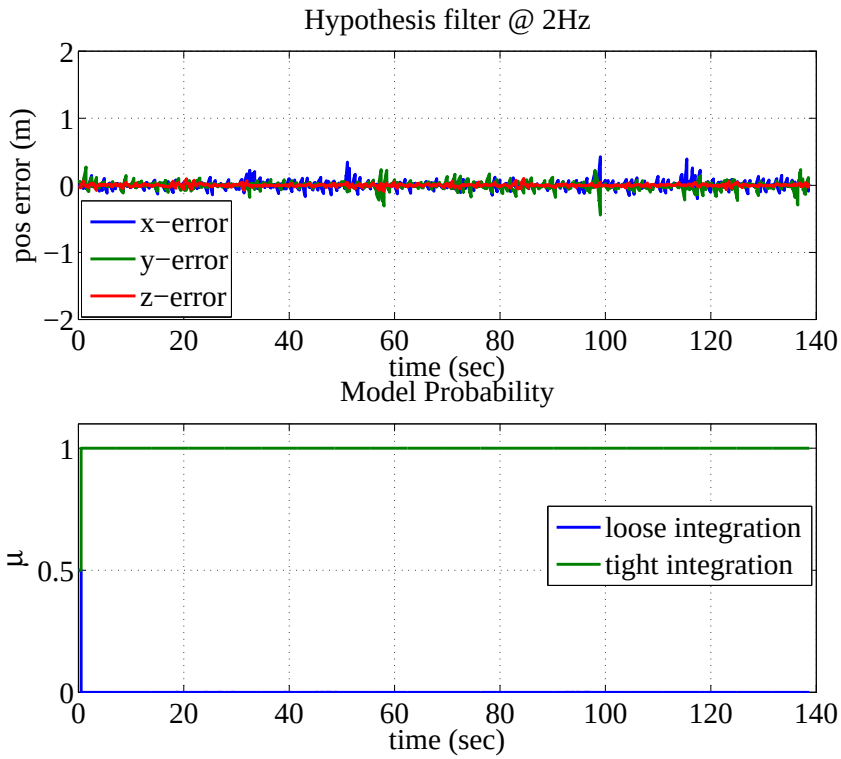

Figure 5.7: Position Error and Model Weight of DHT Filter @ 2Hz

Figure 5.8 shows when the dual hypothesis filter is at 0.5 Hz, model switching occurred at multiple time instances. This is due to unfavorable landmark geometry causing the tight integration filter to diverge. This is in line with the figure shown in Figure 5.6. Therefore, the filter assigns higher weight to loose architecture and lower the contribution of tight integration to the final estimate but still keep tight architecture in the loop. As the vehicle passes through the unfavorable region, the filter reverts back to weighing the tight integration higher. The result is stable compared with the standalone tight integration (bottom plot in Figure 5.6) and outperforms the standalone loose architecture (bottom plot in Figure 5.5) in terms of accuracy since tight integration has higher contribution to the final estimation most of the time in the span of the testing.

The statistics of position error (standard deviation of position error) of loose, tight and dual hypothesis filter is tabulated in Table 5.1.

