3-24-2016

Real-Time Implementation of Vision-Aided Monocular Navigation for Small Fixed-Wing Unmanned Aerial Systems

Timothy I. Machin

REAL-TIME IMPLEMENTATION OF VISION-AIDED MONOCULAR NAVIGATION FOR SMALL FIXED-WING UNMANNED AERIAL SYSTEMS

THESIS

Timothy I. Machin, 2d Lt, USAF

AFIT-ENG-MS-16-M-035

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-16-M-035

REAL-TIME IMPLEMENTATION OF VISION-AIDED MONOCULAR NAVIGATION FOR SMALL FIXED-WING UNMANNED AERIAL SYSTEMS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Masters of Science in Electrical Engineering

Timothy I. Machin, B.S.

2d Lt, USAF

March 2016

AFIT-ENG-MS-16-M-035

REAL-TIME IMPLEMENTATION OF VISION-AIDED MONOCULAR NAVIGATION FOR SMALL FIXED-WING UNMANNED AERIAL SYSTEMS

THESIS

Timothy I. Machin, B.S.
2d Lt, USAF

Committee Membership:

Dr. John Raquet
Chair

Dr. David Jacques, PhD
Member

Maj. John Pecarina, PhD
Member

AFIT-ENG-MS-16-M-035

Abstract

The goal of this project was to develop and implement algorithms to demonstrate real-time positioning of a UAV using a monocular camera combined with previously collected orthorectified imagery. Unlike previous tests, this project did not utilize a full inertial navigation system (INS) for attitude, but instead had to rely on the attitude obtained by inexpensive commercial off-the-shelf (COTS) autopilots. The system consisted of primarily COTS components and open-source software, and was flown over Camp Atterbury, IN for a sequence of flight tests in Fall 2015. The system obtained valid solutions over much of the flight path, identifying features in the flight image, matching those features with a database of features, and then solving both the 6DOF solution, and an attitude-aided 3DOF solution. The tests demonstrated that such attitude aiding is beneficial, since the horizontal DRMS of the 6DOF solution was 59m, whereas the 3DOF solution DRMS was 15m. Post processing was done to improve the algorithm to correct for system errors, obtaining a 3DOF solution DRMS of 8.22 meters. Overall, this project increased our understanding of the capabilities and limitations of real-time vision-aided navigation, and demonstrated that such navigation is possible on a relatively small platform with limited computational power.

Acknowledgements

I would like to foremost thank Dr. John Raquet for his guidance and mentorship during the completion of this thesis. Without his direction, I would never have completed the research getting lost down tunnels of "what if". I would also like to thank Don Venable, without whom I would have been lost in the spider web of his PnP framework. Special thanks to Rick Patton and Jeremy Gray, who helped me turn what ideas I had into a functioning, flying experiment. Many late nights, last minute corrections, and frustrating hours were spent on this research, but for all of these hours I could always rely on these wonderful people.

Lastly but most importantly, I would like to thank my wife, whose unfaltering support, love, and scolding helped push me through graduate school. I look forward to every new day with her.

Timothy I. Machin

Table of Contents

Page

List of Figures

List of Abbreviations

Abbreviation		Page
AOI	Area of Interest	35
GCS	Ground Control Station	47
HUD	Heads up Display	47

REAL-TIME IMPLEMENTATION OF VISION-AIDED MONOCULAR NAVIGATION FOR SMALL FIXED-WING UNMANNED AERIAL SYSTEMS

I. Problem Statement

In recent decades, small unmanned aerial systems have become more common across the globe. These systems are used by the military, commercial, and even the hobbyist communities, but all systems contain similar components and perform closely tied basic operations. For autonomous systems, a navigation solution is necessary for localization to determine the vehicles position and orientation. For the vast majority of systems, even those used for military purposes, the primary means of navigation is GPS. This is usually seen as the cheapest, easiest, and most effective solution, but it comes with some disadvantages as well.

1.1 Background

GPS was made available to the public in 1983 in response to the tragedy of Korean Air Lines flight 007, which could have been avoided with a more reliable navigation solution [10]. Since then the use of global navigation satellite systems (GNSS) has spread to almost every aspect of life. Farming, banking, vehicle navigation, even cellular phone navigation and timing have all become dependent on the GPS system. Even for military purposes, GPS is seen as a primary means of navigation for autonomous operations. For military operations, the primary issues with the GPS signal are that the acquisition signal is available to the public and, as such, is easily accessible by inexpensive means and easily jammed or blocked. Though loss of GPS is not as large of an issue for piloting manned aircraft, as the pilot can correct for

poor or false navigation solutions when encountered, it is catastrophic for autonomous unmanned vehicles. One potential solution to these problems is that of vision-aided navigation in its various forms.

GPS vulnerability has been noticed by the Department of Defense, and the USAF in particular. In the Air Force Future Operating Concept for the view of the Air Force in 2035, a primary desire is for "New concepts and capabilities to counter the increasing technology and proliferation of anti-access and area denial threats, to include multi-domain approaches and systems that can be rapidly modified when adversaries adapt their defenses.[11]"

1.2 Definition of Terms

According to Beard and McLain, small unmanned aerial systems (SUAS) or small unmanned aerial vehicles (SUAV) are used to describe "the class of fixed-wing aircraft with a wing span between 5 and 10 feet. Small unmanned aircraft are usually gas powered and typically require a runway for take-off and landing...."[12]. There are many interpretations on how small or large a SUAS can be, but the common threshold is having a wingspan under 15 feet. The aircraft used in this research, though it is slightly larger than Beard and McLain's definition of a SUAV, is still considered a SUAS by many other sources. The DoD Unmanned Systems Integrated Roadmap provides a slightly different definition of Small UAS, as demonstrated in Figure 1. According to this definition, the Telemaster would be a small UAS, in particular a Group 2 small tactical SUAS [1].

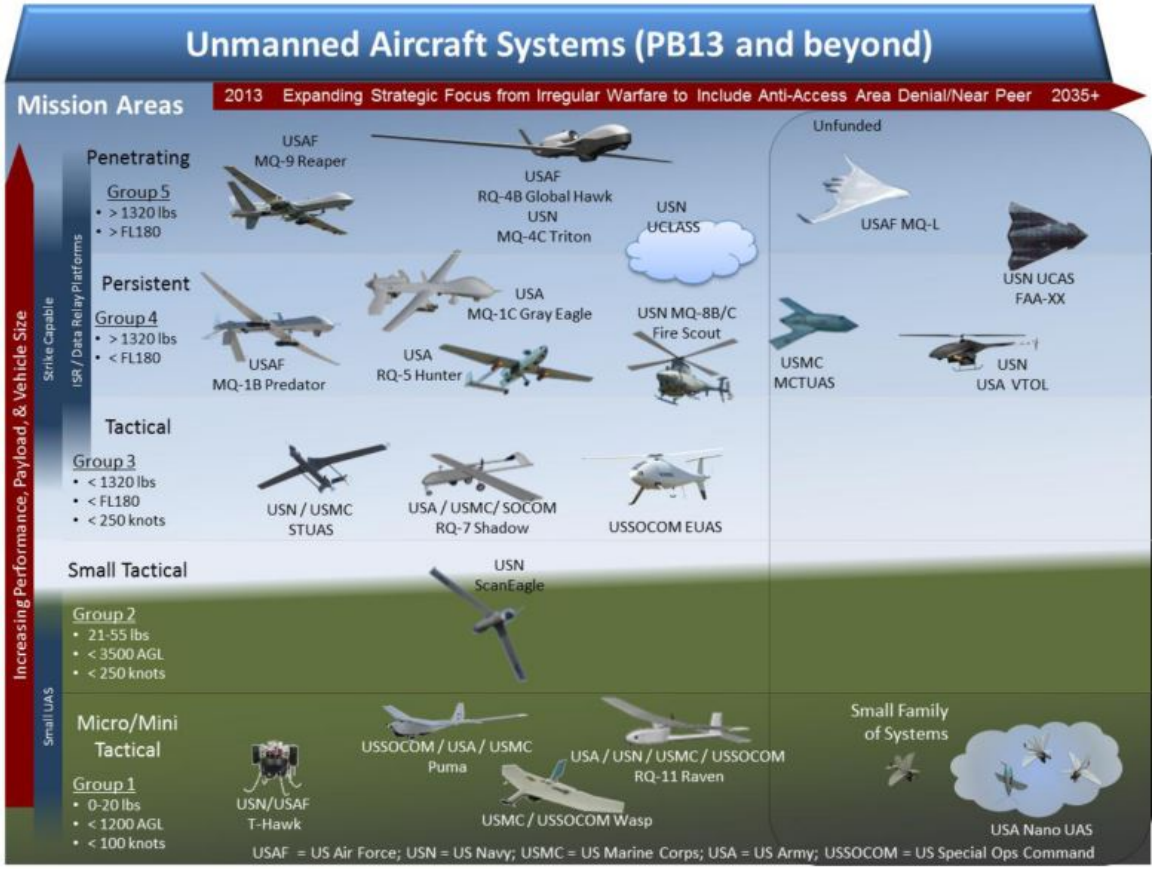

Figure 1. An illustration defining the UAS categories according to the DoD UAS Roadmap [1].

Navigation Warfare (NavWar) is used to define the realm of operations that will utilize the results of this research. This style of warfare refers to either the passive or active denial of specific navigation means in a defined environment, such as GPS jamming, and the alternative way to navigate in such an environment. DoD Instruction 4650.08 describes the integration of NavWar to acquire and secure position, navigation, and timing (PNT) information in the battlefield. The international incidents listed in the Background section are clear examples of how NavWar was used to intercept military aircraft and, as such, is a constant problem for current and future autonomous vehicles.

Vision navigation, or VisNav, is used to describe the use of computer vision to

either navigate an environment by images only or to aid other navigation systems. The VisNav system proposed in this research is a vision-aided navigation system, meaning that the computer vision is only a part contributing to the whole of the navigation solution. This field has been expanding greatly in the last decade in conjunction with the computation power available and the quality of digital imaging— making this solution more suitable for accurate navigation.

Many VisNav systems utilize some form of imagery as a reference. The images used are typically orthophotos or orthoimages: images captured from specified heights and then geometrically corrected so that the scale is uniform. Such an image that has been geometrically corrected is said to be "orthorectified".

Of the many methods that can be used for VisNav, the primary method proposed for use is that of perspective-n-point (PnP). This is a common computer vision technique to determine the pose of the camera from known 3D points and their projections in the image plane[13]. The advantages and disadvantages of this approach will be discussed later in this thesis.

Real-time is used to describe a process or function that can be achieved at the moment the data is gathered, as opposed to the post-processing of data, which must be done after all or large portions of the data has been gathered. In imaging and computer science, real-time is often used to describe the rate at which frames are acquired, typically 30 frames per second (fps). For this research, to avoid confusion, 30fps is not the desired definition of real-time, and in fact the frame rate is only a portion of the measurement rate. For this system, pose calculation will be considered real-time if a solution can be calculated while the vehicle is still performing autonomous flight over the target area. The desired measurement rate, the rate from which the command is given for image acquisition to the final calculation is presented or saved to the on-board computer, should be as close to 5Hz as possible in order to

integrate with the chosen autopilot for the SUAS. The specific rate desired for this research was 1Hz, similar to the rate used in [7].

In the interests of making this solution easily attainable for furthering future research, the SUAS was determined to primarily utilize commercial off the shelf (COTS) equipment and open source software (OSS) in the creation of the VisNav solution. All individual hardware components are available either through common re-sellers, or from the Internet. Open source software and scripts are available for free online and are developed and improved by a community of programmers. The VisNav solution presented in this thesis uses these to build a working system, leveraging the open-source functions for faster development. The building blocks of the system are all COTS/OSS, and the final solution framework can be recreated from OSS and COTS components.

1.3 Problem Statement

For any autonomous vehicle, a form of navigation is required in order for the system to accurately determine its pose at a given moment. The primary downfalls of the GPS navigation solution, as discussed above, are urban canyons, indoor environments, and accidental or purposeful interference. Accidental or purposeful interference can include active jamming or spoofing of the GPS signals, and both fall under the NavWar domain in general. Extensive research has been explored on this subject matter over the last several decades, but an ideal final solution has not yet been discovered. This research aims to solve for a fully implemented VisNav solution using COTS equipment for autonomous, real-time navigation of a SUAS. If this is achieved, what improvements can be made to the system to improve performance?

1.4 Research Objectives

The primary objective of this research, as stated above, is the demonstration of a real-time vision-aided navigation system on-board an autonomous SUAV. In order to achieve this objective, many of the individual parts of the system had to be completed to a sufficient degree first. The algorithm itself had to be either developed or modified from existing PnP solutions to provide accurate results for the given environment and airframe. The integration of the algorithm with the modified airframe is the primary objective for successful implementation, as the modifications to the airframe are what allow the autonomous vision navigation in a real-time scheme. The intended accuracy of the navigation solution is to be that of near-GPS accuracy to validate it as a reasonable, alternative solution in the instance of GPS signal loss or disruption. The current civilian GPS achieves 1-10 meters accuracy by single point positioning, so a solution of approximately 10 meters of error or better is preferred for the final solution [10].

1.5 Research Focus

The overall foci for this research are centered around the final SUAS design and its ability to perform real-time implementation of the PnP solution. The first of these is a real-time solution, meaning the processing capabilities must be of sufficient level to perform image analysis and advanced calculations while still in autonomous flight. The real-time solution is superior to post-processing, which must be done after the flight is completed and all data and images gathered. Post-processing would not provide a solution at the time of flight and does not meet the requirements for implementation with the system, but will be useful in developing and improving the real-time solution.

The second focus is the algorithms used in generating the navigation solution.

The algorithms chosen have been proven to provide accurate results on past projects and as such will be used for implementation of the full system. The majority of these algortithms were developed by Venabale, utilizing an accumulation of open-source software and customized frameworks to calculate a navigation solution based on real-time imagery and a feature database. Though the base of these algorithms use the open source computer vision library (OpenCV) and other OSS packages, these were only leveraged to produce an effective framework for the VisNav algorithms.

The last focus is the implementation of the full system. Implementation was performed on a real SAUS, which integrated the algorithms with the necessary components of the airframe in order to achieve the real-time solution for autonomous flight. Any errors from the individual components, including the algorithms, will be propagated through the implementation and expressed clearly in the final solution errors. These errors are calculated by determining the difference of the aircraft's trajectory compared to its truth trajectory. The integration of the individual components is necessary for the implementation of the full system, but the algorithms necessary to provide the navigation solution from the images is the primary focus and concern, as small errors in the solution can result in larger errors during implementation. The details of the algorithm development will be covered in Chapter 3, as well as how the known risks were mitigated with the solution.

The scope of this research is focused to demonstrate of the capabilities of the SUAS, primarily on the integration of components to obtain the VisNav solution. Extensive studies were not conducted on the individual components, as they were proven to perform similar operations in previous experiments [14]. Most of the individual components were chosen based on availability, and only a small amount of analysis was done for the primary performance characteristics of each component to verify the capabilities to perform as desired.

1.6 Methodology

The goal of this research is to provide a physical implementation of a working system, not just the theory or a simulation of results. The methodology of creating and implementing such a system will be covered in extensive detail in Chapter 3. The methodology of the experiment is broken down into three main categories: Database Development, PnP Algorithm, and Integration and Implementation. The database development is completed pre-processing, meaning that it is done before any flights are conducted. The database begins with reference satellite imagery, ortho-rectified to create a high-resolution orthophoto of the area of interest. The reference imagery is then reduced to feature keypoints by use of the Scale Invariant Feature Transform (SIFT) algorithm [5]. SIFT will be covered in more detail in Chapter 2. These keypoints are then registered in a database for use in the PnP algorithm.

The algorithms and image analysis are performed by an on-board computer. The SUAV will carry an Intel Next Unit of Computing (NUC) during the flights that will perform all of the computations necessary to provide an accurate solution. The algorithm itself was modified from several existing PnP solutions to yield a fast and accurate solution to the real-time navigation problem.

The implementation of the system, aside from any analysis and improvement of the performance, is the final step in the experiment. The system is composed primarily of COTS equipment and algorithms generated from open-source software, making it easier for replicating the research, as well as modifying the solution to further the field of study after this experiment. The full system will consist of a UAV using computer vision, an on-board machine vision camera, on-board image processor, and the pre-generated database of keypoints to autonomously navigate an area of interest. There are other components to the test system, such as a ground station to monitor the progress of the vehicle and NUC, but these ground station components are not

required for the system to function autonomously. More detail will be covered in Chapter 3, with the steps, results, and specifications for these three processes.

1.7 Assumptions

Though the system is designed to work in GPS-denied environments, for testing and safety purposes, the system Autopilot utilized GPS for flight. Autonomous take-off and landing, even without the use of GPS, have already been accomplished, so they are not the primary focus of the research in this thesis [15]. Since this capability has already been demonstrated, it will be assumed that for a full system in the field, similar methods would be used. While using such equipment has many advantages, such as the ability to modify as desired, it also carries with it many disadvantages, which a proprietary system would not have. Most of these disadvantages manifest in the integration of the COTS and OSS components and scripts. For the VisNav pose calculation, the origin of the camera frame is set to be the same as the UAV body frame. In truth there is a translation between these frames, but this was seen as negligible for the overall pose calculation.

Another primary assumption is that the experiments, including all implementation flights, will be conducted in a controlled setting, such as laboratory bench tests or flights at a controlled range and airstrip. No testing will be conducted in an actual GPS-denied environment, and no GPS jamming/spoofing will be implemented. The flights will be conducted in a pre-defined area of interest, limiting the scope necessary for database development.

1.8 Implications

Developing a system that does not rely on GPS will allow for safer air travel for all users currently dependent on the system. VisNav is also important in the

NavWar realm as it removes an easily accessible vulnerability. A working alternative navigation solution implies that aircraft using this solution will be harder to divert using NavWar and more likely to achieve mission objectives. Since VisNav is a passive system, it is not vulnerable to electronic warfare techniques. With a less vulnerable ISR system, more actionable intelligence can be obtained in the field, regardless of GPS denial or electronic warfare counter-operations. This system would increase the capabilities of all UAS, establishing Position, Navigation, and Timing (PNT) and maintaining operational use for PNT information-dependent systems as compliant with DoDI4650.08 [16].

Over recent decades, radio controlled aircraft have grown more common, leading to an increase in the technical maturity of these systems. So much so that the technical gap between military grade SUAS and hobbyist/COTS UAVs is diminishing rapidly [16]. This is especially true in autopilot designs, providing systems with much more versatility and adaptability for performance due to their open source and easily accessible autopilot software. In Ukraine, an air force entirely composed of COTS/hobbyist was used to provide actionable ISR to forces for targeting artillery and provide information on armor and troop movements [17]. This demonstrates the ability for COTS/OSS UAVs to operate in military operations for NavWar, and as such the technical maturity of these systems to be utilized effectively for VisNav research as well. Details on the specific components utilized in this thesis will be covered in Section 3.4.

1.9 Chapter Review

The research and objectives described above will be discussed in more adequate detail in the following chapters, including a review of the current state of research in this field, the methodology and equipment used in this particular research, and

the results obtained from the implementation test flights. Chapter 2 will primarily address the current level of research previously conducted for VisNav solutions of autonomous vehicles, as well as introduce some of the commonly used concepts that will be implemented in this thesis. Chapter 3 will provide a detailed description of how the experiments will be conducted, the data gathered, and the specific equipment required to perform the flight tests for implementing the solution. Chapter 4 will discuss the results obtained from the implementation flight tests and the analysis of said results, and Chapter 5 will provide the conclusions gathered from the analysis, as well as suggestions for furthering this specific research topic.

II. Background

Over the recent decades, there have been many steps taken to solve the problem of navigation in GPS-denied environments using computer vision. Many of these solutions utilize monocular vision [18], whereas in other research a stereo camera system was used to obtain navigation solutions [19]. The primary principles considered in this chapter range from using feature detection methods in images, to optical flow (OF), to visual simultaneous localization and mapping (VSLAM), to the more varied methods of visual odometry (VO). Another method is the use of an *a priori* world model, such as proposed in this research, to obtain a navigation solution using a Perspective-n-Point (PnP) solution. All of these use computer vision techniques and algorithms, but only the methods used in the research will be considered in depth. For the majority of these navigation techniques, a filtering process must also be used, and this too will be discussed briefly in this chapter.

In this chapter, the related methods of vision-based and vision-aided navigation are reviewed, including a discussion of the relevant research in the field. In each of these methods, a feature detection method is used to determine keypoints in the image plane to be used for matching to keypoints of a different image. Various methods of feature detection are discussed later in this chapter, as well as the basics of the estimation techniques used in determining the pose of the vehicle at a given time. The review of related works is used to build a framework for developing the COTS SUAS autonomous vision-aided navigation system.

2.1 Estimation and Filtering

Navigation most commonly entails an estimation of the vehicle pose using measurements. In most navigation schemes, there is uncertainty in the measurements of

the sensors as well as the estimated states of the vehicle. In order to account for these errors, a Bayesian filter is commonly used. The primary goal of this filter is to minimize the error between the true pose of the UAV and the estimated pose. A secondary goal for filters in similar research is to minimize the computational intensity of the filter to promote a real-time or near real-time implementation. The filter used in most navigation schemes is the Kalman Filter (KF), which calculates the optimal estimate in the minimum mean squared error (MMSE) sense. For a standard KF, it is assumed that the dynamics model is linear and Gaussian in nature, and as such the probability distribution can be completely modeled using the mean and covariance.

From the dynamics model, the discrete linear equation for the states is given in Equation 1, and the measurement model is given by Equation 2.

$$\dot{\mathbf{x}}_k = \Phi \mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{G}\mathbf{w}_k \tag{1}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{2}$$

$$\mathbf{Q}_d = E[\mathbf{w}_k \mathbf{w}_k^T] \tag{3}$$

$$\mathbf{R} = E[\mathbf{v}_k \mathbf{v}_k^T] \tag{4}$$

In the equations above, Φ represents the state transition matrix, $\mathbf{B}$ relates the inputs $\mathbf{u}_k$ to the states $\mathbf{x}_k$, and $\mathbf{G}$ relates the process noise $\mathbf{w}_k$ with covariance $\mathbf{Q}_d$ to the states. For the measurement equation, Equation 2, $\mathbf{H}$ relates the measurements to the states, and $\mathbf{v}_k$ is the measurement noise with covariance $\mathbf{R}$. The noise values, $\mathbf{w}_k$ and $\mathbf{v}_k$ represent zero-mean, white, Gaussian noise.

The KF consists of two main operations. The first is the *propagation* step, in which

the estimate and covariance are propagated forward in time to the time of the next measurement. The second is the *update* step, in which the estimate and covariance are adjusted using the new information gained from the current measurement. For each time step forward, the estimate and covariance are propagated using Equations 5 and 6 provided that there are no direct inputs to the system.

$$\hat{\mathbf{x}}_k^- = \Phi \hat{\mathbf{x}}_{k-1}^+ \tag{5}$$

$$\mathbf{P}_{i+1}^- = \Phi \mathbf{P}_i^+ \Phi_i^T + \mathbf{Q}_{d_i} \tag{6}$$

The states and covariance are then updated using what is known as the *Kalman Gain*, given in Equation 7. This is an optimal weighting term used to account for the posterior belief to maintain MMSE optimal conditions [2]. This is then used to update the estimate and covariance solutions, as shown in Equations 8 and 9.