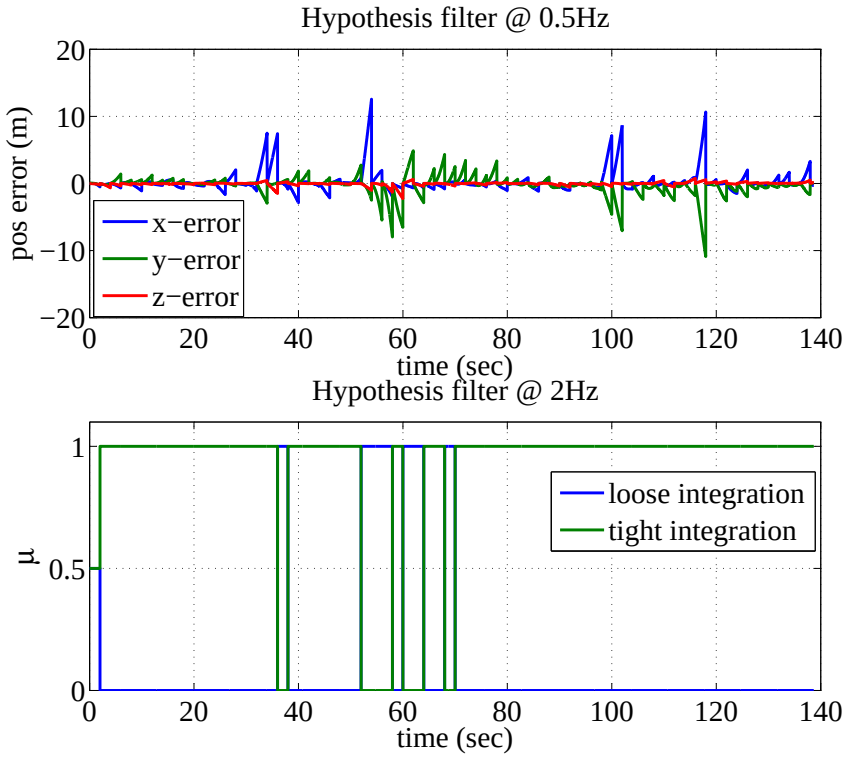

Figure 5.8: Position Error and Model Weight of DHT Filter @ 0.5Hz

Table 5.1: Position Error Statistics of Loose, Tight and Dual Hypothesis Filters

Measurement Update Rate = 2 Hz					
Loose Integration		Tight Integration		Dual Hypothesis Filter	
σ_x	0.0673 (m)	σ_x	0.0523 (m)	σ_x	0.0523 (m)
σ_y	0.1870 (m)	σ_y	0.0542 (m)	σ_y	0.0549 (m)
σ_z	0.0471 (m)	σ_z	0.0189 (m)	σ_z	0.0190 (m)
Measurement Update Rate = 0.5 Hz					
Loose Integration		Tight Integration		Dual Hypothesis Filter	
σ_x	1.4619 (m)	σ_x	N/A	σ_x	1.5840 (m)
σ_y	2.4123 (m)	σ_y	N/A	σ_y	1.4416 (m)
σ_z	0.4759 (m)	σ_z	N/A	σ_z	0.2525 (m)

Chapter 6

Conclusion and Discussion

This thesis introduced a traditional approach of INS/VISNAV integration architecture and named it as tightly coupled integration architecture. The tight integration approach is an optimal approach in terms of estimation accuracy. However, the disadvantage of this method is that it might experiences instabilities in some unfavorable feature points geometries. This is due to the non convexity of the cost function formed by the pinhole camera model.

We proposed an alternative and named it loosely coupled INS/VISNAV integration as a workaround for the instability issue. The loose integration scheme is sub-optimal from an information fusion point of view. However, because it decouples the vision-based navigation (VISNAV) solution from the inertial sensor solution, its increased robustness allows it to work with low cost inertial sensors in the automotive and consumer grades. In a loose integration it is crucial to have an accurate estimate of the measurement noise covariances. For this purpose a linearized algorithm for calculating VISNAV position and attitude covariances was developed and tested. The covariance estimation assumes the only error source in the VISNAV solution is pixel noise. The results show that without GNSS, the integrated INS/VISNAV solution provides a reasonably accurate

navigation estimates. The loose integration is not a replacement architecture for the tight integration. Rather, it is an alternative mode to use when increased robustness is necessary, as in the case when inertial sensor drift is large and other measurement updates are infrequent or of poor quality. As such, a feature that will be required in a fielded INS/VISNAV system using low cost inertial sensors is a methodology to reconfigure the filter between tight and loose in response to the quality of landmarks and overall quality of the state estimate.