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R})^{-1} \tag{7}$$

$$\hat{\mathbf{x}}_{i+1}^+ = \hat{\mathbf{x}}_{i+1}^- + \mathbf{K}_k [\mathbf{z}(t_{i+1}) - \mathbf{H}_k \hat{\mathbf{x}}_{i+1}^-] \tag{8}$$

$$\mathbf{P}_{i+1}^+ = (\mathbf{I} - \mathbf{K}_{i+1} \mathbf{H}_{x_n(t/t_i),t]}) \mathbf{P}_{i+1}^- \tag{9}$$

In the equations above for the update of the system, $\mathbf{I}$ is the identity matrix. Figure 2 demonstrates the iterative flow of the basic KF.

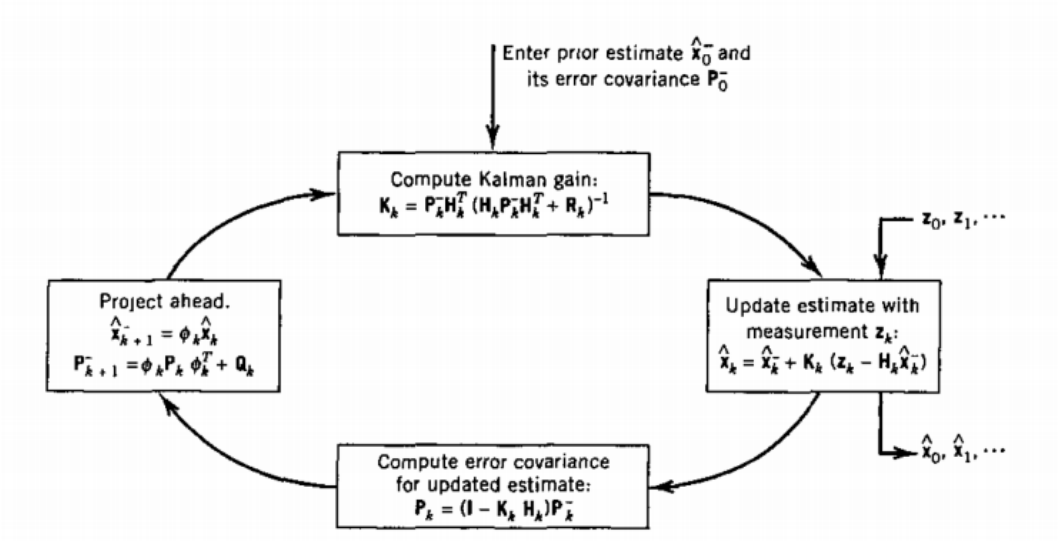

Figure 2. A flow diagram of the process used in basic Kalman Filtering. Includes equations for each step, to be discussed below.[2]

It should be noted that a Kalman filter is only valid for a linear, Gaussian system [20]. Most navigation systems, however, are not linear in nature and as such a Kalman Filter cannot be directly applied. There are many variations to the KF to account for non-linear models. The most common variation is the *Extended Kalman Filter* (EKF), which linearizes the model first before applying it to the KF [20]. Other variations of the KF are the *Unscented Kalman Filter* (UKF), *Extended Information Filter* (EIF), and the *Unscented H∞ Filter* (UHF). A derivation and analysis of each of these filters can be found in [21] when used for the VSLAM solution.

There are many other estimation methods other than the KF, which seek to minimize the difference between the calculated pose and the true pose of the vehicle. One such estimation technique is described in the Perspective-n-Point section of this thesis. Many of the research documents considered below utilized estimation and filtering techniques to develop the VisNav solution.

2.2 Related Research

Though computer vision has been around for decades, its first definitive uses in navigation were for stellar navigation by NASA mars rovers, and has been expanding since [22]. The most common form of vision navigation research is Visual Odometry (VO), which analyzes two consecutive images and determines the motion required to generate the change in images. This technique is used for Optical Flow (OF), VSLAM, and the many other vision-aided and vision-based navigation techniques. Another vision-based navigation technique is structure from motion (SFM), which primarily provides visual construction of the scene rather than calculation of the camera pose. Other techniques exist, such as PnP, that do not require consecutive images in order to calculate the pose of the UAV. For this research, PnP will be the primary method used. Of the many VO techniques, only a few will be addressed briefly in this review. From these cases, a single method will be developed for the vision-aided navigation solution.

Digital Imaging.

All techniques discussed require knowledge of the basics of digital imaging. Figure 3 demonstrates the core functions of digital imaging systems. Computation technologies have advanced significantly since the first digital cameras and feature detection techniques, allowing for use of digital cameras in real-time applications for small, unmanned aerial vehicles. From the digital output from the camera, an array is constructed with each value representing the intensity of light from a specific direction. This array is then used to perform calculations and analysis of the area in the visual spectrum, as well as detecting the features in the image.

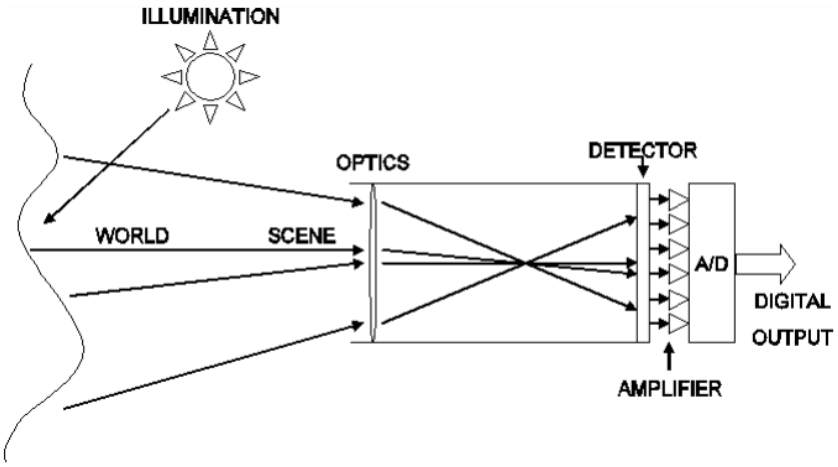

Figure 3. An illustration of the concept behind digital imaging systems. [3]

Feature Detection.

For almost any VisNav system to function, a feature detection method must first be used to identify unique objects that can be found again using the same or a similar process. These objects, or features, are generally used in matching algorithms to determine if two images share any features. There have been three primary feature detection methods used in vision navigation since the 1980s. These are line detection, edge detection, and blob detection [23]. All three of these are discussed in more detail by Lowe [5]. For this research, *Scale Invariant Feature Transform* (SIFT) was determined to be the best choice. Figure 4 details the comparison of several feature detection methods discussed in [4]. It can be seen that compared to the other blob detectors considered, SIFT performed equal to or better than the rest in all regards except efficiency.

	Corner Detector	Blob Detector	Rotation Invariant	Scale Invariant	Affine Invariant	Repeatability	Localization Accuracy	Robustness	Efficiency
Haris	x		x			+++	+++	++	++
Shi-Tomasi	x		x			+++	+++	++	++
FAST	x		x	x		++	++	++	++++
SIFT		x	x	x	x	+++	++	+++	+
SURF		x	x	x	x	+++	++	++	++
CENSURE		x	x	x	x	+++	++	+++	+++

Figure 4. The table is from [4], detailing the performance of several feature detection algorithms.

SIFT was developed in 2004 by Lowe, and is currently viewed as the standard used in many vision navigation techniques[23]. SIFT is completed by performing several operations on the image to deconstruct it to features, or as they are defined in the case of SIFT, keypoints. First, the image is subjected to Gaussian blurring for a range of covariance values, and then the differences taken from each level of blurring to the next. A representation of this process can be seen in Figure 5. In each of these *Difference of Gaussian* (DOG) images, local extrema are found and compared to their eight neighbor pixels. These local extrema are then compared to the DoG octave, and a global extrema found, as can be seen in Figure 6.

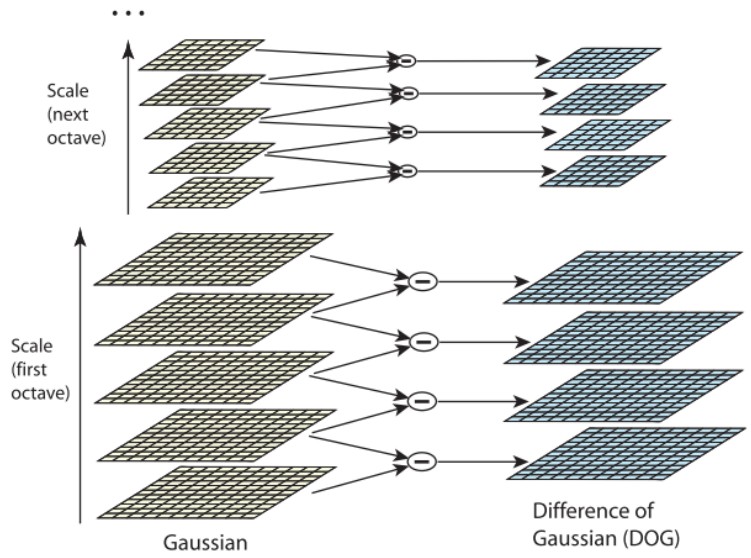

Figure 5. An example of calculating the difference in Gaussian blurs for an image [5]

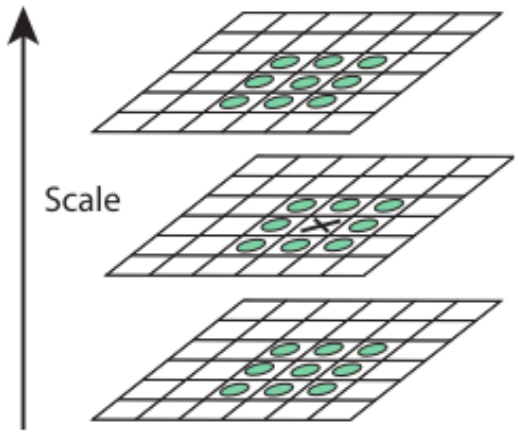

Figure 6. From the DoG calculated, an extrema can be found not only in the image, but also in the scale of DoGs. [5]

Once extrema are found, keypoint localization is used to remove points with low contrast that might be sensitive to noise in the image. During this process, keypoints near or along the edges of the image are also removed. Orientation is then assigned to the extrema to develop keypoint descriptors. An example of this can be seen in

Figure 7 for a 4x4 descriptor taken from a 16x16 sample. The final result is a unique feature that can be used for matching to another image if the feature appears both images. Keypoints are defined by their descriptor, seen in Figure 7, and in addition to the descriptor, a detector containing the following information is associated with each feature:

- Pixel location in $[x_i, y_i]$
- Which Scale the feature is located in
- Orientation of the feature, as defined by the descriptor

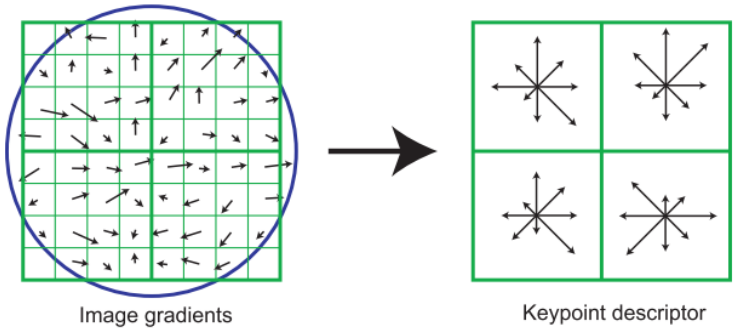

Figure 7. An example of the orientation assigned to a keypoint to generate a unique feature. On the left the magnitude and gradient is calculated around the region of the keypoint. These are then weighted by a Gaussian window, and then accumulated into orientation historgrams, an example of which is on the right. This example shows a $2x2$ array from an $8x8$ whereas the algorithm utilizes a $16x16$ to a $4x4$ array [5].

Though SIFT is commonly used as the feature detection method for VisNav research [24][25][26][27], there have been many variations and alternatives developed. One common alternative is *Speeded-Up Robust Features* (SURF), which was developed in 2008 in the hopes of faster performance than SIFT. SURF has claimed to outperform previously-proposed schemes with respect to repeatability, distinctiveness, and robustness, while still being faster computationally [28]. It was demonstrated that SURF was indeed faster computationally but reduced the accuracy of the estimates

used in analysis [21]. There have been many cases in which SURF was utilized in lieu of SIFT because of its faster computation [26] [29].

SIFT, although proven to be the more computationally intensive feature detection method, provides a larger quantity of unique keypoints for matching and, as such, will be used as the feature detection method for this research. SIFT will be used for deconstructing the images gathered in real-time, as well as the satellite orthophotos that will comprise the pre-generated database of *a priori* information.

Visual Odometry.

As stated in the chapter introduction, VO is the process of determining the structure, or pose, of a vehicle based on the motion from an image taken in one moment to the next. There are many different vision navigation approaches, but the three broad categories considered are pure Visual Odometry (VO), Optical Flow (OF), and Visual Simultaneous Localization and Mapping (VSLAM). OF and VSLAM are examples of varied applications of VO, which use the changes between images to determine change in pose. Each of these will be described briefly below, as well as how they can be utilized for vision-aided and vision-based navigation

Visual Odometry is derived from the same concept as mechanical odometry, in which the current position is determined by integrating the known circumference of the wheel over the amount of time moved. Likewise, for VO, the pose of the vehicle is determined by integrating the change in images over the amount of time elapsed from when the first image was taken to the next [22]. A good survey on the development and uses of VO throughout the last 30 years can be found in [22]. OF and VSLAM are often considered subsets of VO, and as such have many common attributes in their derivation. VO was directly used to obtain a viable and accurate navigation solution by many [7], [30].

A less common VO method for outdoor airborne applications is the use of OF. While OF is used by many animals in nature [27], and has proven useful in indoor applications [31], its uses for pure navigation are limited. The limitations arise primarily from the larger distance between the camera and the objects, requiring a greater change in position of the camera in order to obtain a change in view of the features. A more detailed survey of OF is provided in [27]. Due to the nature of the COTS solution, many of the more sophisticated techniques for vision navigation were discarded due to computation power required for real-time operations.

The last method of VO to be discussed, and more commonly used with todays growing computational power available, is VSLAM. This approach is described in detail by Durrant-Whyte and Bailey in [6],[32]. It was also used in an indoor environment [33]. In this approach, the localization of the aircraft (pose) is estimated at the same time a database is constructed from the images and features seen by the onboard digital imaging system. This database, or map, is used for correlation with images taken at a later time during flight. An analysis of SFM, as well as a comparison of VO and VSLAM was done at the University of Johannesburg South Africa in 2011 [34]. This gives a brief discussion of the various forms of VisNav using stereo cameras, whereas more generic comparisons and analysis of the methods are available through the VO tutorials mentioned above [22]. Figure 8 demonstrates the basic fundamentals of using VSLAM for an airborne UAV.

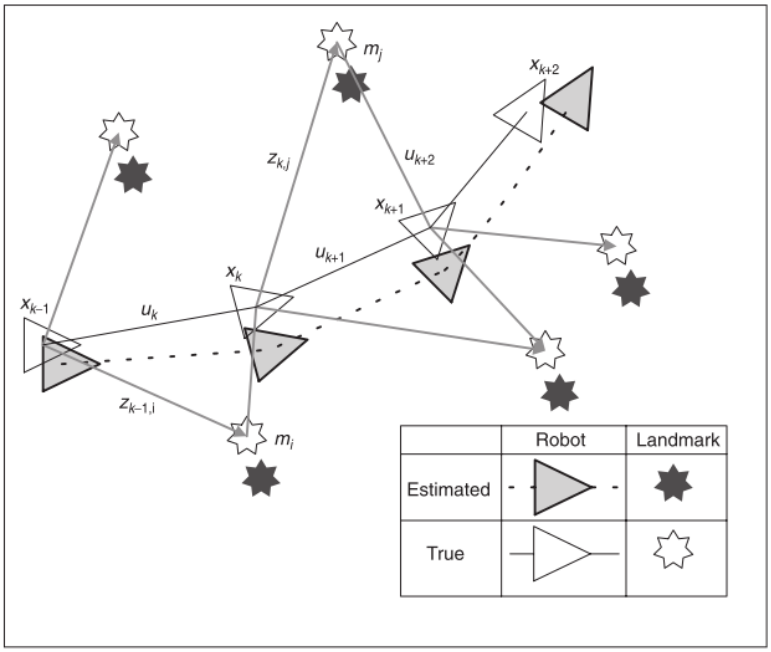

Figure 8. A simple example of the premise of VSLAM. The position is estimated from the visible features, and updated as the UAV progresses. It can be seen how this is typically close to the truth solution, but still contains errors. This image was borrowed from [6].

VSLAM has been frequently used to obtain vision navigation solutions of autonomous vehicles [24], and has been proven capable of near-real-time implementation under limited circumstances as well [35]. The primary components that make up the VSLAM solution are the localization, or pose estimation, and the mapping, or database generation [36]. Research has been focused on one specific attribute of the solution, such as the use of feature detection used for image stitching and mapping [15], or on the total implementation of the entire method [37]. The VSLAM algorithms are most useful when there is no *a priori* information available about the world, such as the feature database used for this thesis [32]. However, it has been seen that with information available, a vision-only navigation solution is available that does not depend on any sensors other than the monocular or stereo cameras used onboard the UAV [38].

Though VSLAM is commonly used as the vision-based and vision-aided naviga-

tion solution in most research applications [34][36][24][39], it can be computationally expensive. It is based on stochastically determining the pose of the vehicle while building a database of reference imagery for later correlation. This method provides results with sufficient accuracy, especially for post-processed data from flights [18]. Due to the relative nature of VO and VSLAM initialization and calculation, an absolute pose cannot be calculated using these methods [32]. Given that a pure SLAM technique cannot determine the absolute position of a vehicle, it was not considered for this thesis.

Perspective-n-Point.

The proposed method of navigation for this research is the PnP algorithm, as it is computationally simple, obtains an absolute position solution, and is readily available through open source computer vision libraries. All three of these were equally important in the decision to use PnP as the VisNav soltuion for this research. PnP is based on the concept of Bundle Adjustment, and it is described in detail in [40]. Exact and approximate solutions are available for the PnP problem [41], as well as analytic solutions[42]. This problem has been addressed for decades, and as such, has many solutions available for use. A common use for the PnP solution is the camera calibration process [43], which is used in all vision navigation systems to obtain the camera calibration matrix. This matrix consists of the focal length, and necessary corrections for lens distortions [23]. Figure 9 shows the PnP process used for this research, relating the 3D locations of the features in the world frame to the 2D location of the feature in the image frame.

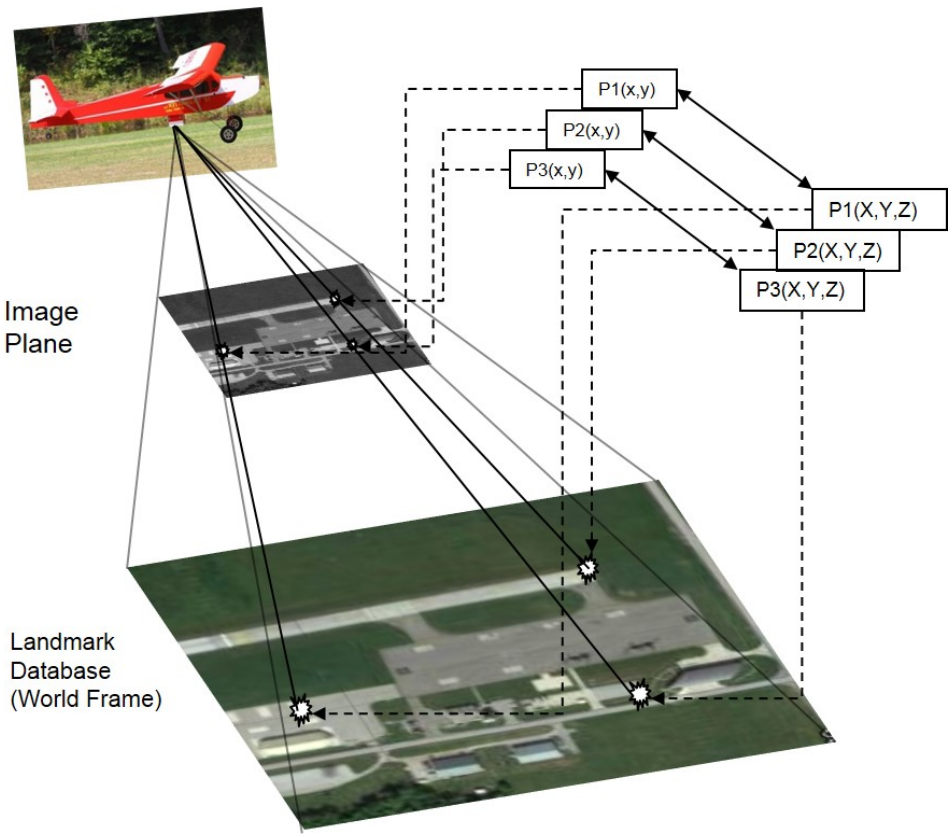

Figure 9. An example of the Perspective-3-Point problem. The pose of the UAV is located using the correlation between the known 3D feature locations of the world frame and the 2D locations of the image frame.

The PnP solutions uses n known points to relate the pixel image locations to known world locations. This technique was considered as it only requires the single image taken and the database of features to compare to. This means that the position calculated at any given time is independent of the previous or future position estimates. This is useful when developing a filter for the overall system. The final PnP implementation used in this research was developed by Donald Venable at AFIT based on the works of [40].