Tight and loose architectures have the advantages which are complementary to each other. Therefore, we proposed a sensor fusion algorithm which can improve the robustness of camera-aided inertial navigation system but still maintaining the accuracy. The approach is based on integrating two camera-aided INS architectures (loose and tight architectures) under a Bayesian framework of hypothesis testing. This framework implements a dual hypothesis testing filter which assigns different weights to the two architectures depending on the likelihood test between the filter output and the camera measurement. The architecture with higher likelihood will be assigned a larger weight and contributes more to the final estimate. This weight is constantly updated at every measurement epoch. This kind of weight assignment is referred to as a "soft-switching" mechanism between different models. It is shown in simulation that the proposed filter enhances the robustness of the camera-aided INS solution relative to a standalone tight integration architecture. It is also shown that the filter can achieve higher estimation accuracy compared to a standalone loosely integrated camera-aided INS. The advantage of the proposed approach is that it only requires slight modification of the existing architectures and therefore minimize the filter redesign effort.

6.1 Future Work

The research doesn't end here. More works can be done in the future. For example, Direct Linear Transformation (DLT) is only one of the algorithms which can solve Perspective-n-Point problem (PnP). Although this method is mathematically straightforward, it has disadvantages such as sensitive to large pixel noises (or outliers) and it cannot generate solutions if the feature points are coplanar. Other PnP algorithms are more robust to pixel noises and capable to deal with planar case. However, the analytical approach for covariance analysis, like what we proposed for DLT, is still underdevelopment. This can be a research direction in the future for replacing DLT with better PnP algorithms in the loose integration.

References

[1] J. Hesch and S. Roumeliotis. A direct least-squares (dls) method for pnp. In *Proceedings of the International Conference on Computer Vision*, Barcelona, Spain, November 6-13 2011.

[2] P. Misra and P. Enge. *Global Positioning System, Signals, Measurements, and Performance*. Ganga-Jamuna Press, 2001.

[3] Scott Gleason and Demoz Gebre-Egziabher. *GNSS Applications and Methods (GNSS Technology and Applications)*. Artech House, 2009.

[4] Paul D. Groves. *Principles of GNSS, Inertial and Multisensor Integrated Navigation Systems*. Artech House, 2008.

[5] Dee Ann Divis and Glen Gibbons. Dod, dot deputy secretaries say no practical solutions exist for lightsquared interference to gps. In *Inside GNSS*, Janurary 2012.

[6] Marcio Aquino, Terry Moore, Alan Dodson, Sam Waugh, Jock Souter, and Fabiano S. Rodrigues. Implications of ionospheric scintillation for gnss users in northern europe. *Journal of Navigation*, 58:241–256, 2005.

[7] Trent A. Skidmore and Frank van Graas. An investigation of tropospheric errors on differential gnss accuracy and integrity. In *Proceedings of the 17th International Technical Meeting of the Satellite Division of The Institute of Navigation*, 2004.

[8] Michael Bendera, Galina Dicka, Maorong Gea, Zhiguo Denga, Jens Wickerta, Hans-Gert Kahlea, Armin Raabeb, and Gerd Tetzlaffb. Development of a gnss water vapour tomography system using algebraic reconstruction techniques. *Advances in Space Research*, 47:1704–1720, 2011.

[9] Ulrich Foelsche and Gottfried Kirchengast. Tropospheric water vapor imaging by combination of ground-based and spaceborne gnss sounding data. *Journal of Geophysical Research: Atmospheres (19842012)*, 106:2722127231, 2001.

[10] Guenther Retscher and Qing Fu. Continuous indoor navigation with rfid and ins. In *Proceedings of IEEE/ION PLANS 2010*.

[11] Guenther Retscher and Qing Fu. Active rfid fingerprinting for indoor positioning. In *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation*, September 16 - 19 2008.

[12] U B Syed and T. Arslan. 3-dimensional approach to wifi indoor positioning. In *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation*, 2011.

[13] M.M. Atia, A. Noureldin, and M.J. Korenberg. Bayesian machine learning in ins/wifi integrated navigation systems for indoor and gnss-denied environments. In *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation*, 2011.