PnP provides an estimate for both the rotation of the camera in the world frame, $\mathbf{R}_c^\omega$, and the translation of the camera in the world frame $\mathbf{t}^\omega$ at the time the image is taken. These are calculated using known world positions of features $\mathbf{X}^\omega$ and measurements $\mathbf{z}$ of the location features in the image frame. Using the camera cal-

ibration matrix $\mathbf{K}$, the projection model is calculated in Equation 10, showing the pixel location of the projection in the image plane [40].

$$h(\mathbf{K}, \mathbf{R}_{\omega}^{c}, \mathbf{X}^{\omega}, \mathbf{t}^{\omega}) = \mathbf{x}^{i} = \mathbf{K}[\mathbf{R}_{\omega}^{c}\mathbf{X}^{\omega} + \mathbf{R}_{\omega}^{c}\mathbf{t}^{\omega}] \tag{10}$$

To formulate the estimation for the rotation and translation, a non-linear optimization strategy is used to minimize a cost function, $\mathbf{F}(\mathbf{R}_{\omega}^{c}, \mathbf{t}^{\omega})$. This cost function is defined as the difference between the measured feature location in the image $\mathbf{z}_k$ and the projected features from the projection model given in Equation 10.

$$\mathbf{F}(\mathbf{R}_{\omega}^{c}, \mathbf{t}^{\omega}) = \mathbf{z}_k - \mathbf{x}^{i} \tag{11}$$

The goal of the PnP algorithm is to estimate the pose $(\mathbf{R}_{\omega}^{c}, \mathbf{t}^{\omega})$ of the vehicle which minimizes the L2-norm of the cost function.

$$\hat{\mathbf{R}}_{\omega}^{c}, \hat{\mathbf{t}}^{\omega} = \underset{\mathbf{R}_{\omega}^{c}, \mathbf{t}^{\omega}}{\operatorname{argmin}} ||\mathbf{F}(\mathbf{R}_{\omega}^{c}, \mathbf{t}^{\omega})||_2^2 \tag{12}$$

In general, the minimization in Equation 12 is non-linear. The function is linearized around the current state estimate, and the error in the states $(\Delta\mathbf{x})$ is estimated. This change in state estimate is then used to update the state estimate, and the process is repeated until it converges on a solution. The linear approximation for this is calculated using a 2nd order Taylor series expansion, provided in Equation 13.

$$F(\mathbf{x} + \Delta\mathbf{x}) \approx F(\mathbf{x}) + \mathbf{g}^{T}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^{T}\mathbf{H}\Delta\mathbf{x} \tag{13}$$

In Equation 13, there were several characteristics calculated in order to obtain the linearized approximation. The Jacobian Matrix, $\mathbf{J}$ is the partial derivative of the cost function with respect to the estimated parameters:

$$\mathbf{J} = \frac{d\mathbf{F}}{d\mathbf{x}} \tag{14}$$

The gradient vector, $\mathbf{g}$, is defined as $\mathbf{J}(\mathbf{x})^T\mathbf{F}(\mathbf{x})$, and is the direction of the greatest rate of change in the parameter space. The Hessian matrix, $\mathbf{H}$, is the second derivative of the cost function. There are two approaches to iterating the approximation to determine a final solution. The first approach is the pure calculus approach, in which the derivative of Equation 13 is set to zero in order to determine the value for $(\Delta\mathbf{x})$. This process is defined as [40]:

1. Set the derivative to zero: $\frac{df}{dx}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{H}\Delta\mathbf{x} + \mathbf{g} = 0$
2. Use this to develop the **Newton Step**: $\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{g}$
3. Update the function: $F(\mathbf{x} + \Delta\mathbf{x}) \approx F(\mathbf{x}) - \frac{1}{2}\Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x} = F(\mathbf{x}) - \frac{1}{2}\mathbf{g}^T\mathbf{H}^{-1}\mathbf{G}$
4. Iteration of this process gives **Newtons Method**

This approach will yield a correct solution, but it does not usually work due to the nature of the Hessian matrix. $\mathbf{H}$ may not be invertible or full rank, which can cause the method to break down, or the approach to never reach a meaningful convergence. One solution to this problem was developed by [44], iterating between the Gausss Newton and the gradient descent to reach a minimum of the function. This is done by introducing a Tikhonov regularization component, which changes the problem to:

$$\Big(\mathbf{J}(\mathbf{x})^T\mathbf{J}(\mathbf{x}) + \lambda\mathbf{D}(\mathbf{x})^T\mathbf{D}(\mathbf{x})\Big)\Delta\mathbf{x} = -\mathbf{g} \tag{15}$$

where $\mathbf{D}(\mathbf{x})$ is the scaling matrix defined below. While some of the higher order terms are lost in this process, an approximation for $\mathbf{H}$ is made using the Gauss-newton approximation given in Equation 16. This is based on an assumption that

the second-order approximation is either roughly linear, or that the resulting errors are relatively small.

$$\mathbf{H} = \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) \tag{16}$$

In Equation 15, the $\mathbf{D}(\mathbf{x})$ term represents a scaling matrix, which is typically the square root of the diagonals of the Hessian matrix. For the LevenbergMarquardt algorithm (LMA or LM), the following process was used to determine the solution.

1. Set and Average Hessian matrix value: $\mathbf{H}_{avg} = \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) + \lambda \mathbf{D}(\mathbf{x})^T \mathbf{D}(\mathbf{x})$
2. Solve for $\Delta \mathbf{x} = -\mathbf{H}_{avg}^{-1} \mathbf{g}$
3. Update the state vector as previously done. Methods addressed in [40] are used to account for the possibilities of internal constraints for states such as the rotation.
4. Adjust the damping factor, λ based on the behavior of the cost function in Equation 11. This will allow for a more accurate estimate, if given the proper termination criteria.

From this second process, a more reliable solution can be computed in a timely manner. For the actual computation of this, OpenCV has a built-in function that has been optimized for speed as well as accuracy. This OpenCV solution utilizes RANSAC for outlier elimination, allowing for adjustable parameters to expand or narrow down the solutions and accepted results [45].

2.3 Analysis of Similar Research

While reviewing literature in this field of research, several articles were found that use a process similar to the research conducted in this thesis, the foremost being the work done by Chiu [7] and Samadzadegan [8]. Chiu and Samadzadegan both

utilized a pre-generated database of images and keypoints referenced with geo-spatial information similar to the one utilized in this thesis. Unlike many of the previously discussed works, no VSLAM or VO method was utilized in either of these research documents.

The first research document mentioned, [7], estimated the 3D absolute pose using an inertial measuring unit (IMU) and two visual measurements with a pre-processed reference image. The primary commonality is the feature database, created from reference satellite/aerial imagery rendered from a 3D terrain model. This is similar to the feature database proposed for the research detailed in this thesis. The second commonality is the 2D-3D tie points used to define the navigation states for Chiu's research. These tie points create the 3D absolute pose from successful geo-registered features from the reference image. Their second visual measurement was created from geo-registering feature tracks, which created 3D reference features if these features were shared between consecutive video frames. They performed 2 test scenarios, similar to the test scenarios proposed in Chapter 3 of this thesis. The first scenario consisted of acquiring flight images from urban and forested environments, whereas the second scenario focused on primarily urban environments. Figure 10 shows the process used to gather video images, compare frames to the geo-referenced database to develop the 2D-3D geo-registration, and apply these and the IMU data to the inference engine to develop the final 3D absolute pose solution.

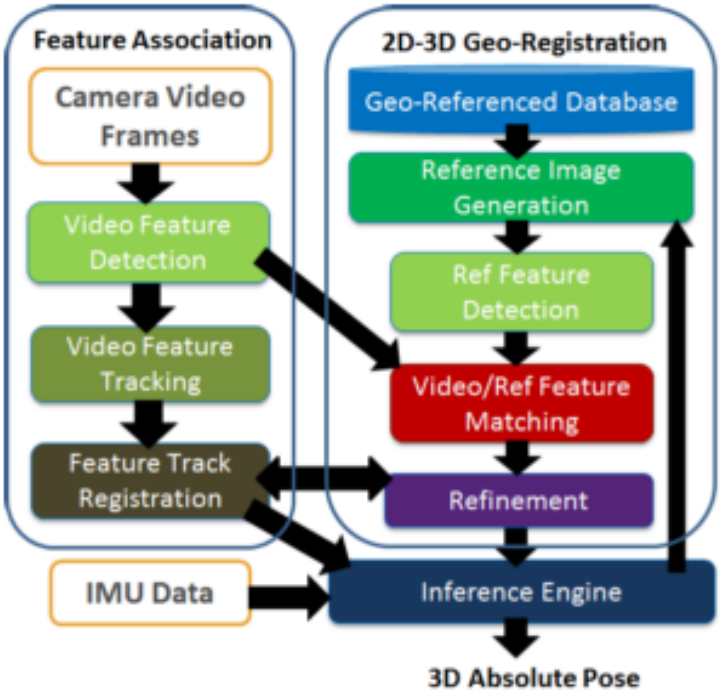

Figure 10. The method used by Chiu and others to gather images for processing to develop a 3D pose estimation of the UAV [7]

Though the research proposed by Chiu has many similarities to the research proposed in this thesis, there were a few key differences as well. One of the main focuses of Chiu's work was the implementation of the complex filtering scheme, the factor graph sliding-window. This greatly increased the complexity of the navigation solution, but it also greatly improved the performance of the system. Their test system was also of a much higher quality, containing propriety systems, algorithms, and components such as the IMU, iSAM2 smoother, and RTK differential GPS system. These higher quality components greatly outperform COTS components in most regards, and introduce less errors into the final navigation solution. Lastly, the aircraft flown for the flight tests was piloted rather than the UAV proposed for this thesis.

The results obtained by Chiu were promising, providing an absolute pose estimate with 3D distance root-mean squared (DRMS) error of 9.83 meters when the 2D-3D tie points were applied to the filtering method for the scenario containing urban and

forested environments. When applied to the purely urban environment, a 3D DRMS error of 9.35 meters was achieved. These results are promising, and demonstrate the capabilities of a much more complex system to provide an accurate VisNav solution for real-time calculations.

The second work considered was done by Samadzadegan [8] in 2011. The proposed method from this research is nearly identical to the method proposed for this thesis, shown in Figure 11.

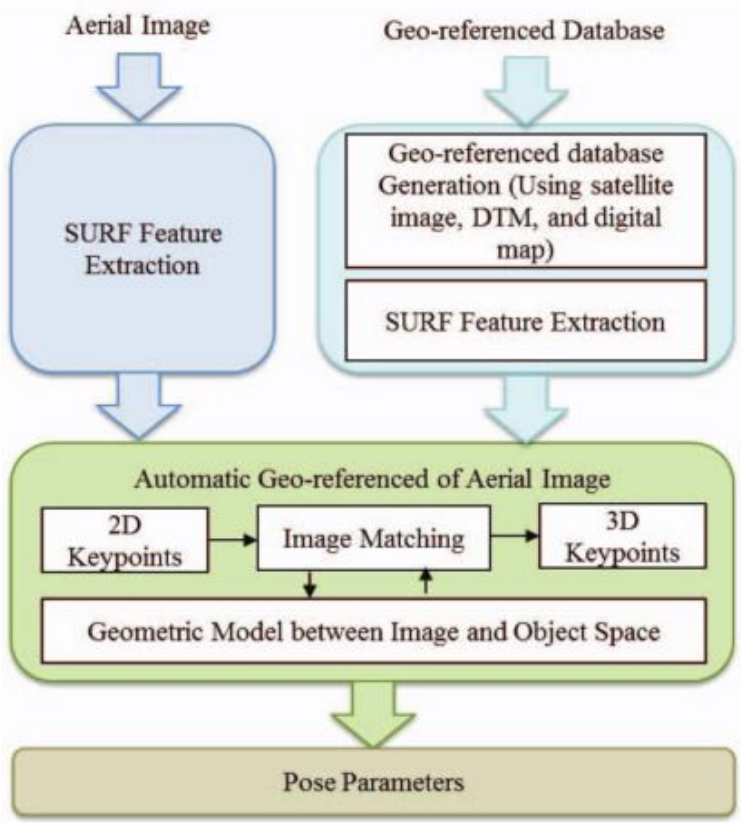

Figure 11. The method used by Samadzadgen to gather images for processing to develop a 3D pose estimation of the UAV given a pre-processed feature database [8]

For the above method, the geo-referenced database developed during pre-processing was nearly the same as the feature database developed in the Chiu research as well as the research proposed in this thesis. The matching process is also similar, utilizing a nearest neighbors scheme to match features, and RANSAC to remove outliers from

matching. This is where the commonalities end for this research unfortunately.

The method used during flight was not a common VisNav solution, and was not covered in sufficient depth for replication. The tests performed were only simulations and not in fact real-time test flights. The geo-registration process was used in lieu of the more formal PnP process, which led to a more poorly defined solution derivation. The primary difference in methods from the Samadzadegan research and the research proposed in this thesis is the feature algorithm utilized to find keypoints in the images. The Samadzadegan research used the SURF algorithm rather than the SIFT algorithm. The research boasted a 5 meter pose error and a 1 degree attitude error using this method. These results are seen as the ideal solution. The research presented in this thesis aims to have the same or greater accuracy, while being applied to a real-time autonomous COTS SUAS.

Extensive research has been conducted in the field of visual navigation for post-processed data [46], as well as for simulated results [8]. For the work using simulated results, the accuracy of the method proposed by Abdi is comparable with GPS/IMU system.[8] Another work using similar processes was conducted by Duo-Yu Gu in 2010, which verified the proposed algorithms ability to solve VisNav problems for simulated data [47]. Even with the amount of research dedicated to this navigation problem in recent decades, a full implementation of real-time autonomous navigation using monocular vision for a COTS SUAS has not yet become a common method for testing navigation solutions. Many solutions utilized COTS equipment, but not for real-time or near real-time analysis. Many others only used simulations, but this research aims to fill this gap in the field.

2.4 Chapter Summary

Though there are many VisNav solutions available, the algorithm decided upon for this research was the Perspective n Point algorithm as it provides an adequate solution for a simple system, and does not require a framework to be established for calculating between consecutive images. Using similar processes to PnP, a VisNav solution of less than 10 meters was proven to be possible, and as such this research strives to produce a comparable solution. With the rising technological level of COTS and OSS components for UAVs, a system can be constructed to implement VisNav solutions for a small unmanned aerial system with little cost compared to implementation on a full sized piloted aircraft. The primary focus of the research will be on the vision-aided navigation solution and the implementation of these navigation solutions, which includes the integration of the COTS equipment. This will be done by refining the PnP solution to work with the COTS SUAS, and to develop test strategies to prove the validity of the system for VisNav application in GPS-denied environments. These, as well as the test flights and procedures are detailed in the next chapter.

III. Methodology

3.1 Chapter Introduction

The overall objective of this research it to determine the accuracy of the COTS vision-aided navigation system and compare it to the accuracy of a GPS navigation solution for real-time use. In order to achieve this objective, information must be collected on the test area of interest, the algorithms developed in order to utilize this information, and the hardware integrated with the data and software. A series of ground and flight tests were used to verify the integration of the system components as well as to verify the accuracy of the system at different stages of development. When proven capable, the system was demonstrated using near real-time operations. Key information for analysis included the true location of the UAV during flights, the telemetry of the aircraft, the database of *a priori* information, and images obtained from the downward facing camera on the aircraft. The information was obtained from images and log files from past flights, data available on the internet, as well as images and 3D telemetry from the autopilot and payload during flight used for real-time applications. The various system components, as well as their integration, will be discussed in detail in this chapter, including how these will be used to meet the test objectives.

3.2 Database Development

In order for the PnP solution to work, a database of *a priori* information is required. This database is a HDF5 [48] patterned structure of 3D geo-located SIFT keypoints and descriptors from satellite imagery, terrain elevation data, and other Geographic Information Systems such as geoid shift files. For the database, a varying range of satellite and aerial imagery is desired in order to verify the results for a

range of pre-processed information. The main sources considered for imagery are GoogleMaps [49] and DigitalGlobe Imagery[9].

Of these sources chosen, the GoogleMaps imagery is completely open-source and available on the open internet, but the DigitalGlobe imagery requires special permissions and/or purchase. GoogleMaps utilizes DigitalGlobe at several zoom levels, but the high resolution imagery from DigitalGlobe is not available directly from Google. There is varying quality of the imagery gathered from each source as well, as described by the ground pixel density (GPD). GPD is defined as the number of image pixels per real world area. Of the two sources, DigitalGlobe provided a greater range of high resolution orthophotos with more reliable, higher GPD. This means that the detail of this imagery theoretically provides the most keypoints when the image is analyzed. The orthophotos are downloaded as GeoTiff open file geospatial information formats when possible, and if not, then as JPEGs or Tiffs, and the geospatial data is added to them afterward. This process will be discussed in the next chapter in the Database Variants section in greater detail. .

The database was constructed using the HDF5 format, with the database consisting of just the area of interest (AOI). For this research, the AOI was only a 3km by 1.72km area, limiting the size of the database to reduce the scope of the database development. The amount of data used for the feature database was miniscule to the storage capacity of the system, and did not effect the computation time of the system, even for smaller processors like the NUC. This orthophoto was then split to smaller regions, or tiles, which are then stored in the database so that only a single tile must be loaded into local memory at a time for use in the PnP calculation. Figure 12 shows the image used for the database referenced in the Flight Tests, which contains the primary location for flight tests and the secondary location as well.

Figure 12. The location of all Flight Tests will be Himsel Airfield located at Camp Atterbury, IN. The area shown is the entire AOI used for the database of geo-located keypoints. A high resolution version of this image was used for the database orthophoto. The image is from DigitalGlobe [9] and requires purchase for full access.

After the orthophoto class is created in the HDF5 format, the SIFT feature descriptors were added as well. The orthophoto is split into strips and passed into the SIFT algorithm. The WGS-84 position of each keypoint is calculated and added to the database along with the DTED-1 vertical datum and the EGM-96 ellipsoid-geoid shift values. For these keypoints, only the top 1000 were kept per strip in order to reduce the overall size of the database. This particular database structure was decided upon to allow for real-time access to the data, limiting the latency of the system for determining the geo-location of the UAV. The algorithm that utilizes this database is described in the following section.

3.3 Algorithm Development

All algorithms developed for the PnP solution were based on the solutions developed by Venable. The algorithm framework for the solution utilized OSS for solving basic operations, but was heavily modified to provide the VisNav solution in the preferred format. Similar solution algorithms can be developed using OSS and the

methods described below. The image processing unit of the system, to be described in section 4 of this chapter, runs a Linux operating system (OS), and utilizes primarily Python for acquisition and analysis of the images. A repository of Python packages called Anaconda [50] is utilized as it contains the majority of packages necessary for computing the PnP solution. Anaconda includes numpy, matplotlib, and many other commonly used packages. One of the critical packages included is OpenCV, an open source computer vision package commonly used for a wide array of computer vision processes, including finding keypoints and matching images [45]. A version of SIFT is included in OpenCV, but as it is open source, the specific SIFT scripts can vary from the source of the OpenCV package. Also included in most OpenCV packages are scripts for image matching, camera calibration, etc; all of these scripts are modifiable and will be customized as required to obtain a functioning algorithm for 3D geo-location of the UAV.

Image Matching.

In order to reduce the number of points analyzed in the algorithm, and thus reduce the computation time, the image from the UAV is matched to a tile in the region of interest. Many matching algorithms and scripts currently exist for python and MATLAB, and most use some form of image decomposition to match edges, lines, or keypoints. The images from the aircraft are decomposed using SIFT, and the keypoints used for matching the image to the specific keypoints in the tile. This is the first method to narrow in on the geo-location of the UAV from the region of interest in a single panel.

There were two methods considered for image matching in this project. The first method is known as the "Brute Force Matcher"(BFM) and is, as the name suggests, the brute force method of matching a single keypoint descriptor at a time. The

feature descriptor from the image is compared to each descriptor in the database for the specific image being matched. The second matching method, and the one used for the results presented in this thesis, is the Fast Library for Approximate Nearest Neighbors (FLANN) [51]. This matcher is optimized for finding matches in large datasets, and as such is ideal for this research. Eventually the image database will be too large for the use of BFM to be practical, when larger AOIs are utilized.

Attitude-Aided PnP Calculation.

The six DOF solution for PnP was shown in the previous chapter. Though this calculation is ideal for an aircraft using just vision navigation, this particular research utilized a COTS autopilot as well that fed the attitude solutions to the image processor for computation. Since the PnP solution is attitude-aided, only the 3 DOF solution for translation must be calculated. The 6 DOF solution is calculated using an implementation of an OpenCV algorithm, and then this solution is used to initialize the 3 DOF implementation. The process for calculating the translation only PnP is very similar to that of the 6 DOF solution, with only minor adjustments. The calculation of the attitude-aided solution starts with the un-normalized Jacobian, shown below [40].

$$\frac{\delta F_j(x)}{\delta \mathbf{t}^\omega} = \frac{\delta}{\delta \mathbf{t}^\omega}(-\mathbf{K}\mathbf{R}_\omega^c t^\omega) = \mathbf{k}\mathbf{R}_\omega^c \tag{17}$$

In Equation 17, the K is the camera matrix, $\mathbf{R}_\omega^c$ is the rotation from the world frame to the camera frame, and t^ω is the translation of the from the reference location to the calculated position of the vehicle expressed in the world frame. From this, the 3-vector $[x^i, y^i, z^i]^T$ must be normalized for the z^i component to find 2-dimensional measurement $\mathbf{z}_k$. In order to normalize the solution, the quotient rule is applied, as seen in Equation 19.

$$F(x) = g(x)/h(x) \tag{18}$$

$$\partial F(x)/\partial x = \frac{g'h - h'g}{h^2} \tag{19}$$

In Equation 19 the multiplication is element-wise, not matrix multiplication. From the PnP model, the following decomposition is given:

$$g = [\mathbf{z}]_{xy} = [x_k, y_k]^T, \text{2x1 vector} \tag{20}$$

$$h = [\mathbf{z}]_z = [z_k]^T, \text{ 1x1 scalar} \tag{21}$$

$$g' = [-\mathbf{KR}^c_\omega]_{rows[1:2]}, \text{2x3 matrix} \tag{22}$$

$$h' = [-\mathbf{KR}^c_\omega]_{rows[3]}, \text{ 1x3 matrix} \tag{23}$$

3.4 Implementation

The implementation of the system proved to be a large piece in the scope of this research, as much of the hardware and software used were either COTS or OSS, and as such were not specifically designed to work together. In some instances, especially for OSS, integration was not guaranteed due to improper operating systems or just incompatible software connections. Work was done to customize these components to communicate together in order to utilize the algorithms and database to provide a VisNav solution to the system. Most components are COTS or OSS other than the specific PnP algorithms and the DigitalGlobe High resolution imagery. Each component will be covered in detail as well as the integration of the components to form the functioning SUAS.

Airframe.

The airframe selected for this research was the 12 Foot Telemaster with Drop Box, as shown in Figure 13. This particular aircraft was chosen for its size, stability, and customizable payload box with an open bottom. This allows for the payload box to be adjusted to easily allow movement of components as needed, as well as the addition of other components if required.

Figure 13. The 12ft Telemaster, used as the platform for this research.

This particular aircraft has an overall length of 90inches from nose to tail, a wingspan of 12 feet, an empty weight of 31lbs, and a wing area of 3050 square inches. Normally for hobbyist UAVs, the wings are the primary source of lift, and the tail is typically a symmetric airfoil. The Telemaster, however, has a lifting tail to aid its 12 foot high lift wing, allowing the UAV to easily carry payloads up to 20 pounds. Technical specifications for this airframe can be found at [52]. The particular aircraft used by the AFIT ANT center is powered by an 85cc 2 stroke gasoline engine.

Autopilot.

For autonomously controlling the UAV, a COTS autopilot was chosen that would allow for possibility of autonomous flight using vision-aided navigation. The Pixhawk

Autopilot was determined to give the best control of flight parameters and modifiable flight software, running the Arduplane 3.3.0 firmware. Details on the autopilot can be found at [53] as well as the many contributors to the open source packages used by the autopilot. The autopilot can be seen in Figure 14 below.

Figure 14. The Pixhawk autopilot is used as the primary autonomous control of the aircraft.

One of the reasons this autopilot was chosen was that the firmware includes a built-in Extended Kalman Filter (EKF) for updating position and orientation. Many COTS autopilots utilize a filter, but it is not always available in the autopilot firmware for modification. This means that the position solution and telemetry of the UAV will be more accurate than that of other autopilots that do not have this capability. With the primary focus of this research being to implement the system in a simulated GPS denied environment and test the accuracy, an autopilot allowing for filter parameter modification increases the number of factors available for improving the accuracy. The computation time for the firmware on the autopilot to run the EKF is negligible, demonstrating the ease of running an EKF on most platforms.

The other justification for choosing this autopilot is its many built in redundancies. It has redundant power supply inputs and automatic failover for safer use in case of power outages or shorts in the system. It also has multiple ports for accessing the telemetry of the aircraft, as well as multiple ports for GPS or navigation solution

input. Particular interest was given to the redundant telemetry ports, allowing roll, pitch, and yaw solutions to be read directly from the autopilot. These pieces of information, as well as others from the telemetry, are used in the algorithm solution for the vision-aided navigation.