[14] Xing Zhao, Chris Goodall, Zainab Syed, Bruce Wright, and Naser El-Sheimy. Wi-fi assisted multi-sensor personal navigation system for indoor environments. In *Proceedings of the 2010 International Technical Meeting of The Institute of Navigation*, 2010.

[15] Neil Harper and Martin Dawson. Hybrid a-gps and ranging for handset positioning. In *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation*, 2007.

[16] Miao Zhang, Stefan Knedlik, Pakorn Ubolkosold, and Otmar Loffeld. A data fusion approach for improved positioning in gsm networks. In *Proceedings of IEEE/ION PLANS 2006*, 2006.

[17] G.N. Desouza and A.C. Kak. Vision for mobile robot navigation: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):237 –267, feb 2002.

[18] N. Frietsch, O. Meister, C. Schlaile, J. Seibold, and G. F. Trommer. Vision based hovering and landing system for a vtol-mav with geo-localization capabilities. In *Proceedings of AIAA Guidance, Navigation and Control Conference and Exhibit*, page 18.21, August 2008.

[19] V. Indelman, P. Gurfil, E. Rivilin, and H. Rotstein. Real-time mosaic-aided aerial navigation: I. motion estimation. In *Proceedings of AIAA Guidance, Navigation, and Control Conference*, volume 10-13, August 2009.

[20] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein. Real-time mosaic-aided aerial navigation: Ii. sensor fusion. In *Proceedings of AIAA Guidance, Navigation and Control Conference*, 2009.

[21] S. Saeedi, F. Samadzadegan, and N. El-Sheimy. Vision-aided inertial navigation for pose estimation of aerial vehicles. In *Proceedings of the 22nd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2009)*, pages 453–459, September 2009.

[22] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, and et. al. Coupled vision and inertial navigation for pin-point landing. *Journal of Field Robotics -Special Issue on Space Roboti*, 24(5):357–378, April 2007.

[23] R.M. Haralick, H. Joo, C-N Lee, V.G. Vaidya, and M.B. Kim. Pose estimation from corresponding point data. *IEEE Transactions on System, Man, and Cybernetics*, 19(6):1426–1446, Nov/Dec 1989.

[24] D. Mortari, J.M. Rojas, and J.L. Junkins. Attitude and position estimation from vector observations. In *Proceedings of AAS/AIAA Space Flight Mechanics Meeting*, Feb 2004.

[25] J.L. Crassidis, R. Alonso, and J.L. Junkins. Optimal attitude and position determination from line-of-sight measurements. *Journal of Astronautical Sciences*, 48(2-3):391–408, 2000.

[26] D. Nister. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, 2004.

[27] Taragay Oskiper, Zhiwei Zhu, Supun Samarasekera, and Rakesh Kumar. Visual odometry system using multiple stereo cameras and inertial measurement unit. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[28] J.-P. Tardif. Monocular visual odometry in urban environments using an omnidirectional camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*, 2008.

[29] David Nistr, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23:3–20, 2006.

[30] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Proceedgins of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[31] Michael J. Veth. *Fusion of Imaging and Inertial Sensors for Navigation.* PhD thesis, U.S. Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2006.

[32] M. Veth and J. Raquet. Fusion of low-cost imagining and inertial sensors for navigation. In *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, Fort Worth, TX, September 26 - 29 2006.

[33] S. I. Roumeliotis, A.E. Johnson, and J.F. Montgomery. Augmenting inertial navigation with image-based motion estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 02)*, volume 4, pages 4326–4333, 2002.

[34] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, and J.F. Montgomery. Vision-aided inertial navigation for precise planetary landing: Analysis and experiment. In *Proceedings of Robotics System and Science Conference*, 2007.

[35] J. Raquet and M. Giebner. Navigation using optical measurements of objects at unknown locations. In *Proceedings of the 59th Annual Meeting of the Institute of Navigation*, pages 282–290, 2003.