The autopilot has a primary GPS port, and a redundant serial port often used for a secondary GPS unit. A primary GPS unit will be maintained for the aircraft during all flights, as for a traditional aircraft, but the second port for the redundant receiver will be used as the input for the vision-based solution, mimicking a pseudo-GPS solution. This connection was intended for the Flight Tests discussed in Section 3.5, but not utilized for this thesis. The autopilot is programmed to switch between the two geo-location solutions based upon the number of visible satellites, and this functionality will be taken advantage of by the on-board computer to force the autopilot to use the desired solution at any given time. If the VisNav solution is deemed accurate enough, the pseudo-GPS solution from the NUC will set the number of visible satellites greater than the GPS receiver while in flight. The level of accuracy required for this transition is a solution within the same magnitude of the GPS receiver or better.

Image Processor.

The on-board computer is used as the image processor for determining the pose of the UAV and sending the geo-location to the autopilot as well was the primary computation engine for determining the VisNav solution. Although the computer performs more operations than just image processing, the processing and pose calculation are the primary functions of this component component. Secondary functions include telemetry and data storage of in-flight calculations and observations. The on-board processor used for this task is an Intel Next Unit of Computing (NUC) D53427RKE [54]. This is a mini computer including an i5 processor and 8GB RAM,

and will not only analyze the images captured and output the geo-location solution, but will also store the images and input telemetry on the 500GB SSD for any post-processing necessary. The NUC runs a Linux OS Ubuntu 14.04. The algorithms run on the NUC are primarily Python scripts, utilizing the many packages described in Section 3.3. More specific technical details of the NUC are available at [54].

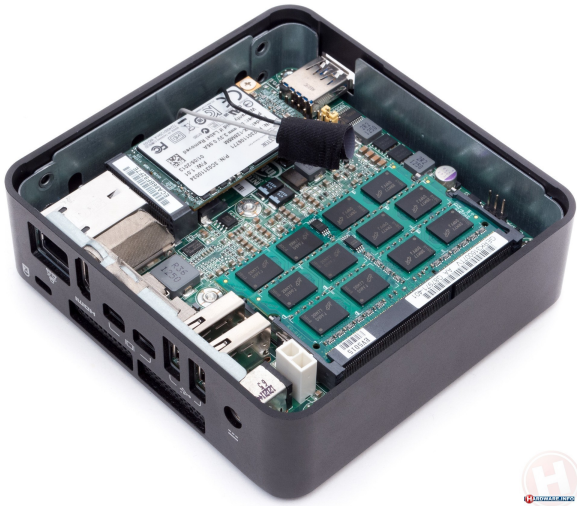

Figure 15. Intel NUC used as the primary processing unit of the system.

Payload.

The payload of the aircraft is be contained in the payload box, described below. The box is a 12.25in by 8.5in by 5.5in plywood construction split into two compartments. The shape of the box, including the division of compartments, is designed to provide structural support for the aircraft as well, and as such only minor changes to the structure of the box could be made. The primary components placed in the payload are the Intel NUC, the Prosilica 1660C machine vision camera [55], Netgear Ethernet switch for communication between components, redundant autopilot, and power assemblies required to provide power to all components in the payload.

The Prosilica 1660C camera was chosen for the digital imaging system due to its size, resolution, and global shutter speed. The prosilica is a 2.52in by 2.15in by 1.67in

camera weighing 0.392 pounds, and is capable of 34fps at 1.9 Megapixel, with image resolution of 1600 x 1200. The lens used was the M0814-MP Computar Factory Automation Lens [56]. The camera and lens were chosen primarily due to their previous use in research conducted using this airframe, camera, lens, and processor [14].

For the specific flight tests, the camera was calibrated before and after each flight to ensure there was no change in the intrinsic properties of the camera. This was done using the methods and MATLAB scripts developed by the California Institute of Technology [43]. The camera runs on 5-24VDC, but performs optimally at 12 VDC, and can be powered using a 3s LiPo battery for flight tests. It is connected to and controlled by the NUC through the Ethernet switch. The camera also outputs an analog voltage pulse each time an image is taken. This output pulse was intended to be sent to the autopilot and recorded in the telemetry.

Figure 16. The Allied Vision Technologies Prosilica 1660C machine vision camera is the primary electo-optical sensor utilized in this research.

The Netgear Ethernet switch used in this payload is a powered 5 port Gigabit switch, as seen in Figure 17. It is used to connect the network components of the NUC, the prosilica camera, and the ground control laptop for monitoring the NUC. The connection to the ground control laptop is achieved through the Wave Relay Mobile Ad-Hoc Network (MANET) [57], which will be described in the following section. This particular switch was chosen for its size and durability. Even though it has a metal frame and adds more weight to the payload, it is physically smaller than many other switches considered and allowed for more room in the payload box. This

switch was crucial in the connection to the ground control station from the payload due to the limited Ethernet connection of the NUC. Technical specifications for this particular switch can be found in product specification [58].

Figure 17. The gigabit switch used to connect the various network components is the Netgear Prosafe 5 Port Gigabit Switch.

The Telemaster UAV has one side of the box closed and the other open, so that the heavier components can be held on the more secure side of the payload box and the camera and other components needing to be open to the outside in the other side. In order to power the NUC, a voltage regulator was used to reduce the voltage of a 6s 3300 mAh LiPo battery from its nominal 22.2V to the necessary 19V input to the NUC. This was placed on the same side of the box as the NUC and the 6s LiPo battery. The other side contained the camera, ethernet switch, and the 3s LiPo required to power these two components. Figures 18 and 19 below show the configured box for flight tests.

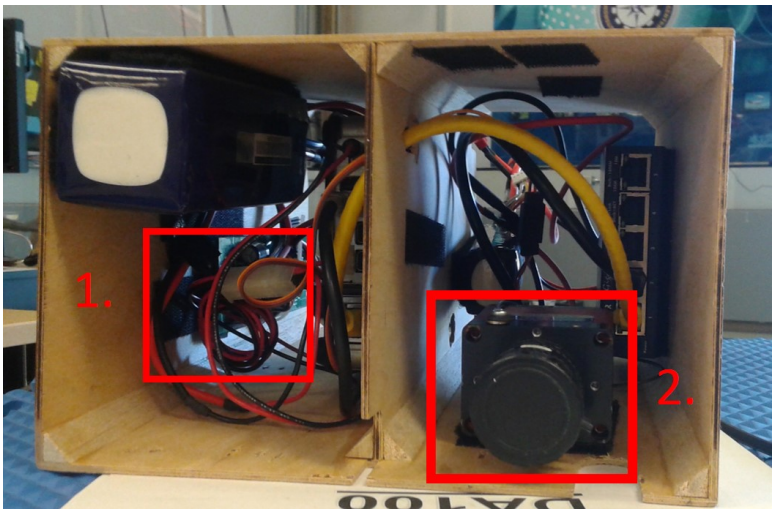

Figure 18. The bottom view of the constructed payload box to be flown in the Telemaster. The Prosilica camera can be seen on the right (2.), and the voltage regulator required for the NUC can be seen on the left(1.).

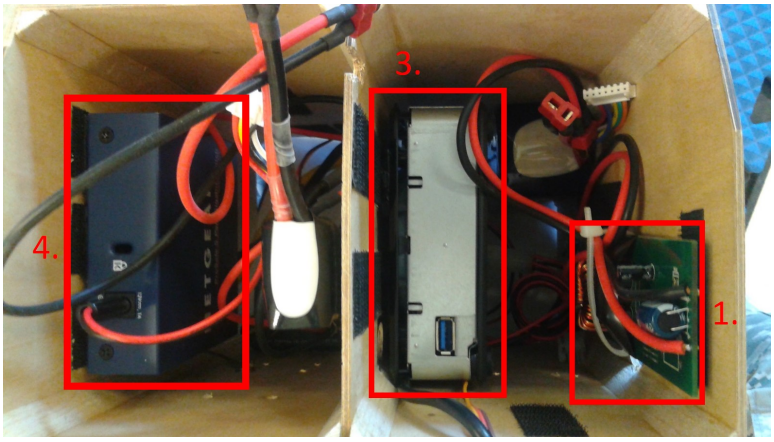

Figure 19. The top view of the constructed payload box to be flown in the Telemaster. The NUC (3.) and Voltage Regulator (1.) can be seen on the right side of the payload, which is closed to the outside of the aircraft, and the Netgear Switch (4.) can be seen on the left.

Ground Control Platform.

The ground control platform consists of multiple components as well. This subsystem includes the communications for both the aircraft and the payload, the ground control stations for the autopilot and the payload, and the required software on the ground control stations (GCSs). For communication to the autopilot from the primary

GCS, an RFD900 modem is used at 1W transmitting power and 915MHz frequency. For communication to the NUC from the secondary GCS, the Wave Relay MANET was used. This system works as a wireless IP router connecting the GCS to the airborne node with up to 2W transmitting power. This was utilized in the form of a remote desktop set up on the secondary GCS to see and control the NUC, purely for monitoring the progression of the algorithm and navigation solution. The remote desktop is established using an open source network client called RealVNC[59] on the secondary GCS to connect to the virtual network computing (VNC) programs built-in for Ubuntu 14.04 on the NUC.

For controlling the autopilot, Mission Planner v1.3.24 was used on the primary GCS. Mission Planner is open-source software that includes an extensive graphics user interface (GUI) that allows for full control and modification of parameters of the autopilot. Figure 20 shows an example of what this GUI looks like on the "Flight Data" screen, showing the aircraft's position as well as a virtual heads up display (HUD) to show the attitude of the aircraft. This software is used for the control of the UAV in autonomous mode, setting waypoints, and changing the flight parameters of the aircraft. Mission Planner hosts many technical specifications and software information online [60].

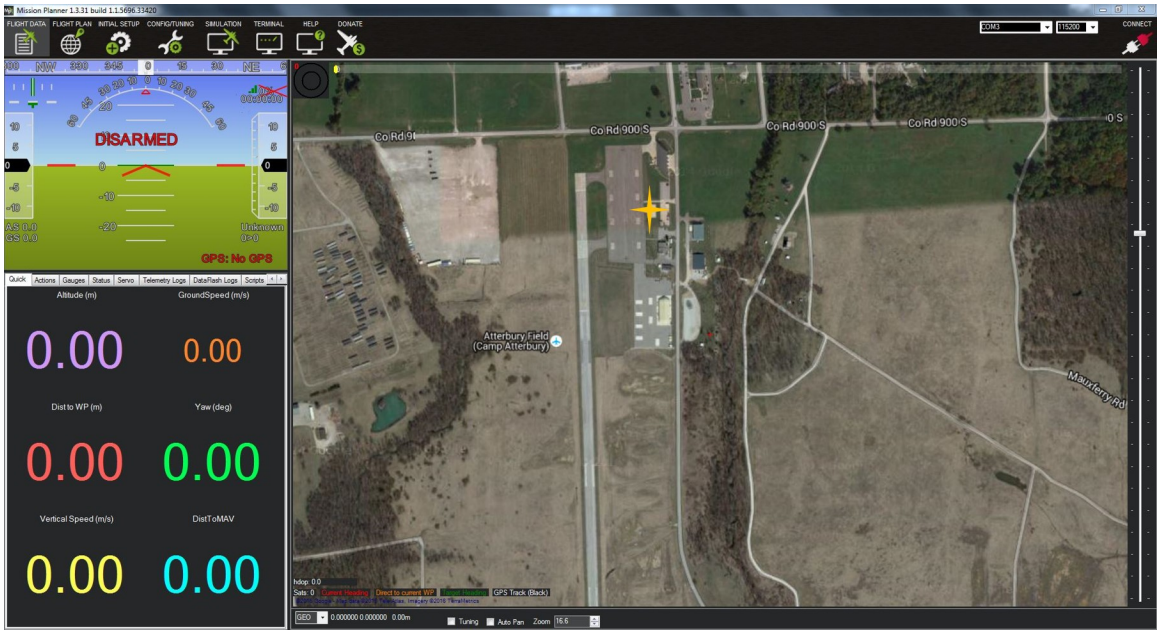

Figure 20. Mission Planner is the primary sofware run from the GCS, and is responsible for control of the aircraft during autonomous mode. Waypoints can be written to the autopilot using the 'Flight Plan' tab seen at the top, and the real-time position and orientation seen on the map and HUD shown in the image. The primary position of all GCS for Flight Tests is denoted by the golden star.

Integration.

The above components and subsystems were integrated to form the full UAS, as well as the vision-aided navigation solution. Figure 21 shows the diagram of the full system as well as the information passed from each subsystem to the next.

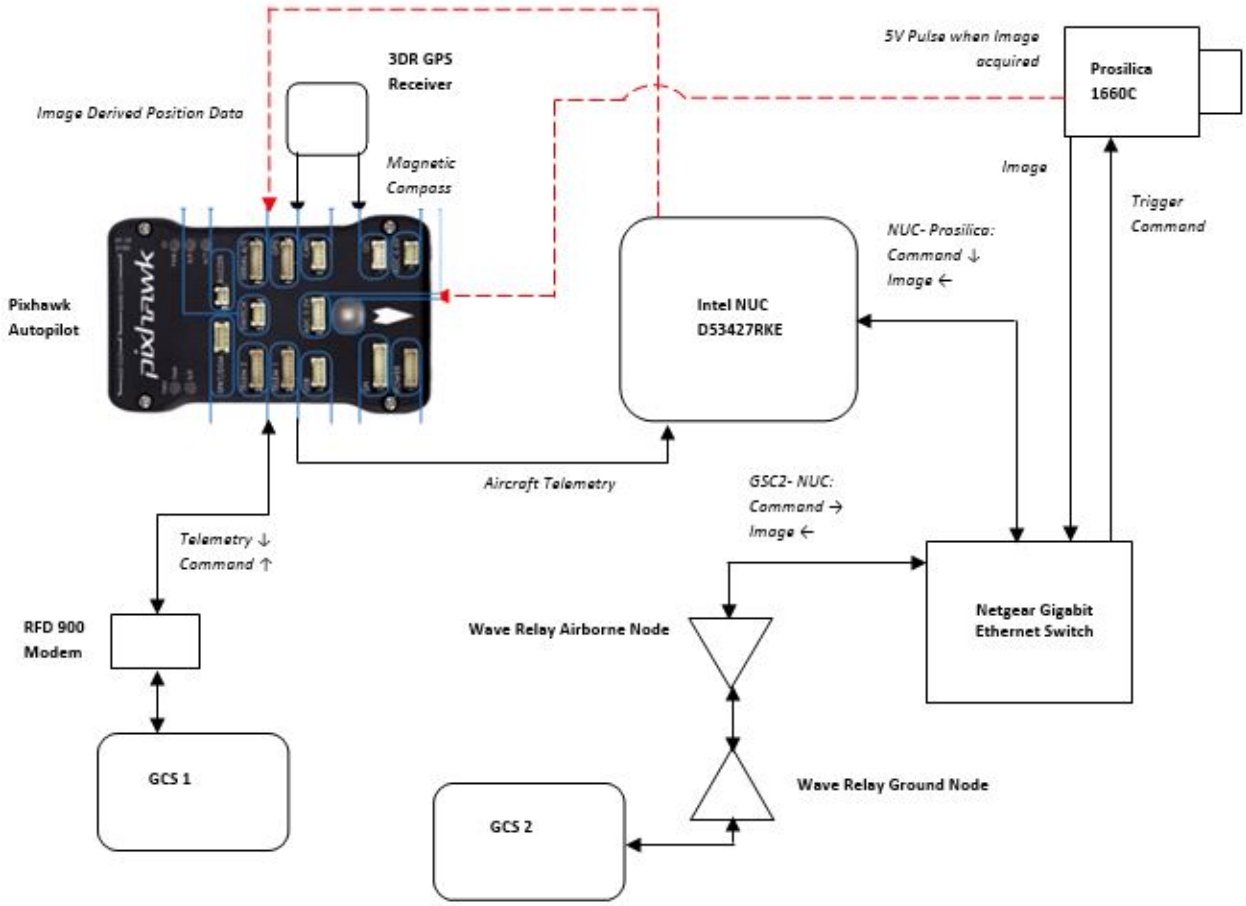

Figure 21. The system diagram for the integration of the image-aided navigation system and the Pixhawk autopilot. The control of the VisNav solution stems from the NUC, is monitored by the second GCS, and is then sent to the Pixhawk for autonomous navigation in GPS-denied environments. The dotted, red lines denote intended connections not achieved in this thesis, whereas the black denote the current operating connections.

It can be seen in Figure 21 that the autopilot is the end point for the information, with the final geo-location being sent to the autopilot in the serial port as an alternative GPS 3D pose. The inputs to the autopilot are the final geo-location solution, the information from the primary GCS, and the camera signal pulse when images are acquired. The primary outputs are the telemetry down to the GCS and to the NUC. The camera receives the command to take an image from the NUC, and sends the image back, but also sends a 5V pulse to the autopilot when the image is taken. The

5V pulse is recorded on a channel in the autopilot logs so that when the telemetry is sent to the NUC, the attitude of the aircraft can be matched with when the image was taken. However, due to integration issues, this was not the method utilized in this research. Instead the telemetry data was read from the autopilot immediately after the image was acquired, before the image was even buffered into memory and converted to a standard image format. It is imperative to obtain the aircraft telemetry as close to the time of the image as possible, and while the first method is ideal, the second method was used due to integration issues. An example of the coding for image acquisition is provided in Appendix A, while an example for the coding for reading the telemetry is provided in Appendix B, and Appendix C contains an example script containing the telemetry read immediately after the image was acquired.

The NUC utilizes a Python wrapper called Pymba in order to control the image acquisition of the camera. Pymba is based on the software development kit called Vimba provided by the Prosilica manufacturer, Allied Vision Technologies. This allows control of the internal parameters of the camera, including exposure, aperture settings, and many others. The lens allowed for manual adjustments of focal length and aperture, but these were fixed before calibration, and remained constant throughout the flight tests.

The NUC commands the camera to acquire an image, analyzes the image, reads the telemetry, and uses the PnP algorithm to produce a 3D pose of the aircraft. This is then sent to the autopilot via the serial port as a redundant pseudo-GPS signal for automated flight.

3.5 Test Strategies

Using the database, algorithms, and hardware described above, ground and flight tests were set in place to achieve the objectives. These tests are defined as either

functional tests, where the primary goal is to validate the integration and operation of the components, and performance tests, where the focus is on how well the components perform their intended functions. In total, three flight tests were conducted, and numerous ground tests were performed between each flight test. All Flight Tests were conducted at Himsel Airfield, located in Camp Atterbury, Indiana.

Flight Test 1.

The first flight test occurred in August 2015. This test was primarily to verify the performance of the airframe and suitability to carry the desired payload, as well as to verify the individual hardware components of the payload. The area flown was a single rectangular pattern in the area of interest, intended to gather images of both the open field by Himsel airfield as well as the helicopter parking apron. While the waypoints were a rectangular pattern, due to winds and turning parameters of the Telemaster, the final flight profile appears more like a peanut. From this test, telemetry data was saved on the autopilot and the images were saved on the NUC for post-processing. The flight pattern is shown in Figure 22.

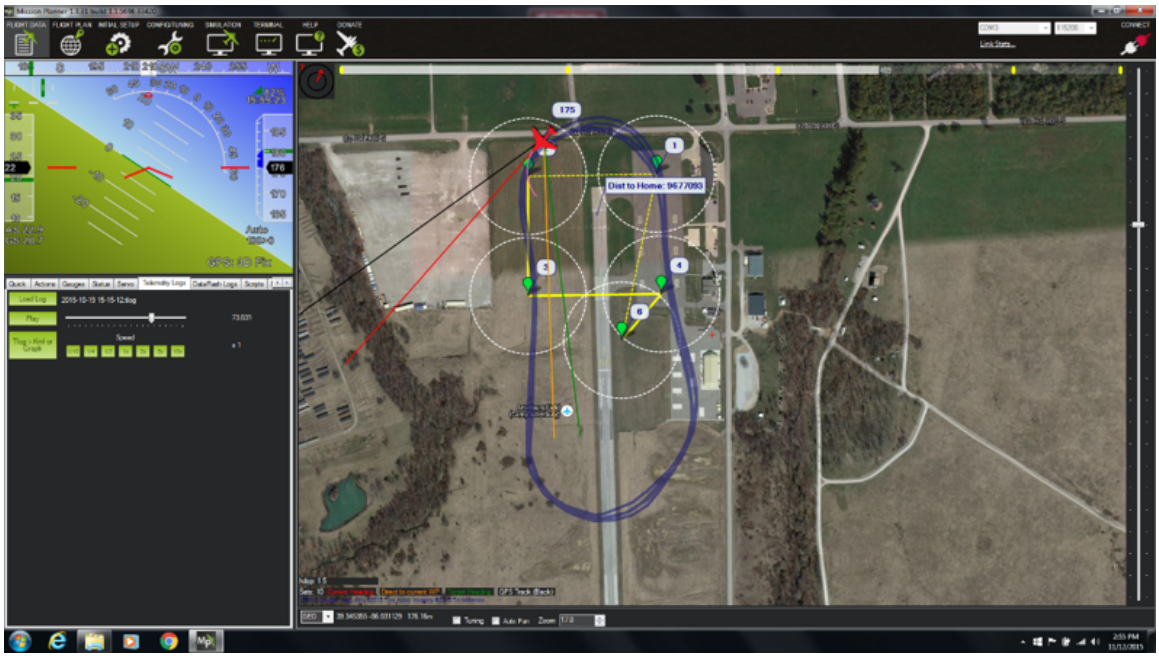

Figure 22. The Mission Planner telemetry display of the rectangular waypoint flight pattern A, flown for all three Flight Tests. It can be noted that the pattern includes the grassy fields next to Himsel airfield as well as the parking apron next to the airstrip.

From the data gathered, the PnP algorithm was constructed and tested. The autopilot was tuned to allow optimum performance while flying autonomously. The tests performed during the first flight test were almost entirely functional tests. The primary focus was to verify the feasibility of the aircraft and payload components to perform the required operations.

Flight Test 2.

The second flight test took place in October 2015, and was focused on the integration of the system components and collection of aerial imagery of the AOI. Images were acquired at multiple altitudes, as described by Table 1, to allow for testing altitude effects on the PnP solution. A total of 1740 images were collected while flying the same pattern as described in Figure 22 from Flight Test 1. Of these images, only the ones at the desired altitudes are listed below. The number of images from 160m

and 190m were an approximation from the telemetry data, since the wind made it difficult to discern the exact moment the change in altitude occurred.

Table 1. The test matrix used when conducting the Second Flight Test in October 2015.

Altitude [m]	Flight Pattern	Number of Images
130	A	304
160	A	290
190	A	570
250	A	280

From the images and telemetry data collected, the PnP algorithm was tested and improved for performing in this AOI. From the data, the PnP algorithms were modified to perform real-time and near real-time operations in preparation for Flight Test 3, improving performance and computation time of the solution. Numerous tests were conducted to analyze the data in order to prepare the solution for the last flight test, including hardware in the loop (HIL) tests, in which the autopilot was connected to the NUC while in ground tests in order to ensure proper reading of actual attitude from the Pixhawk autopilot.

Flight Test 3.

This demonstration flight was used to verify that the system not only performs to the desired degree of accuracy, but also at real-time or near-real-time for the autonomous operations. From Flight Test 2, it was noticed that there were limited measurements possible from the flight pattern flown, previously shown in Figure 22. To account for this, a different pattern was flown in the hopes of obtaining more measurements, referred to as Pattern B shown in Figure 23. The test matrix below, Table 2, describes the flight parameters used to verify real-time autonomous flight of COTS UAVs. The results from this flight test are discussed in detail in the next chapter.

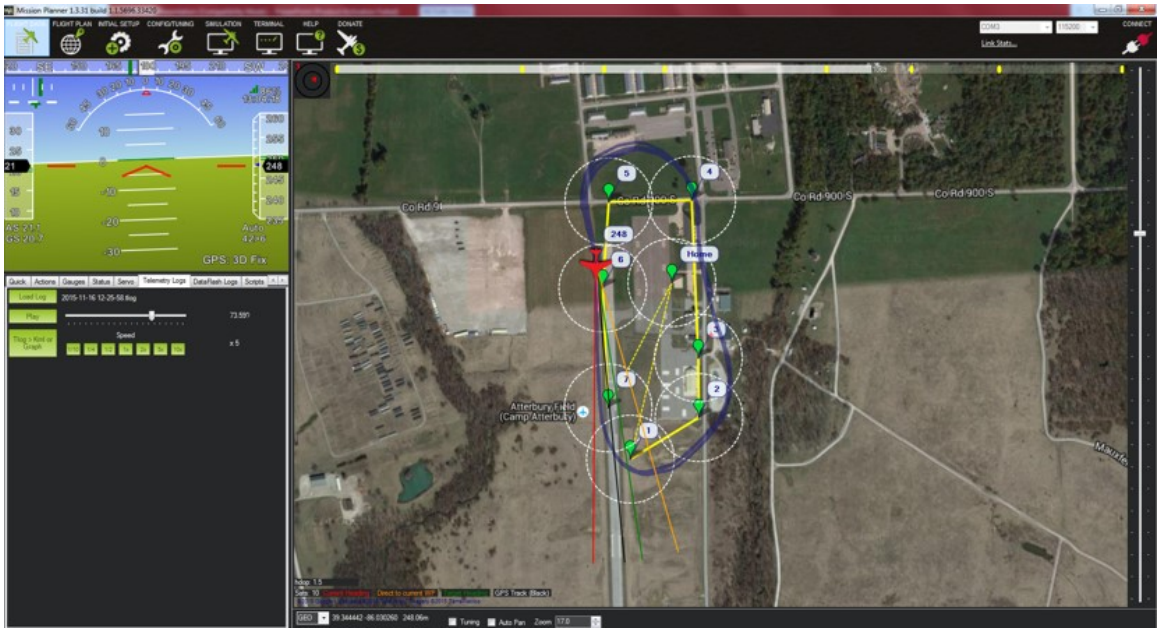

Figure 23. The Mission Planner telemetry display of the rectangular waypoint flight pattern B flown for the third Flight Test. This pattern was chosen to overfly more obviously unique features, especially the buildings located at Himsel airfield.

Table 2. The test matrix used when conducting the Third Flight Test. It specifies the heights flown by the Telemaster UAV for each flight pattern and the resulting images and successful PnP measurements obtained at each altitude.

Altitude [m]	Flight Pattern	Number of Images	Number of Measurements
190	A	159	14
250	A	130	35
250	B	227	87
190	B	155	35

The final pre-flight process used for these flight tests is given in Appendix E. This includes the pre-flight setup for the UAV and the payload as well, including brief instructions for how to calibrate the camera for VisNav flight tests.

Chapter Conclusions.

This chapter introduced the details of the feature database development, and how it was utilized in the position solution of the system. The specifics of the attitude-aided PnP solution were also described, leveraging OSS to create a stable PnP algo-

rithm. Both of these processes were developed by Venable, and modified to fit the specific AOI of this research. The system structure was designed for implementation and Flight Tests in the specific AOI. The test structures were developed for the Flight Tests in order to test the VisNav solution on post-processed flight data as well as real-time flight data. The results of these flight tests as well as the modifications to further improve the system will be detailed in the next chapter.

IV. Results and Analysis

4.1 Chapter Introduction

This chapter focuses on the results obtained from real-time implementation of the vision-aided navigation system, and the analysis of these results. The primary focus is on the real-time results and analysis of the data from the flight tests. The second flight test acquired images of the AOI for post-processing development of the real-time PnP solution. The third flight test implemented this real-time algorithm for different regions of features in the AOI. The secondary focus is on alterations done to the database, PnP algorithm, and other parameters used in the calculation of the navigation solution to improve the accuracy of the system. The primary objective of this chapter is to present the data gathered in a manner useful for comparisons to other navigation systems and to address potential limitations of the current system. The analysis will consist of a look into the position calculations from the flight tests; breakdowns of the time and errors for the computation, database variants, parameter adjustments; and limitations to the system.

4.2 Real-Time Analysis

Real-time testing of the system was performed in October 2015 at Himsel airfield in Camp Atterbury, IN. During this flight test, two different ground-track patterns were flown, as seen in Figure 24 below. The first ground-track was the pattern flown during the second flight test, in which it was noticed that position solutions were only able to be calculated in a limited number of areas. Once the system was modified for real-time calculation, this area was flown again, with similar results. The switch to the second pattern was done to increase the number of features the aircraft saw while in flight.

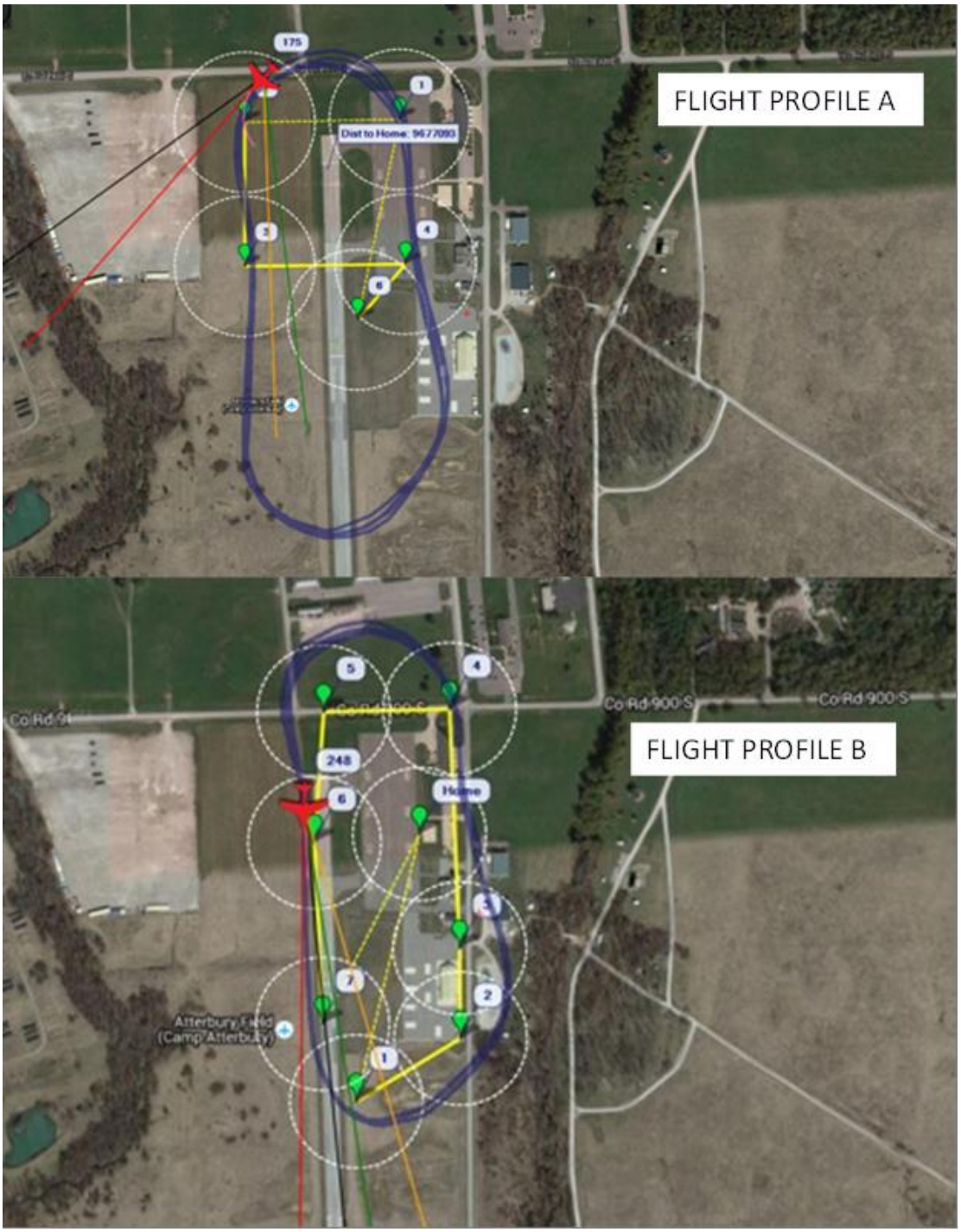

Figure 24. The Mission Planner telemetry display of the rectangular flight patterns flown for the third Flight Test. It can be seen that the first pattern, A, was flown in order to capture the field West of the airstrip, and the second pattern was flown to capture the buildings East of the airstrip as well as the airstrip itself. It was noted that the system was able to calculate more position solutions from Pattern B.

From these different ground tracks, the position error was calculated with respect to the onboard GPS position measurement. Instead of a position error in regards to the vehicle frame, the position error is presented in north, east and down coordinates. The 6DOF solution was obtained using the OpenCV RANSAC algorithm described in Chapter 2. The 3DOF solution was obtained through the attitude-aided PnP calculation process defined in Section 3.3. Figure 25 below shows the errors calculated from the open-source 6 degree of freedom PnP calculation. Figure 26 is the rotation constrained PnP solution errors.

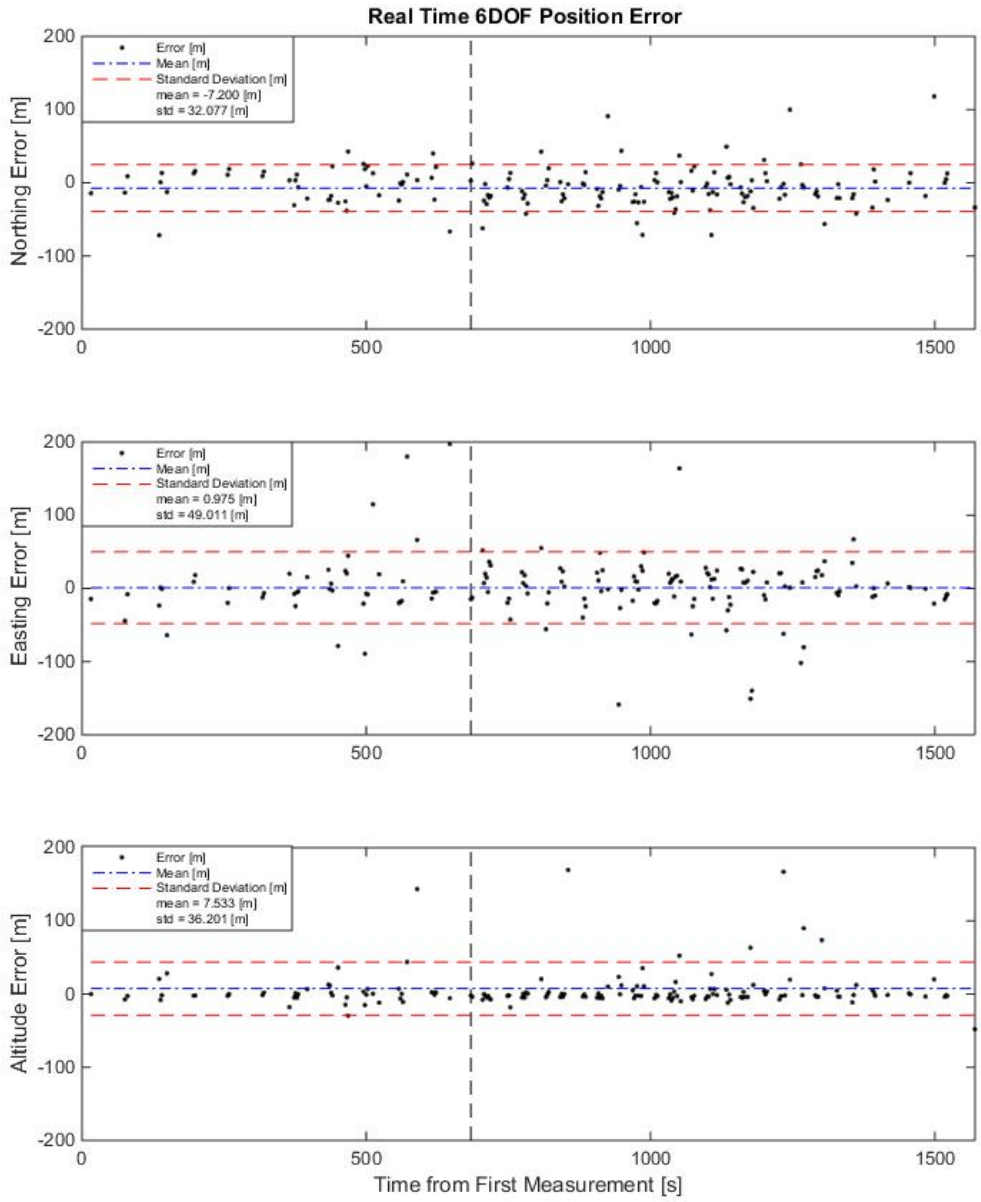

Figure 25. The NED error of each measurement from the 6DOF OpenCV PnP RANSAC calculation for the real-time verification Flight Test. The vertical dotted line denotes the separation from Flight Profile A and Flight Profile B.

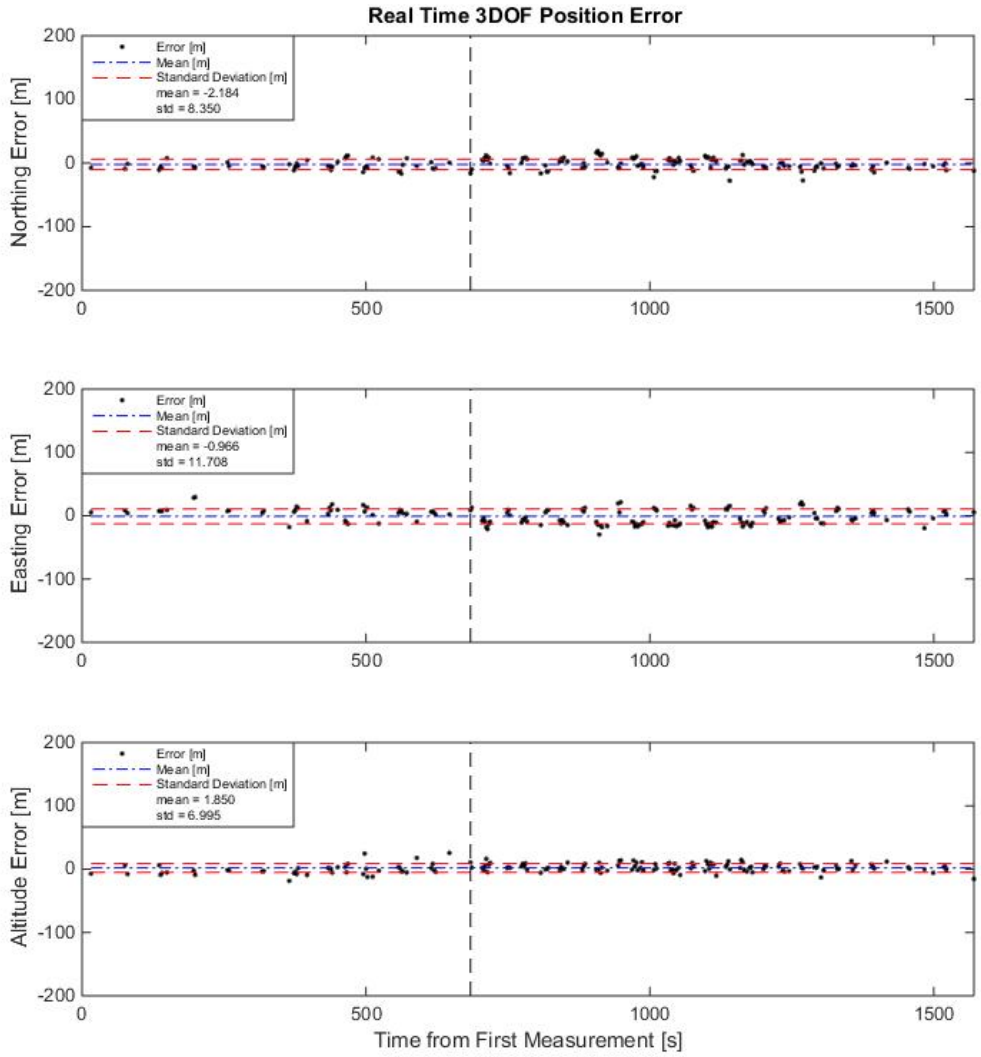

Figure 26. The NED error of each measurement from the attitude-aided 3DOF PnP calculation for the real-time verification Flight Test. The vertical dotted line denotes the separation from Flight Profile A and Flight Profile B.

As can be seen by the figures above, the 6DOF solution does provide less accurate solutions as expected. The solutions obtained using the 6DOF OpenCV method had an overall DRMS of 58.841m and and MRSE of 69.435m. The attitude-aided 3DOF solution is more difficult to see in Figure 26, which has the same scale as Figure 25 for comparison purposes. A closer view of the 3DOF solution is given in Figure 27.

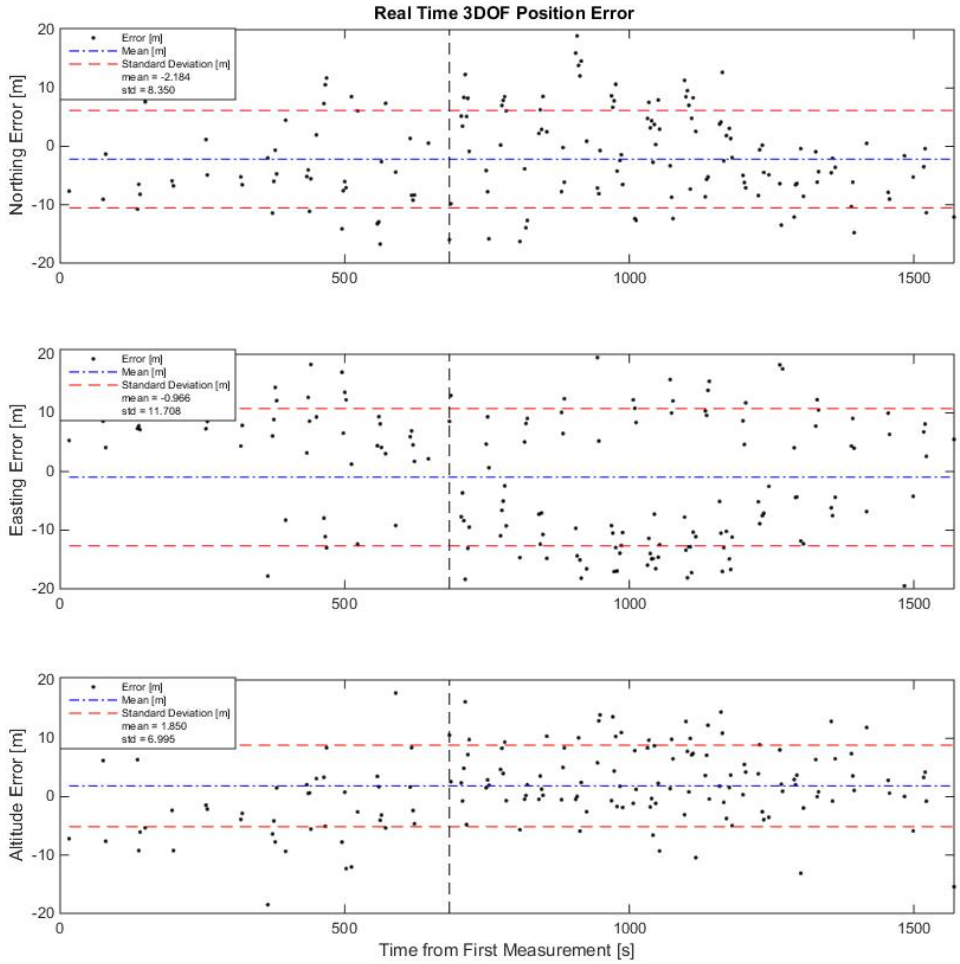

Figure 27. The same data as shown in Figure 26, but not forced to the same constrained axes as the OpenCV solution. This allows for greater visible inspection of the measurement errors. The vertical dotted line denotes the separation from Flight Profile A and Flight Profile B.

The 3DOF attitude-aided solution error in any direction is much smaller than the 6DOF solution. A common error characteristic was present in the East direction, as can be seen throughout the flight data. When the aircraft was flying either North or South, a near constant error appeared in the East and West direction, dependent on the flight direction. This led to the suspicion of an attitude error in the VisNav solution, predominating in the roll characteristic of the solution. The speculation was

further enforced from the fact that the error is greater at 250m altitude than at 190m altitude, and a constant attitude error would present this sort of characteristic. This suspicion is tested using post processing in Section 4.4.

For clarification between different analysis techniques, the mean, standard deviation, distance root mean square error, and mean root mean square error as used in this thesis are defined as follows:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{24}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{25}$$

$$\mathbf{RMS} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}} \tag{26}$$

$$\mathbf{DRMS} = \sqrt{\frac{\sum_{i=1}^{n}(x_i^2 + y_i^2)}{n}} \tag{27}$$

$$\mathbf{MRSE} = \sqrt{\frac{\sum_{i=1}^{n}(x_i^2 + y_i^2 + z_i^2)}{n}} \tag{28}$$

Table 3. The Results of the Real-Time verification flight. The Pattern refers to the Flight Profile as defined in Section 3.5, and the % Measurements is used to show the percentage of measurements obtained for the given set of flight images.

Pattern	Direction	Mean [m]	STD [m]	DRMS [m]	MRSE [m]	% Valid Measurements
A	North	-4.04	7.12	13.37	16.17	16.96
	East	5.74	9.45			
	Down	-0.93	8.73			
B	North	-1.58	8.78	14.90	16.29	31.94
	East	-3.51	11.49			
	Down	3.01	5.89			

Even with the real-time errors present in the solution, the system preformed close to the desired specifications with an attitude-aided solution as defined in Chapter 1. The DRMS was 14.901m and the MRSE was 16.294m for the second flight profile. Though these results are not within the same accuracy of the GPS used (2.5m), it is of a similar order of magnitude while proving the capabilities of the system for near real-time VisNav. The results presented in Table 3 show clearly the increase of nearly double in percentage of successful measurements acquired when the flight profile was changed to Flight Profile B, seen in Figure 24. Even with the change in flight profile, there were still areas in which measurements were not possible. Figure 28 shows both flight profiles in the AOI, with the locations of valid measurements shown in red.

Figure 28. Flight Test 3 telemetry for both Pattern A and B is shown by black dots for each image saved. The red dots are the calculated locations for which a PnP measurement was possible.