[36] Niklas Karlsson, Enrico Di Bernardo, Jim Ostrowski, Luis Goncalves, Paolo Pirjanian, and Mario E. Munich. The vslam algorithm for robust localization and mapping. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2005.

[37] David Lowe. Object recognition from local scale-invariant features. In *Proceedings of the international Conference on Computer Vision*, 1999.

[38] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[39] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *SURF: Speeded Up Robust Features",*, 110(3):346–359, 2008.

[40] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, 1988.

[41] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2004.

[42] Yi Ma, Stefamp Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models.* Springer, 2004.

[43] Jan J. Koenderink. The structure of images. *Biological Cybernetics*, 50(5):363–370, August 1984.

[44] Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21:224–270, 1994.

[45] R.M. Haralick, C. Lee, K. Ottenberg, and M. Nolle. Analysis and solutions of the three point perspective pose estimation problem. In *Proceedings of IEEE Conf. Computer Vision and Pattern Recognition*, pages 592–598, 1991.

[46] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communication of the ACM*, 24(6):381–395, 1981.

[47] P.H.S. Torr and A. Zisserman. A new robust estimator with application to estimating image geometry. *Journal of Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[48] B. J. Tordoff and D. W. Murray. Guided-mlesac: Faster image transform estimation by using matching priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1523–1535, 2005.

[49] O. Chum and J. Matas. Randomized ransac with td,d test. In *Proceedings of 13th British Machine Vision Conference*, September 2002.

[50] C-P. Lu, G. D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transaction on Pattern Analysis and Machine Intelligent*, 22(6), 2000.

[51] C. McGlove and J. Bethel E. Mikhail. *Manual of photogrametry.* American society for photogrammetry and remote sensing, 2004.

[52] X. S. Gao, X. R. Hou, J. Tang, and H. F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:930–943, 2003.

[53] Long Quan and Zhongdan Lan. Linear n-point camera pose determination. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENC*, 21(7):1–7, July 1999.

[54] Gene H. Golub and Charles F. van Loan. An analysis of the total least squares problem. *SIAM Journal on Numerical Analysis*, 17(6):883–893.

[55] Matthias Muhlich and Rudolf Mester. The role of total least squares in motion analysis. In *Proceedings of European Conference on Computer Vision (ECCV98)*, pages 305–321, 1998.

[56] Vincent Lepetit, Francesc Moreno-noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81(2):155–166, 2009.

[57] A. Kipnis and A. Shamir. Cryptanalysis of the hfe public key cryptosystem by relinearization. In *Proceedings of Advances in cryptology CRYPTO99*, pages 19–30, 1999.

[58] F. Landis Markley. Attitude determination using vector observations and the singular value decomposition. *The Journal of Astronautical Sciences*, 36(3):245–258, 1988.

[59] G. W. Stewart and Ji-Guang Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

[60] T. Papadopoulo and M.I.A. Lourakis. Estimating the jacobian of the singular value decomposition: Theory and applications. In *Proceedings of the European Conference on Computer Vision, ECCV00*, 2000.

[61] Richard I. Hartley. Triangulation. *COMPUTER VISION AND IMAGE UNDERSTANDING*, 68(2):146–157, 1997.

[62] Kenichi Kanatani, Yasuyuki Sugaya, and Hirotaka Niitsuma. Triangulation from two views revisited: Hartley-sturm vs. optimal correction. In *Proceedings of 19th Machine Vision Conference*, pages 173–182, 2008.

[63] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, 2001.

[64] Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, Hoboken, N.J., USA, 2006.

[65] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, New York, New York, 4 edition, 2002.

[66] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control*. Hemisphere Publishing Corp., 1975.

[67] Robert F. Stengel. *Optimal Control and Estimation*, pages 343 – 351. Dover, New York, New York, 1994.

[68] D. G. Kottas, J. A. Hesch, S. L. Bowman, and S. I. Roumeliotis. On the consistency of vision-aided inertial navigation. In *Proceedings International Symposium on Experimental Robotics*, Quebec City, Canada, June 2012.