The algorithms and feature database were further modified to improve the accuracy of the solution. Tests were conducted on the images and flight data from Flight Test 3, creating simulated flight tests. From these tests it was assumed that if the system were to fly in the altered configurations, the real-time performance would be the same or very near the same as the simulated results. These post-processing algorithm refinements will simply be referred to as either post-processing refinements or just refinements for the remainder of this thesis. These refinements will consist of

tests run using the entire Flight Test 3 data including the telemetry and images, not just the data specific to one flight profile or another.

4.3 Time Break-down

For determining the measurement rate of the SUAS, a time breakdown of the calculation must be conducted. The time considered for this breakdown is the time it takes from the image acquisition command from the NUC to the time a final pose solution is calculated. This breakdown will consist of analyzing the image acquisition process, the data transfer within the system, the OpenCV position solution calculation and the attitude constrained position solution. As stated in Ch 1, the desired measurement rate is approximately 1Hz or faster. A slower rate is acceptable as long as the system still calculates the VisNav solution in real-time. The time required to present and save of data was not considered in this analysis, because these tasks are dependent upon the user's preference, and times to accomplish these can very greatly upon the amount of data desired to be saved or displayed to a screen.

For image acquisition, as covered in Chapter 3, a Python wrapper is used to communicate between the NUC and the Prosilica. With this came many issues dealing with timing and integration of the system. Vimba, the SDK supplied by Allied Vision Technologies for controlling the Prosilica cameras is capable of more than 30 fps, dependent on the exposure time used, but recreating this rate of acquisition using Pymba was not possible. Table 4 below shows the time break-down of the VisNav system for ideal operations regarding exposure settings and background processes running.

Table 4. Time breakdown of system solution from Image Acquisition to final position solution for optimum settings.

Sequence	**Time** [s]
Image Acquisition	0.3718
Color/Format Conversion	0.0500
Read Telemetry	0.0001
Image Matching	0.0527
OpenCV PnP RANSAC Calculation	0.0837
Attitude-Aided PnP Calculation	0.1054
Total	0.6637

Using ideal times for each of these categories, a total pose calculation time was analyzed to determine if the system performed at the desired measurement rate. For the optimum times, the VisNav solutions were calcualted faster than 1Hz, but during Flight Test 3 however, the solutions were calculated closer to 0.45Hz. From Table 4, it can be seen that the greatest contribution to the calculation time was the *Image Acquisition* process. Improvements to this process, as well as the other processes, will be discussed in Section 5.2.

4.4 Error Analysis

The primary method for improving the solution is to characterize the errors from the position solution, and determine methods for improving the system to reduce these errors. For this section, there are two types of errors addressed: known errors, and calculated errors. Known errors are ones that may require calculation, but are constant throughout the flights and can be determined before flight. Calculated errors are those that can be accounted for during or after flight, but overall are more difficult to determine and account for beforehand.

For many error parameters, residuals will be discussed as a metric for the improvement of the system. For this research residuals are defined as the difference between predicted and actual feature locations in the image, as previously detailed in Section 2.3. These are calculated using the GPS truth and autopilot attitude at the time of the given image. The primary metric for gauging improvement of the system for each test was the final horizontal position error DRMS of each simulated Flight Test.

Known Errors.

The primary known errors considered are those from the intrinsic and extrinsic camera calibration, and the known characteristics of the GPS system used as the truth position. The majority of these, as mentioned above, can be calculated before flight and are held constant for the duration of the tests. For many of these errors, a true, accurate value cannot be determined due to limitations of the known area of interest. Instead approximations are used, where actual values cannot be determined.

For camera calibration, the intrinsic parameters, such as the $\mathbf{K}$ matrix and distortion parameters, were calculated using methods and scripts described in [43]. With the refinement process of intrinsically calibrating the camera, a pixel error is used as the metric for determining the most accurate parameters possible. The pixel error from the intrinsic camera calibration introduces only minor errors, and as such is considered negligible for this thesis. Extrinsic calibration, while introducing some errors, is used to reduce the errors that arise from an improper camera frame to body frame rotation matrix. This calibration must usually take place while in flight or during post-processing. Figure 29 shows the relationship of the camera frame to the body frame of the aircraft, separated to show greater clarity.

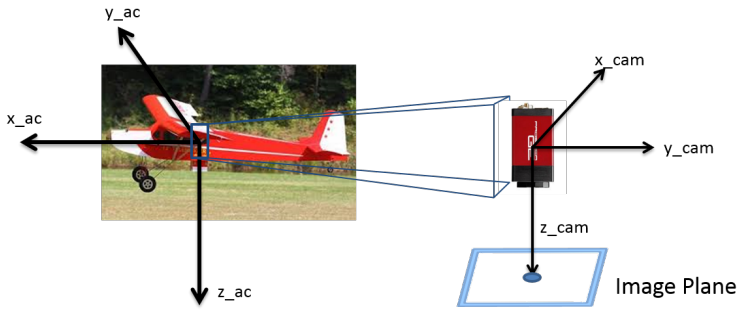

Figure 29. The ideal placement of the digital camera with respect to the COTS UAV autopilot. The autopilot frame is oriented with x_{ac} pointed out the nose of the UAV, y_{ac} pointed out the right wing of the UAV, and z_{ac} pointed directly down from the fuselage. The two frames ideally share the z_{ac}.

An initial guess was used for the DCM to calculate the residuals and position solutions, assuming the camera was positioned exactly as depicted in Figure 29. The measured location of the object in the image frame was compared to the calculated location in the image frame, and the residual between these two locations was calculated. Small corrections to roll, pitch, and yaw were then applied until a minimum residual was found. These adjustments were applied to the initial guess to develop a final, roughly calibrated DCM. Though this solution is not perfect, it is the closest approximation available with the given resources.

Calculated Errors.

Of the many calculated errors that can contribute to the total position error of the solution, only one will be considered in detail in this thesis. The primary calculated error is the attitude error from the autopilot at the time the image was acquired. These errors are unique to the stage of flight and conditions the autopilot are experiencing. The calculation of this error was done in the same manner as the extrinsic camera calibration, with varying the adjusted roll, pitch, and yaw angles to reduce the residuals and the DRMS error of the attitude-aided solution. After this test was done, the resulting angle rotations were used again in post-processing

the real-time data. Figures 30 and 31 show the DRMS error calculations for each refinement post-process test. Figure 30 shows the corrected roll and pitch angles in degrees for the full range of $[-5.0 : 2.0]$, and Figure 31 shows the corrected angles focusing on the lower end of the range.

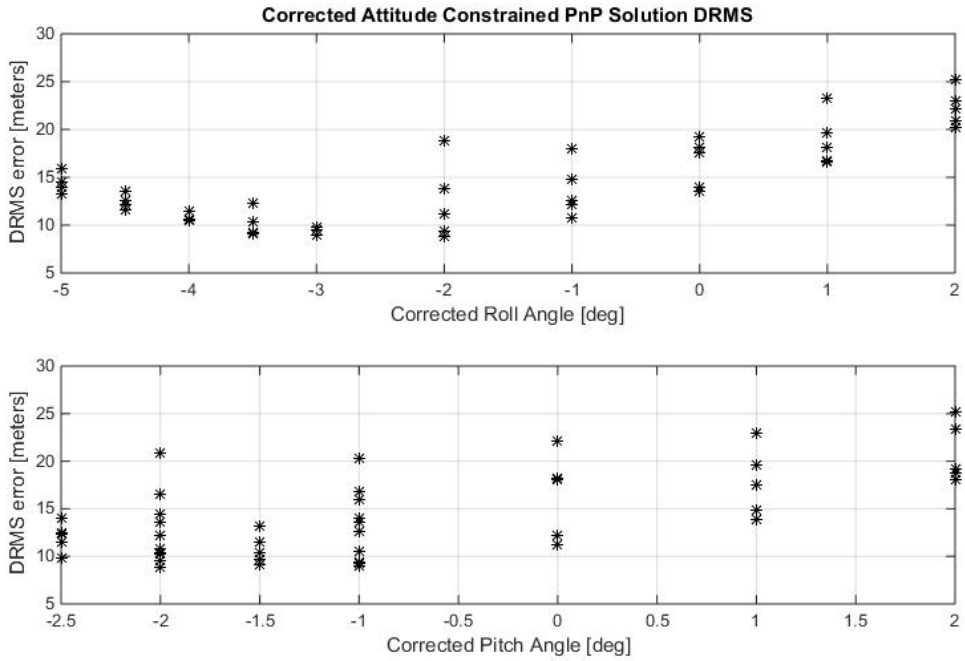

Figure 30. Simulated flight tests were conducted with augmentations to the attitude, within the range of $[-5.0 : 2.0]$ degrees for roll and pitch was from $[-2.5 : 2.0]$ degrees.

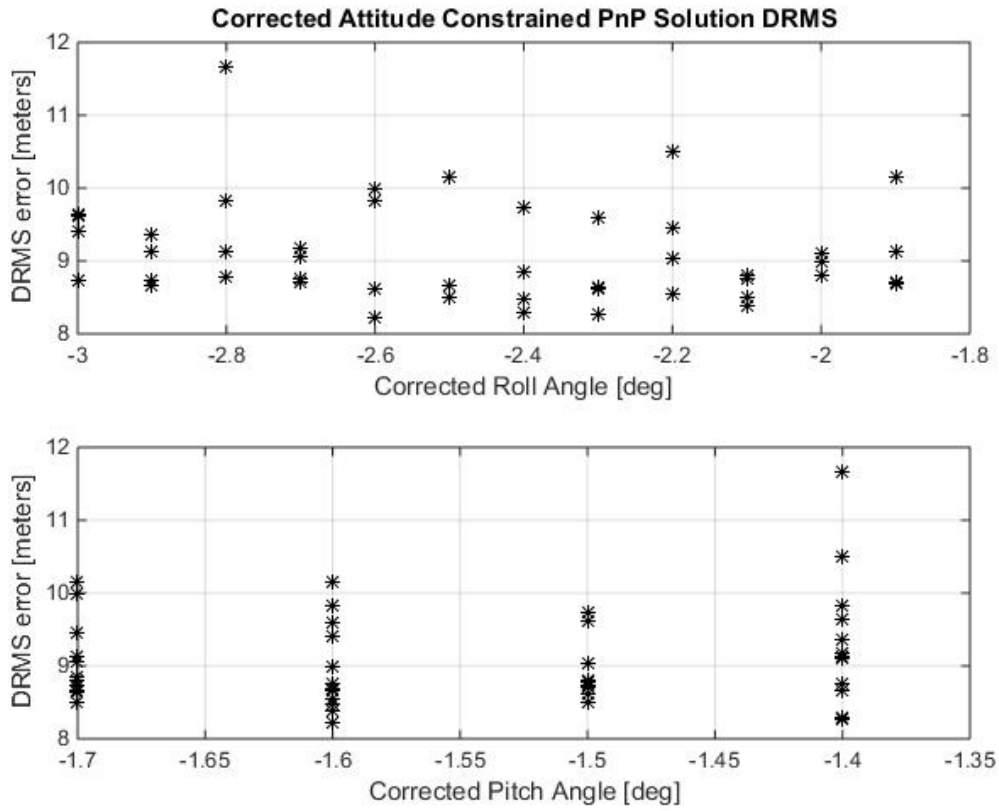

Figure 31. Simulated flight tests were conducted with augmentations to the attitude, within the range of $[-3.0 : -1.8]$ **degrees for roll and** $[-1.7 : -1.4]$ **for pitch.**

From the corrected angle analysis, it was determined that in these tests, for a corrected roll angle of approximately -2.5 degrees, the DRMS error was significantly decreased. For the corrected pitch angle, however, it was noticed that for negative angles the DRMS values were roughly the same with only minor differences. The second corrected attitude test was used to analyze the roll angles around -2.5 degrees to determine the minimum error in the range accurate to 0.1 degrees. The lowest DRMS error was noted with corrected roll and pitch angles $[-2.6, -1.6]$ with a DRMS of 8.22 meters. Similar analysis was done with the Flight Test 2 data and the optimum augmented attitude was $[-2.0, -1.4]$, showing a consistent attitude error in the system. This attitude correction aims to correct for errors in the calculated attitude from the autopilot, errors induced from the timing error between the telemetry and the image

acquisition, as well as errors from the extrinsic camera calibration. Since none of these errors can be individually characterized, they are all lumped together as the attitude error of the system. The corrected attitude from Flight Test 3 will be used when performing further post-processing refinement tests for all other results in this chapter.

4.5 Database Variants

Another intended improvement to the system to increase accuracy and sampling time was to develop variants of the feature database used for 2D to 3D correspondence. This section of the chapter will include an analysis of the database used for real-time flight, as well as variations of the same GeoTiff image to improve performance of the system. A second GeoTiff was created using entirely open-source images and geographic information, and from this GeoTiff a new feature database was created. All of these database variants were used in post-process refinement tests of the real-time position calculation to compare overall performance.

Database Parameter Adjustment.

Figure 32 is the image used for the database, detailing the positions of the database features in the image plane. It was noticed that the features are primarily located on trees and other areas that are of no interest when performing matching.

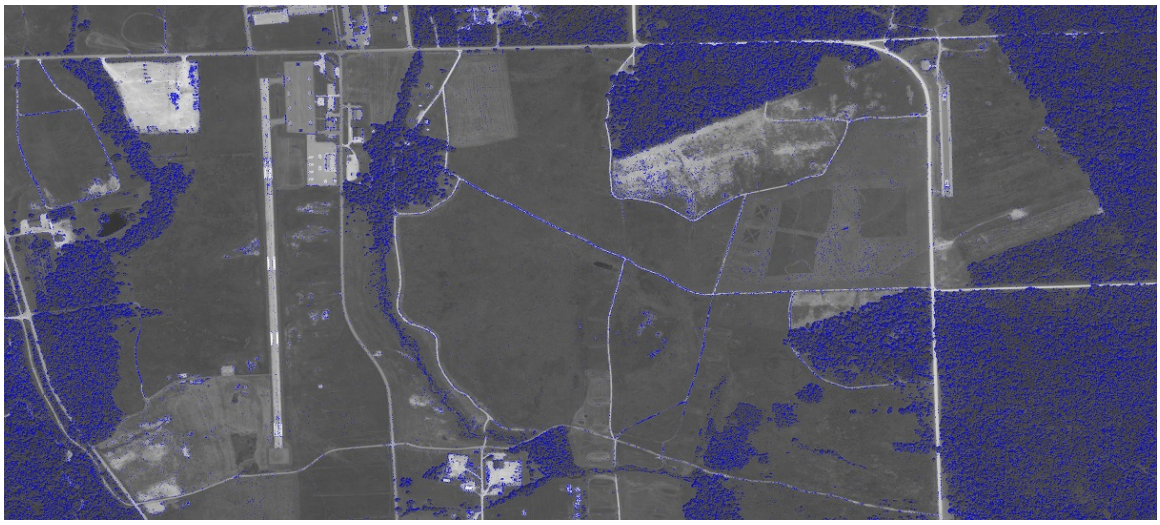

Figure 32. The database image with the SIFT features overlayed in blue used for all Flight Tests. This is referred to as the default Database.

Features that were located in trees were considered unusable, since the foliage is never guaranteed to be in the same location from one image to the next, and as such the features are not guaranteed to be located in the same area of the foliage. Since the majority of features were unusable for image matching, and as such unusable for position calculation, parameters were adjusted for database variants to either move these parameters to more repeatable locations and/or to find more features useful for matching and localization. The primary method of variation for databases was to adjust the SIFT parameters used in the feature extraction. The parameters used in developing the SIFT feature extraction algorithm are given below with the affect of each on the number and/or quality of features in the system.

- **Contrast Threshold:** filters out weak features in semi-uniform (low-contrast) areas
- **Edge Threshold:** filters out edge-like features
- **Number of Features:** number of "best" features retained, scored by local contrast

- **Number of Octave Layers:** number of layers considered in SIFT
- **Sigma:** the sigma of the Gaussian blurring applied to the input image at the zero Octave

For the parameters listed above, the numbers of features and number of octaves were kept constant, and the thresholds and sigma were altered and the feature responses in the altered databases recorded and applied to simulated test runs. In order to gauge how varying each parameter effects the database, tests were run for an increase and decrease of each parameter. For each test the total number of features, the number of features located in the wooded regions, and the number of features in the specific flight test area of interest were recorded for analysis. Included with the data is the percentage of useful features. These are features that are not located in the wooded regions, and thus are useful for the image-aided navigation solution. Table 5 below shows these tests as well as the results generated. Figures showing each database variant with feature locations can be found in Appendix D.

Table 5. Analysis of Database Variants for Contrast Threshold, Edge Threshold, and Sigma deviations.

Contrast Threshold	Edge Threshold	Sigma	Wooded Features	AOI Features	Total Features	% Useful
0.02	10	1.6	437443	16471	624535	29.957
0.03	10	1.6	304677	8793	386591	21.188
0.04	10	1.6	178629	5312	220361	18.938
0.08	10	1.6	3736	1038	6677	44.047
0.04	5	1.6	165673	4239	200680	17.444
0.04	15	1.6	180784	5805	225301	19759
0.04	10	0.8	384452	12597	474490	18.976
0.04	10	2	103724	3122	127226	18.473

As can be seen from the database tests, reducing the contrast threshold creates more features overall, including in the trees, but it also allows for more features in the uniform regions such as the field West of Himsel airfield. This created the

most features in the specific region of the flight tests while also increasing the overall percentage of features not located in the trees. Figure 33 shows the definition of the flight test region of interest as well as the specific areas considered to be wooded. Lowering the initial sigma parameter also increased the number of features in the flight test region, while keeping the percentage of useful features relatively constant. Variation to the edge threshold did not vary solutions or performance enough to constitute a definitive improvement to the system, and as such these database variants were not considered further.

Figure 33. The areas considered "wooded" are outlined white rectangles. Features located in these regions were not useable for VisNav calculations. The area of interest shown in blue is the area in which the Flight Tests occurred.

The databases for the reduced contrast and reduced sigma were used in post-process refinement tests, and the behavior of these compared to the behavior of the results from the corrected attitude flight data shown in the previous section of this thesis. The comparison of these tests are given in Table 6, detailing how each database variant performed in the refinement tests.

Table 6. Refinement Flight Test results for Database Variants of reduced Contrast Threshold and reduced Sigma values.

Database Variant	Measurements	DRMS	MRSE	% Valid Measurements
Base DB	184	8.219	10.184	27.422
Contrast = 0.02	291	9.818	13.140	43.815
Contrast = 0.03	243	9.480	11.368	36.214
Sigma = 0.8	167	10.870	13.977	24.888

From these tests, it was determined that for accuracy the original database with attitude correction performs the best with a DRMS of 8.219 meters, but for obtaining the most measurements the lower contrast threshold database is preferred. The lowest Contrast Threshold of 0.02 produced solutions almost twice as often as the baseline case while increasing the DRMS by only 1.2 meters, and using a threshold of 0.03 increased the percentage by approximately 10 while increasing the error by only 1 meter. Though lowering the contrast threshold increased the rate at which measurements were obtained, the goal is to find the most accurate system, and as such the base SIFT parameters will be used for the remainder of this research. It was also found that for the refinement tests that the default SIFT parameters generated the most accurate results as well as a higher percentage of measurements when used to analyze the flight imagery. The default SIFT parameters were used in the creation of the GoogleMaps feature database due to accuracy of the refinements when using the default database.

OSS Imagery.

A GoogleMaps [49] database image is shown below in Figure 34 with the features overlayed on the image. The image was obtained from Google Maps, and was only a fraction of the resolution of the DigitalGlobe database image, being (5600x3944)

pixels as compared to (12098x5398) pixels. The Geospatial Data Abstraction Library (GDAL) [61] was used to apply the geospatial information to the image file, creating the GeoTiff used for this feature database.

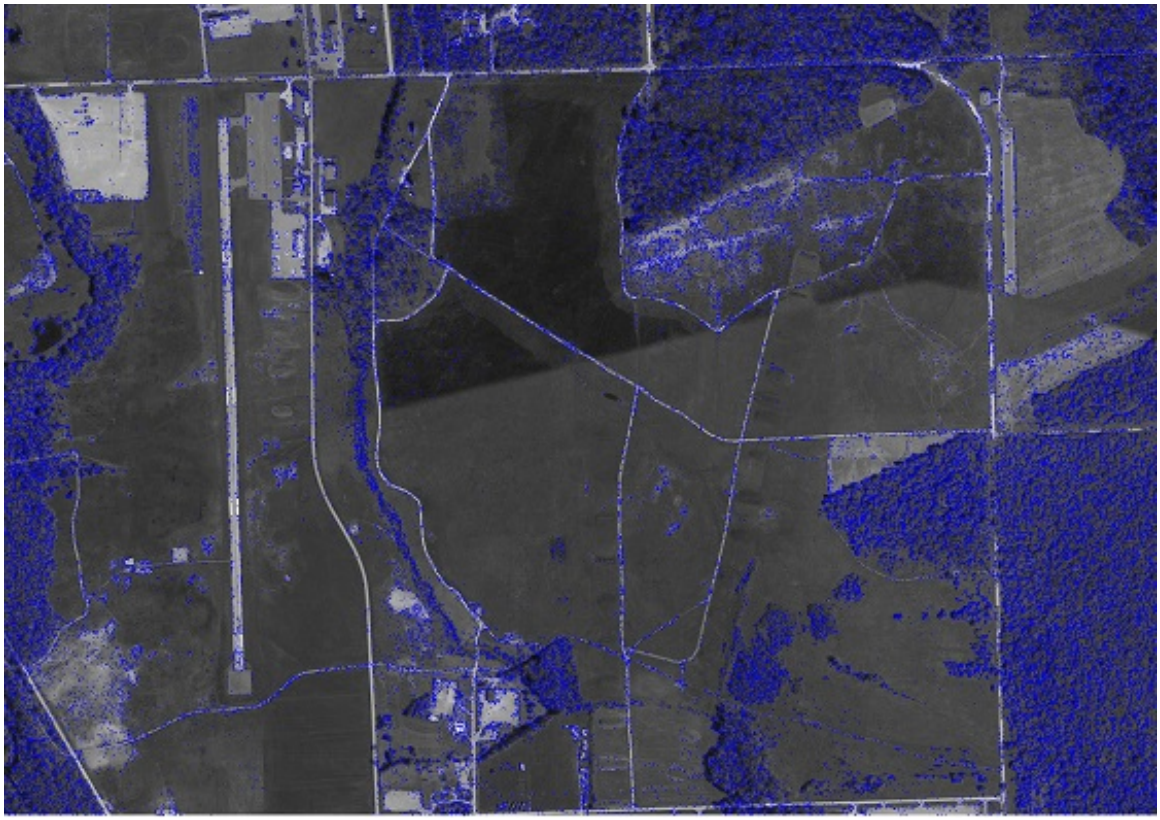

Figure 34. The GoogleMaps image used in generating the second feature database. It should be noted that the high resolution orthophoto obtained still retained the labels applied by Google in the image.

From analyzing the database variants, it was determined that the base SIFT parameters created the database capable of the most accurate PnP solution, and as such these parameters were used in the creation of the GoogleMaps database. This database was then used in refinement flight tests to determine the accuracy possible using such a database. Like the other database variants, the features of the GoogleMaps database were analyzed and their locations used to determine the number of useful features for VisNav solutions. Of the total 92350 features detected in the image, 4163 were located in the AOI, and a total of 36% were located in non-

wooded regions. These results can be compared to the other variants from Table 5, and seen that this variant has only 1000 fewer features located in the AOI, but double the amount of useful features overall than the database used for the Flight Tests. The size of this database was also much smaller, being only 18.8MB of data as opposed to the 44.8MB database used during the Flight Tests. The real-time results from Flight Test 3 are shown below to compare the results from the attitude-corrected solution using the DigitalGlobe database as well as the attitude-corrected solution using this database.

Table 7. Refinement Flight Test results for Database Variants of reduced Contrast Threshold and reduced Sigma values.

Database	Attitude-Correction	6DOF DRMS [m]	3DOF DRMS [m]	% Valid Measurements
DigitalGlobe	Uncorrected (Real-Time Flight)	58.84	14.53	23.70
DigitalGlobe	Corrected	64.71	8.22	27.42
GoogleMaps	Corrected	38.47	9.22	48.73

By analyzing the results from the database variants, it was determined that the most accurate database method for future real-time vision-aided navigation research was in fact the original database structure with attitude correction applied. The database variants were compared using several metrics, but the optimum variant was determined by minimizing the DRMS error of the attitude-aided 3DOF PnP solution. Secondary consideration was given to maximizing the percentage of features located on non-wooded regions of the database image. The original database was used for the remaining refinement tests of this thesis. The ideal database would produce both the most accurate solutions and the highest number of measurements per flight image set, however no single database met these requirements, and as such the most accurate database was determined to be the best.

4.6 Parameter Adjustment

Once the optimum database variant was determined, additional parameter adjustments were tested to further improve the performance of the navigation solution. The primary modification for improvements was adjusting the aerial ground pixel density (GPD). For the GPD adjustments, the aerial images were down-sampled to GPD similar to the database image. With both images being the same or nearly the same GPD, the features found with SIFT should ideally be the same scale in both images, potentially making matching easier. Figure 35 shows a sample of the down-sampled images. The down-sampling process was conducted using the OpenCV image pyramid to halve the resolution of the image until it nearly matched the GPD of the database. The DigitalGlobe database GPD was 12.72 pixels/m^2, and the GoogleMaps database GPD was 1.69 pixels/m^2. Table 8 details the change in resolution and GPD of the flight imagery for each step in the image pyramid.

Table 8. Ground Pixel Density Analysis for Steps in the Image Pyramid for Flight Test Imagery at an Altitude of 250 meters

Steps	**Resolution [pixels]**	**GPD [pixels /m^2]**
0	1600x1200	94.34
1	800x600	24.31
2	400x300	6.16
3	200x150	1.48

Figure 35. Image A was an image acquired during the flight tests, shown at its full resolution. Image B is the same image, but down-sampled 3 steps to be of comparable resolution to the DigitalGlobe database image.

Since down-sampling was achieved by use of image pyramids, algorithm refinement tests were run for each step in the pyramid until the resolution was comparable to the database image. For each step, the position error was characterized as well as the percentage of valid measurement recorded to determine if down-sampling improved the performance of the system in any significant manner. Table 9 details the numerical findings under the down-sampling adjustments for the database used for the Flight Tests. For comparison, the same pyramid steps were tested on refinement test runs with the GoogleMaps database.

Table 9. Results from refinement flight tests with various steps of down-sampling applied to the flight imagery. This test was run using the base database from the Flight Tests.

Steps	DRMS	MRSE	% Valid Measurements
0	8.219	10.184	26.528
1	8.457	10.101	25.037
2	9.189	13.394	26.677
3	8.862	10.207	25.037

Table 10. Results from refinement flight tests with various steps of down-sampling applied to the flight imagery. This test was run using the GoogleMaps database for comparison.

Steps	DRMS	MRSE	% Valid Measurements
0	9.216	10.598	48.286
1	9.769	13.535	49.031
2	8.908	10.105	48.584
3	8.716	9.818	47.392

Comparing Tables 9 and 10, it can be seen that down-samping did not greatly change the accuracy of the system for either database, nor did the percentage of measurements change drastically. For the base database tests all down-sampling steps slightly decreased, and the percentage of valid measurements also stayed nearly consistent with no significant improvements. For this database, there was no exact match of GPD using this down-sampling method, as the exact match would be between the first and second steps. No significant improvement was seen for the DigitalGlobe database. For the GoogleMaps database, three steps of down-sampling proved to produce the most accurate results for this database, but the improvement was not significant enough considered to be substantial.

From the down-sampling analysis, it was determined that down-sampling did not in fact greatly impact the results. The greatest improvement seen was the 0.5 meter DRMS from matching the GPD of the flight imagery to the GPD of the GoogleMaps database. For both databases, there were no significant changes to the percentage of valid measurement based on changing the GPD. This leads to the conclusion that the high resolution imagery obtained during flight was much higher resolution than necessary. Acquiring smaller resolution imagery, as considered in Section 5.3, would hypothetically lead to faster image acquisition and thus a greater rate at which measurements can be taken. Smaller image resolutions would also relate to faster image matching operations, but as shown in Table 4, the matching process took very little

time compared to the time required for image acquisition. Overall, image resolution matching database GPD appeared to produce slightly more accurate solutions, but the results were too miniscule to be considered a definitive improvement.

4.7 Chapter Summary

The results from the implementation flight tests were successful, obtaining real-time results with a position DRMS error of only 14.53 meters. After post-processing the data performing algorithm refinement tests, a position DRMS error of 8.22 meters was achieved. The subsequent chapter will provide final conclusions, as well as suggest future work to conduct on this platform for furthering the research.

V. Conclusions and Future Work

5.1 Conclusions

Vision-aided navigation is only possible due to the advancements in digital imaging in recent decades, allowing for high resolution imagery to be acquired at sufficient frequency and quality to be analyzed and used for navigation algorithms. The VisNav algorithm used for this thesis was the Perspective n Point algorithm, allowing absolute position solutions to be produced independent of previous measurements. In order to use the PnP algorithm, a feature database was constructed of orthorectified satellite and aerial imagery used to relate the 2D features located in the flight imagery to the 3D world feature location stored in the database. The database and algorithms were integrated into a COTS SUAS, and flown over Camp Atterbury, IN over a sequence of flight tests in Fall 2015.

The first Flight Test was focused on verifying the capabilities of the 12-ft wingspan ARF Telemaster during autonomous flight. The second test was to acquire flight imagery of the area of operations and test the integration of the image acquisition components with the airframe. During the third Flight Test, the system was proven to be capable of providing a real-time VisNav solution for a COTS SUAS. Analysis of the Flight Test 3 data showed that the system was capable of obtaining a solution roughly 32% of the time while flying a pattern over an urban environment with many buildings and other descriptive features observable. During this flight, the position of the SUAV was calculated with a DRMS error of 14.53 meters. This was accurate enough for verification of the real-time system, but a more accurate system was desired. In order to obtain this solution several alterations and improvements were conducted for the system.

The primary alterations to the system were correction for the attitude error, im-

provements to the database, and applying down-sampling to the flight images to make them comparable GPD to the database imagery. The attitude analysis found that the attitude error was approximately -2.6 degrees for roll and -1.6 degrees for pitch, and applying this correction increased the position accuracy of the system to 8.22 DRMS. Using a GoogleMaps image for the database doubled the percentage of valid measurements, but did not significantly improve the position accuracy of the system. After a final attitude correction and database were determined, down-sampling was applied in an attempt to further improve the system, but it was found to have no significant effect on either the accuracy or the percentage of valid measurements. This implies that using a lower resolution camera would not have reduced positioning performance, but could potentially increase the frame rate of the system by reducing the data transfer and image processing requirements.

The system proved to be capable of performing real-time absolute position calculations for a small unmanned aerial vehicle. The post-processing algorithm refinements improved the performance, allowing for a VisNav 3DOF position calculation with a DRMS of less than 10 meters. This research will lead to more reliable navigation for both piloted aircraft and UAVs in outdoor GPS-denied environments, providing a clear advantage in the NavWar realm even for small UAVs.

5.2 Future Work

As with most experimental research, there were many limitations to the tests that could be performed and as such limitations to the data gathered. The primary limitations were those involving the necessity of GPS for take-off, landing, and other crucial flight regimes. The autopilot, though capable of using an alternative navigation solution, such as the vision-aided solution presented in this thesis, utilized GPS for the flight tests for safety requirements, and as such the attitude outputs from the

autopilot were in fact partially based on the GPS. Even if a vision-aided solution were to be used instead, this is only possible at cruise altitudes. If the system had consisted of a full closed-loop to the autopilot, the attitude outputs would be based on the VisNav solution rather than GPS. The GPS was also required for initialization of any filters or tile estimation processes. The primary limitation to the sample rate is that of the camera interface. The python wrapper, Pymba, allowed for interaction with most of the camera parameters, but it greatly limited the rate at which a single image could be acquired. Measurements were then limited in number and frequency due to this interface.

Over the course of this research, several improvements were considered, but these were beyond the scope of this thesis. These are listed in the sections below.

Filtering and Smoothing.

The primary method proposed for improving the system is the implementation of a filter. The specific filter planned for this system was the EKF, as mentioned in Chapter 2. This implementation would not only potentially increase the accuracy of the system, but will provide position updates at the desired rate required for connecting to the UAV autopilot. Also with an EKF, there is a propagation of the navigation solution, and as such an estimated location can be calculated and used to find the proper tile for searching the feature database. Having this filter will limit the number of tiles required to be loaded into local memory at any given time.

Onboard Processor.

The second area of improvements involves the onboard processor— either increasing computation power or decreasing size. For use on SUAS, it is desired to reduce the size of the processor. One method for this would be to switch to a processor of

similar computation power and smaller size. Typically, reducing the size of the processor or on-board computer directly relates to reducing the size of data allowed for the system, so the GoogleMaps database is recommended to be used for this system, as the database was proven capable of nearly the same accuracy while being less than half the size of the original database.

Another suggestion for future work is to incorporate additional or alternate sensors, such as laser ranging equipment to guarantee accurate altitude measurements. From the analysis two primary issues were discovered with the Prosilica digital camera chosen for this research. The first being the difficult interface, limiting the rate of image acquisition. The issue revolved around the image resolution being much higher than necessary. These two issues combined led to the image acquisition process itself taking over half of the total time of the VisNav solution. A camera with a global shutter and adjustable parameters is required, but there are many cameras available on the market that meet these requirements while still having better interface programming.

Another simple alteration would be to consider different feature detection methods, such as SURF, FAST, FREAK, or CENSURE[4]. SURF was discussed in Chapter 2, and like the others suggested is boasted to perform faster than SIFT [28]. This would allow for faster measurement rate, allowing a faster real-time solution.

Bibliography

1. James A. Winnefeld and Frank Kendall, "Unmanned Systems Integrated Roadmap FY2013-2038," 2013.

2. Kyle Kauffman, "Stochastic Estimation and Control Estimation Theory," 2014.

3. Michael J. Veth, *Fusion of Imaging and Inertial Sensors for Navigation*, Ph.D. thesis, AFIT, 2006.

4. Davide Scaramuzza and Friedrich Fraundorfer, "Visual Odometry," , no. June, 2011.

5. David G Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, pp. 1–28, 2004.

6. Hugh Durrant-Whyte and Tim Bailey, "Simultaneous Localization and Mapping: Part I History of the SLAM Problem," 2006.

7. Han-Pang Chiu, Aveek Das, Phillip Miller, Supun Samarasekera, and Rakesh Kumar, "Precise vision-aided aerial navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. sep 2014, pp. 688–695, IEEE.

8. F. Samadzadegan; Gh. Abdi, "Vision-Based Pose Estimation for Autonomous Outdoor Navigation of Aerial Vehicles," *IEEE International Conference on Control, Instrumentation and Automation*, pp. 883–888, 2011.

9. DigitalGlobe, "My DigitalGlobe with EnhancedView Web Hosting Service User Guide," 2015.

10. Misra and Enge, *Global Position System (GPS): Signals, Measurements, and Performance*, Ganga-Jumuna Press, 2001.

11. Robert D. Putnam, "Air Force Future Operating Concept," pp. 1–48, 2015.

12. Timothy W Beard Randal W. McLain, *Small Unmanned Aircraft: Theory and Practice*, Princeton University PRess, 2012.

13. Jonathan Fabrizio and Jean Devars, "An Analytical Solution to the Persepective-N-Point Problem for common Planar Camera and for Catadioptric Sensor," *International Journal of Image and Graphics*, 2008.

14. Patrick J. Grandsaert, *Air force institute of technology*, Ph.D. thesis, AFIT, 2015.

15. Pengyu Guo Xin Li Yang Gui Xiang Zhou Hongliang Zhang Xiaohu Zhang, "Airborne Vision-Aided Landing Navigation System for Fixed-Wing UAV," *IEEE International Conference on Signal Processing*, 2014.

16. Frank Kendall, "Department of Defense Instruction: Cybersecurity," *Department of Defense Instruction*, vol. 5200.39, no. 5200, 2015.

17. Nicholas Lazaredes, "Ukraine's DIY drone war: Self-taught soldiers facing up to Russian-backed war machine," apr 2015.

18. Chaolei Wang, Tianmiao Wang, Jianhong Liang, Yang Chen, Yicheng Zhang, and Cong Wang, "Monocular visual SLAM for small UAVs in GPS-denied environments," *2012 IEEE International Conference on Robotics and Biomimetics, ROBIO 2012 - Conference Digest*, pp. 896–901, 2012.

19. Prashan Premaratne and Farzad Safaei, "Feature based Stereo Correspondence using Moment Invariants," *2008 4th International Conference on Information and Automation for Sustainability*, pp. 104–108, 2008.

20. Lt Col and K Fisher, "LKF and EKF," 2015.

21. Xiaodong Li, Nabil Aouf, and Abdelkrim Nemra, "Estimation analysis in VSLAM for UAV application," *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 365–370, 2012.

22. Davide Scaramuzza and Friedrich Fraundorfer, "Visual Odometry Part I: The First 30 Years and Fundamentals," *IEEE Robotics & Automation Magazine*.

23. John Raquet, "Vision-Aided Navigation," 2014.

24. Vadim Indelman, Pini Gurfil, Ehud Rivlin, and Hector Rotstein, "Real-Time Vision-Aided Localization and Navigation Based on Three-View Geometry," *IEEE*, vol. 48, no. 3, pp. 2239–2259, 2012.

25. Major Beich, J and Lt.Col Veth, M., "Tightly-coupled image-aided inertial relative navigation using Statistical Predictive Rendering (SPR) techniques and a priori world Models," in *IEEE/ION Position, Location and Navigation Symposium*. may 2010, pp. 552–560, IEEE.

26. Diomidis Katzourakis, Nikos I. Vitzilaios, and Nikos C. Tsourveloudis, "Vision aided navigation for unmanned helicopters," in *2009 17th Mediterranean Conference on Control and Automation*. jun 2009, pp. 1245–1250, IEEE.

27. Haiyang Chao, Yu Gu, and Marcello Napolitano, "A survey of optical flow techniques for robotics navigation applications," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 73, no. 1-4, pp. 361–372, 2014.

28. Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, 2008.

29. S. Rady, a. a. Kandil, and E. Badreddin, "A hybrid localization approach for UAV in GPS denied areas," *2011 IEEE/SICE International Symposium on System Integration, SII 2011*, pp. 1269–1274, 2011.

30. Taragay Oskiper, Han Pang Chiu, Zhiwei Zhu, Supun Samaresekera, and Rakesh Kumar, "Stable vision-aided navigation for large-area augmented reality," *Proceedings - IEEE Virtual Reality*, pp. 63–70, 2011.

31. Matthew B Rhudy, Haiyang Chao, and Yu Gu, "Wide-Field Optical Flow Aided Inertial Navigation for Unmanned Aerial Vehicles," , no. Iros, 2014.

32. T Bailey and H Durrant-Whyte, "Simultaneous Localization and Mapping (SLAM): Part II," *IEEE Robotics & Automation Magazine*, pp. 108–117, 2006.

33. Major M Veth and J Raquet, "Fusion of Low-Cost Imaging and Inertial Sensors for Navigation," .

34. Charles Murcott, Francois Du Plessis, and Johan Meyer, "A critique on previous work in vision aided navigation," *IEEE AFRICON Conference*, , no. September, pp. 13–15, 2011.

35. Hanchen Lu, Feng Zhang, and Jiang Wu, "A New Real Time Environment Perception Method Based on Visual Image for Micro UAS Flight Control," pp. 2515–2519, 2014.

36. M. W M Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, 2001.

37. M. Bryson and Salah Sukkarieh Salah Sukkarieh, "Active airborne localisation and exploration in unknown environments using inertial SLAM," *2006 IEEE Aerospace Conference*, 2006.

38. Yeongju Kim, Dongjin Lee, and Hyochoong Bang, "Vision-only UAV navigation aided by terrain elevation map," *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, pp. 1729–1733, 2012.

39. Vasko Sazdovski and Peter M. G. Silson, "Inertial Navigation Aided by Vision-Based Simultaneous Localization and Mapping," *IEEE Sensors Journal*, vol. 11, no. 8, pp. 1646–1656, aug 2011.

40. Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon, "Bundle Adjustment A Modern Synthesis," *Vision Algorithms: Theory and Practice*, vol. 1883, pp. 298–372, 2000.

41. Daniel DeMenthon and Larry S. Davis, "Exact and approximate solutions of the perspective-three-point problem," 1992.

42. R. Horaud, B. Conio, O. Leboulleux, and B. Lacolle, "An analytic solution for the perspective 4-point problem," *Proceedings CVPR '89: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 500–507, 1989.

43. Jean-Yves Bouguet, "Camera Calibration Toolbox for Matlab," .

44. K. Levenberg, "A method for the solution of some certain problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.

45. Itseez, "OpenCV," .

46. W.Y. Kong, G.K. Egan, and T. Cornall, "Feature Based Navigation for UAVs," *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3539–3543, 2006.

47. Duo-yu Gu, Cheng-fei Zhu, Jiang Guo, Shu-xiao Li, and Hong-xing Chang, "Vision-aided UAV navigation using GIS data," *Proceedings of 2010 IEEE International Conference on Vehicular Electronics and Safety*, pp. 78–82, 2010.

48. M Folk, A Cheng, and K Yates, "HDF5: A file format and i/o library for high performance computing applications," *Proceedings of Supercomputing*, vol. 99, 1999.

49. Google, "No Title," 2016.

50. Continuum Analytics Inc, "Anaconda," 2016.

51. Marius Muja and David G Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration," *International Conference on Computer Vision Theory and Applications (VISAPP '09)*, pp. 1–10, 2009.

52. Hobby Express, "12 Foot Telemaster ARF w/Drop Box (OVERSIZE)," .

53. "Pixhawk," .

54. Intel, "Intel NUC Board D53427RKE Technical Product Specification," .

55. Allied Vision Technology, "Prosilica GE 1660," .

56. Computar: Factory Automation Lenses, "M0814-MP," 2001.

57. Persistent Systems LLC, "Gen 4 Integration Unit," 2014.

58. NETGEAR Inc, "ProSAFE 5-port and 8-port Gigabit Switches," .

59. "RealVNC," .

60. Michael Oborne, "Mission Planner," .

61. "GDAL- Geospatial Data Abstraction Library," .

Appendix . Appendix

A Appendix A: Image Acquisition Test Script

```
#Script to Test time of image capture

from pymba import *
import time
import cv2
import numpy as np
import matplotlib.pyplot as plt
import datetime

#start Vimba
with Vimba() as vimba:
    #get system object
    system = vimba.getSystem()

    #list available cameras (after enabling discovery for GigE cameras)
    if system.GeVTLIsPresent:
        system.runFeatureCommand("GeVDiscoveryAllOnce")

    cameraIds = vimba.getCameraIds() #Obtains the ID for connected cameras, allowing choice of specific camera
    #get and open a specific camera
    camera0 = vimba.getCamera(cameraIds[0])
    camera0.openCamera()

    #set the value of a feature
    camera0.AcquisitionMode = 'SingleFrame' #Sets the camera to only acquire a single frame at a time

    #create new frames for the camera
    frame0 = camera0.getFrame()    #creates a frame
    frame1 = camera0.getFrame()    #creates a second frame

    #announce frame
    frame0.announceFrame()

    #capture a camera image
    camera0.startCapture()
    frame0.queueFrameCapture()
    camera0.runFeatureCommand('AcquisitionStart')
    camera0.runFeatureCommand('AcquisitionStop')

    frame0.waitFrameCapture() #Finalizes the Capture Process

    dtime = time.strftime("h%Hm%Ms%S") #Timing used in saving the image

    #get image data...
    imgData = frame0.getBufferByteData() #Not utilized for this thesis

```

```
47      ##...or use NumPy for fast image display
48      moreUsefullImgData = np.ndarray(buffer = frame0.getBufferByteData(),
49                                      dtype = np.uint8,
50                                      shape = (frame0.height,
51                                               frame0.width,
52                                               1)) #preferred method,
    allowing image processing in OpenCV
53
54      #clean up after capture
55      camera0.endCapture()
56      camera0.revokeAllFrames()
57
58      #close camera
59      imgRGB = cv2.cvtColor(moreUsefullImgData, cv2.COLOR_BAYER_GR2BGR) #
    Convert from BayerGR8 format to BGR
60
61      img = cv2.cvtColor(imgRGB, cv2.COLOR_RGB2GRAY) #Convert from BGR to
    Grayscale
62
63  # Show Image if desired
64  #     plt.imshow(img, cmap = 'gray')
65  #     plt.show()
66
67      fname = '/Image Test %s.png' %(str(dtime))
68      cv2.imwrite('./ImagesCaptured' +fname, img) #Saves image in
    designated directory
69      print fname
```

B Appendix B: Telemetry Read Test Script

```
from droneapi.lib import VehicleMode
from droneapi.lib import Command
from droneapi.lib import mavutil
from numpy import matrix
import numpy as np
import math
import time
from datetime import datetime

print "This is a really sillly test \n"
# Get a local APIConnection to the autopilot (from companion computer or
    GCS).
api = local_connect()

# Create vehicle objects for each vehicle from the APIConnection
v_stuff = api.get_vehicles()[0]
print "Vehicle Object Created \n"


#print read telemetry and print roll, pitch, and yaw from the autopilot
    every second for 10 seconds
cnt = 0
while cnt <= 9:
    try:
        yaw = v_stuff.attitude.yaw            #heading from north (6
    bytes CHECK)
        roll = v_stuff.attitude.roll
        pitch = v_stuff.attitude.pitch
        print "Telem: %s, %s, %s \n" %(str(roll), str(pitch), str(yaw))
        print "Count = ", cnt
        time.sleep(1)
        cnt += 1

    except KeyboardInterrupt:
        break;
```

C Appendix C: Image Acquisition with Telemetry Read Flight Test Script

```
from pymba import *
import time
import cv2
import numpy as np
import matplotlib.pyplot as plt
import time

from droneapi.lib import VehicleMode
from droneapi.lib import Command
from droneapi.lib import mavutil
import math
import datetime

def ImageCapture(Fltnum, Imgnum,v_stuff):
''' Image Capture function used for Flight Test Operations. This
    function utilizes the Pymba wrapper for connecting to an AVT
    Prosilica Camera and the droneapi scripts from MavProxy in order to
    connect to an autopilot.
 Inputs:
     Fltnum: The number used for specifying one Flight Objective. Used
    for organizing images
     Imgnum: The specific number assigned to the acquired image. Used
    for organizing images
     v_stuff: The mavproxy/droneapi vehicle object

 Outputs:
     img:  Greyscale image acquired
     pva:  GPS position and attitude data from the time the image was
    aquired
'''

#start Vimba
    with Vimba() as vimba:
        #get system object
        system = vimba.getSystem()

        #list available cameras (after enabling discovery for GigE
    cameras)
        if system.GeVTLIsPresent:
            system.runFeatureCommand("GeVDiscoveryAllOnce")
        cameraIds = vimba.getCameraIds() #Obtains the ID for connected
    cameras, allowing choice of specific camera

        #get and open a camera
        camera0 = vimba.getCamera(cameraIds[0])
        camera0.openCamera()


        #set the value of a feature
        camera0.AcquisitionMode = 'SingleFrame' #Sets the camera to only
```

```
    acquire a single frame at a time
43
44          #create new frames for the camera
45          frame0 = camera0.getFrame()     #creates a frame
46          frame1 = camera0.getFrame()     #creates a second frame
47
48          #announce frame
49          frame0.announceFrame()
50
51          #capture a camera image
52          camera0.startCapture()
53          frame0.queueFrameCapture()
54          camera0.runFeatureCommand('AcquisitionStart')
55
56          camera0.runFeatureCommand('AcquisitionStop')
57          dtime = time.strftime("%H%M%S")
58
59  #Telemetry read immediately after AcquisitionStop to reduce time
      between telem and image acquisition
60  imgtime = datetime.datetime.now()
61  #Getting Attitude Info
62  yaw = v_stuff.attitude.yaw              #heading from north (6 bytes
      CHECK)
63          roll = v_stuff.attitude.roll
64          pitch = v_stuff.attitude.pitch
65
66  #Get Position Data
67  lat=str(v_stuff.location.lat)           #latitude (9 bytes CHECK)
68          lon=str(v_stuff.location.lon)           #longitude (9 bytes CHECK
      )
69          alt_asl = str(v_stuff.location.alt)     #altitude above sea level
       (6 bytes CHECK)
70          pva = np.array([(imgtime), Imgnum, lat, lon, alt_asl, roll,
      pitch, yaw, (dtime)])
71
72          frame0.waitFrameCapture() #Finalizes the Capture Process
73
74          #get image data...
75          imgData = frame0.getBufferByteData() #Not utilized for this
      thesis
76
77          ##...or use NumPy for fast image display
78          moreUsefullImgData = np.ndarray(buffer = frame0.
      getBufferByteData(),
79                                          dtype = np.uint8,
80                                          shape = (frame0.height,
81                                                   frame0.width,
82                                                   1)) #preferred method,
      allowing image processing in OpenCV
83
84          #clean up after capture
85          camera0.endCapture()
```

```
86          camera0.revokeAllFrames()
87          #convert image to Grayscale
88          imgRGB = cv2.cvtColor(moreUsefullImgData, cv2.COLOR_BAYER_GR2BGR
    )
89          img = cv2.cvtColor(imgRGB, cv2.COLOR_RGB2GRAY)
90
91      #Save Image
92          fname = '/Flt%03d Sett%03d Img%03d Time %s.png' %(Fltnum, Setnum
    , Imgnum, str(dtime))
93          cv2.imwrite('./ImagesCaptured' +fname, img)
94  return img, pva
```

D Appendix D: Database Variants with Feature Locations

The database variants discussed in Chapter 4 are shown below. The SIFT feature locations are shown in blue for each image as a qualitative analysis of the usefulness of database for VisNav solutions.

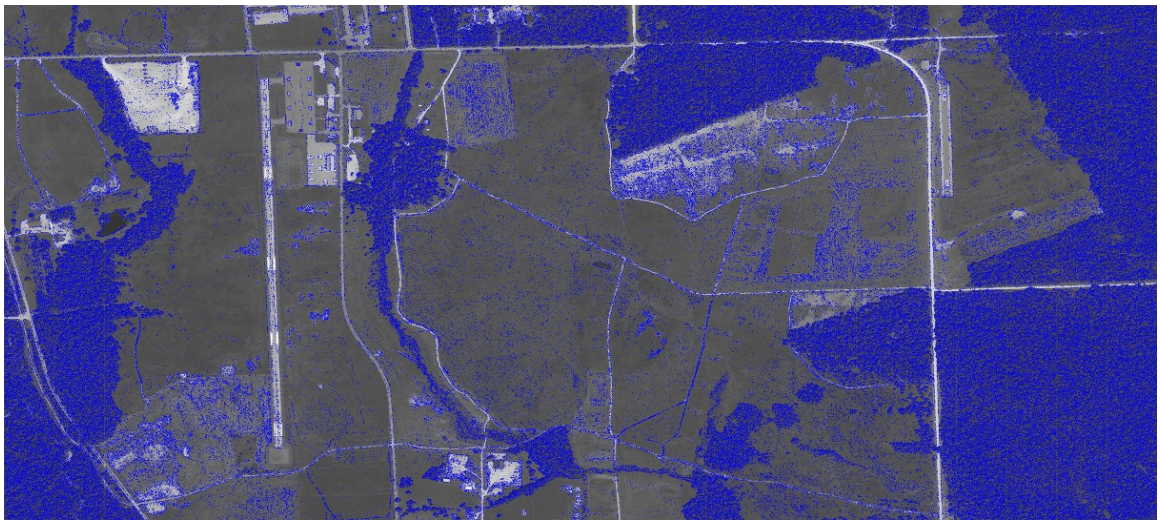

Figure 36. The database variant for the Contrast Threshold set at 0.02 instead of the base of 0.04. Feature locations are shown in blue.

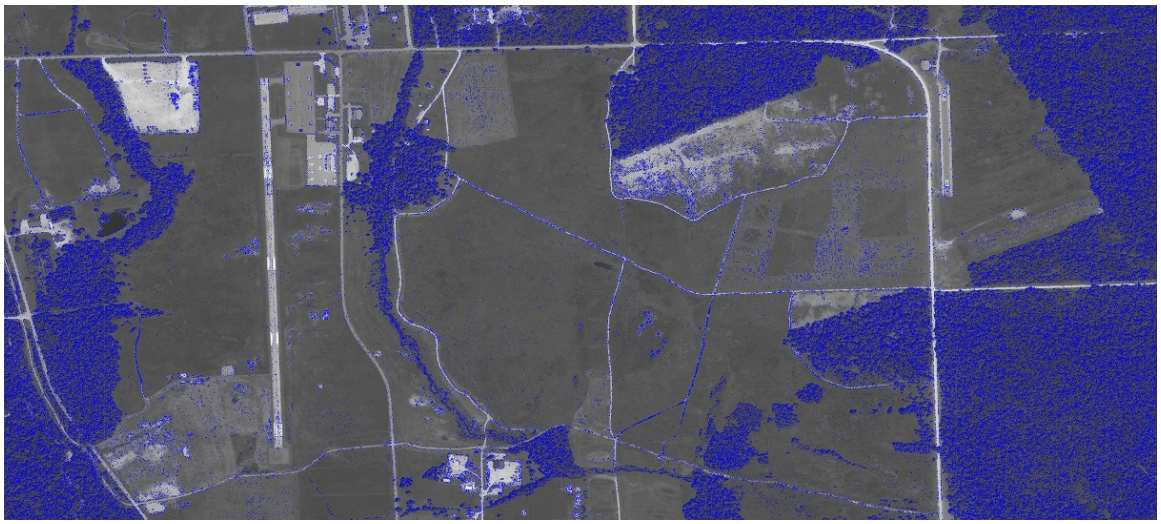

Figure 37. The database variant for the Contrast Threshold set at 0.03 instead of the base of 0.04. Feature locations are shown in blue.

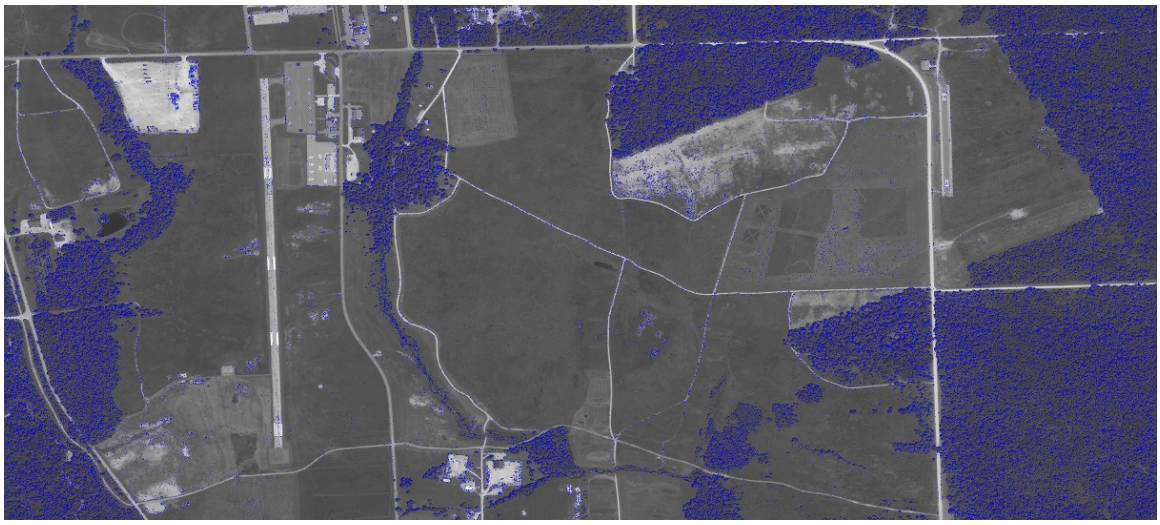

Figure 38. The database variant for the Edge Threshold set at 5 instead of the base of 10. Feature locations are shown in blue.

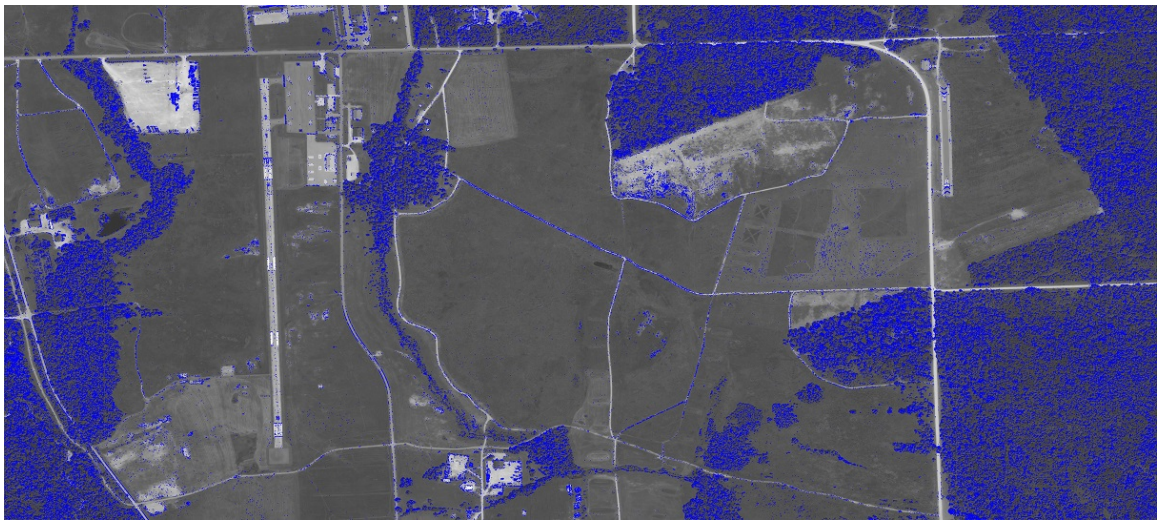

Figure 39. The database variant for the Sigma Parameter set at 0.8 instead of the base of 1.6. Feature locations are shown in blue.

Figure 40. The database variant for the Contrast Threshold set at 0.08 instead of the base of 0.04. Feature locations are shown in blue.

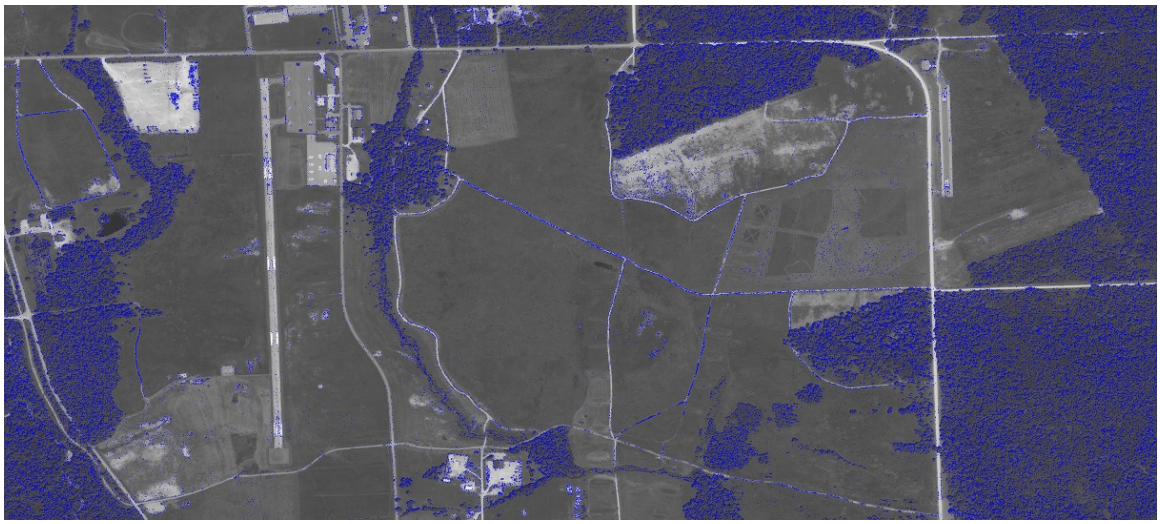

Figure 41. The database variant for the Edge Threshold set at 15 instead of the base of 10. Feature locations are shown in blue.

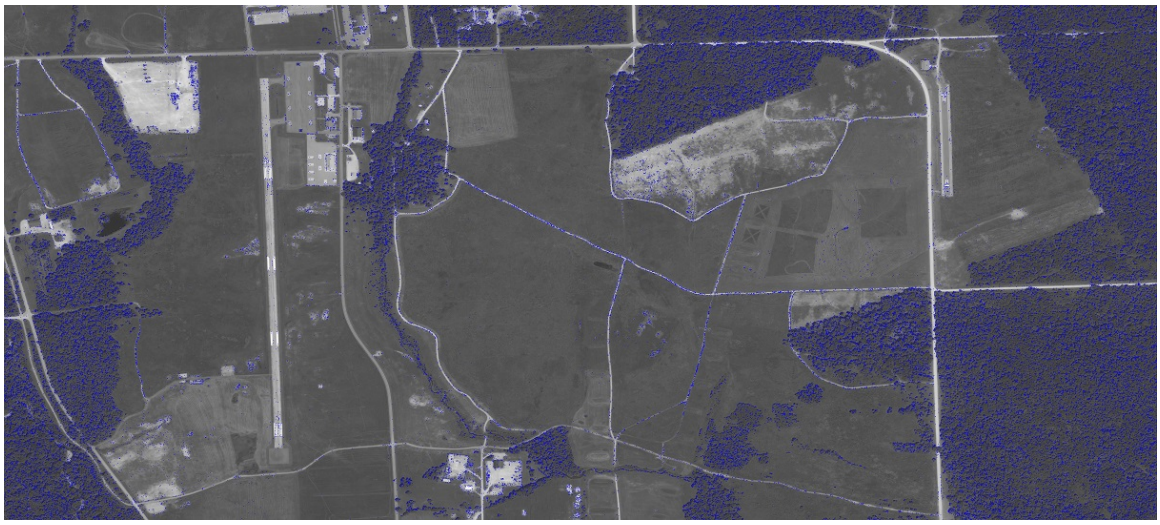

Figure 42. The database variant for the Contrast Threshold set at 0.02 instead of the base of 0.04. Feature locations are shown in blue.

E Appendix E: Pre-Flight Checklist with Calibration Processes

This Appendix provides a basic overview of the pre-flight process used for vision-aided navigation flight tests conducted at AFIT. The individual steps can vary for specific platforms, but the overall process should remain the same. For clarification, there are two seperate components considered for pre-flight: the payload (including camera, onboard computer, and associated groundstation) and the UAV (the specific airframe, autopilot, and associated groundstation). Steps will be laid out for each component in the process.

Table 11. UAV Airframe Setup

1. Assemble UAV	Install wings and make all wiring connections. Install propeller. Do not connect batteries or fuel yet	
2. Inspect UAV	Check props and hub for damage for fatigue, Inspect servo arms and flight control surfaces for damage. Tighten assembly as needed. Ensure pitot connection are made. Check Center of Gravity (CG) location.	
3. Install Fully Charged Batteries (for ignition and payload)	Connect battery cable.	
	Adjust batteries and/or weight/ballast position as necessary to ensure proper placement of CG.	
	Ensure batteries are properly secured.	
4. Fill with appropriate amount of gas	Support contractor is responsible for all storing, pumping and handling of SUAS fuel.	

Table 12. Payload Initial Setup

1. Verify onboard Computer Operations	Connect NUC to ground station power. Connect monitor. Turn on NUC and Bluetooth mouse and keyboard. Ensure system powers up correctly and OS is responding as expected. Double check necessary scripts are loaded into the correct directories. Plug NUC to GCS2 Ethernet and ensure successful Remote Desktop operations.	
	Power off NUC and Bluetooth mouse and keyboard. Unplug monitor from NUC. Unplug NUC from Ground station power.	
2. Assemble Payload	Install pre-flight payload batteries in box. Connect Payload Pixhawk connection (if using separate Pixhawk for payload recording). Do not connect batteries yet.	
3. Inspect Payload	Check all connection of component cables and fasteners. Tighten assembly as needed. Remove lens cap on Prosilica camera.	
4. Install Fully Charged Battery	Connect battery cable. Start pre-flight battery timers.	
	Ensure batteries are properly secured.	

INTRINSIC CAMERA CALIBRATION

Intrinsic camera calibration can be conducted at this point of the pre-flight checklist or prior to starting the pre-flight checklist. If conducted prior to the pre-flight, special care must be given to the lens and camera parameters that nothing changes from the time of calibration to the time of take off. The intrinsic camera calibration process is given by the following link:

`http://www.vision.caltech.edu/bouguetj/calib_doc/`

From this site, download the MATLAB calibration toolbox. This is the GUI used to control the scripts for the intrinsic camera calibration. Use the **"Doing your own calibration"** section to generation the necessary calibration board and images of the board. These will be used to calibrate the camera matrix and distortion parameters used in the VisNav solution. Once calibration images are acquired, use the **"First Calibration Example - Corner extraction, calibration, additional tools"** section to perform the actual calibration process using the MATLAB calibration toolbox. The basic steps for this are:

1. Load Images
2. Extract Corners
3. Run Calibration

4. Analyze Error
5. Refine Solution:
    - adjust winx, winy
    - remove problematic images from calibration
    - re-extract corners
6. Re-Calibrate
7. Repeat process until *pixel error* is within tolerable limits

The *pixel error* is the primary metric used for refinement, and it is desired to minimize this. Repeat steps 3 through 6 until this error is determined small enough. This criteria is based on the image size, so for a very high resolution image, an error of a few pixels is acceptable, but for lower resolution images, lower pixel errors are desired. From the intrinsic camera calibration, the camera matrix **K** and the camera distortion parameters are calculated.

Table 13. Payload Continued Setup

1. Insert Payload box into Aircraft	Secure payload box into aircraft bay. Attach Wave Relay cable to Wave Relay node inside the aircraft fuselage. Connect Serial USB from Pixhawk to NUC. Ensure proper LEDs are lit on the Ethernet switch and the USB cable.	

Table 14. UAV Autopilot Setup

1. Prep Transmitter	Power on transmitter and set to "manual".	
2. Level UAV	Level the UAV on the leveling stand.	
3. Power on Autopilot Board	If no power switch is installed, you must disconnect then reconnect the battery. Arm the Pixhawk autopilot using the arming button.	
	Continue to keep the UAV level until autopilot start up and initialization is complete and leveling has been performed. Monitor audio tones and LEDs to ensure nominal autopilot status is achieved.	

Table 15. UAV Communications/ Ground Control Station Setup

1. Establish Communications	Connect to autopilot using Mission Planner. Observe signal quality on flight data window – should be greater than 95% for co-located ground station. Perform comm. ground check as necessary to ensure proper range performance for autopilot comm. and RC receiver.	
2. Verify flight data is being communicated and GPS lock is achieved	Using the flight data window on Mission Planner, ensure flight data is being transmitted, and observe GPS status.	
3. Verify Communication Link between Safety Pilot Controller and UAV	Input full deflection to individual servos on the controller and verify corresponding movement of control surface on UAV. Verify proper mode switching using RC controller.	
4. Ensure Proper Gains Loaded	Check to make sure the correct gains are loaded for the UAV you are flying (observable on gain tuning windows).	
5. Load Waypoints	Ensure proper waypoints are loaded and that a rally point (return to launch location) is loaded. RALLY POINT IS REQUIRED FOR LOST COMMUNICATIONS SCENARIO.	
6. Perform ground calibration	While aircraft is still on leveling stand, perform single axis accelerometer calibration using Mission Planner.	
	Ensure compass use is check (Pixhawk uses digital airspeed connection through I2C port) and calibrate radio as necessary. Check compass status in HUD window.	
	Ensure airspeed sensor use is checked (Pixhawk uses digital airspeed connection through I2C port). Cup with hand or place empty bottle around pitot tube (not touching pitot tube) and, using preflight calibration command on flight data window, zero pressure to establish ground level in altimeter. Observe altitude and airspeed indications on HUD (both should be approximately zero).	
	Tap or blow on pitot tube to ensure that it is properly reading changes in pressure. Observe response on HUD airspeed indicator.	

Table 16. Payload Communications/ Ground Control Station Setup

1. Establish Communications	On ground station, load Wave Relay in brouser. Ensure connection too airborne Node. Open RealVNC, and connect to NUC IP address. Establish Remote Desktop operations. Connect MAVProxy to aircraft Pixhawk.	
2. Ensure Proper Scripts Loaded	Check to make sure the necessary scripts are loaded properly and ready for continuous operations. Run sample test scripts listed below. Check images captured using build in image-viewer and telemetry read-outs. TEST SCRIPTS: ImgTest.py, TelemTest.py, DAQTest.py	
3. Prepare Scripts	Ensure interface is prepared for EXTRINSIC CAMERA CALIBRATION and in-flight tests.	

EXTRINSIC CAMERA CALIBRATION

Once the UAV autopilot is calibrated for flight, the extrinsic parameters for the camera must be calibrated. This is done to calibrate the rotation matrix from the camera to the body frame, as described in Section 4.4 of this thesis. The camera's position is estimated in an ideal situation, and then calibrated by applying roll and pitch corrections. While the aircraft is still in the leveling stand, follow the process described below to calibrate the extrinsic parameters for the camera.

The calibration lens cap is a lens cap with a plumb line attached to the exact center of the cap. This is used to find the point directly below the camera in the direction of gravity, which is the same direction used by the autopilot calibration to determine the downward direction. The extrinsic camera calibration board, unlike the intrinsic calibration board, consists of a simple white board with a single dark circle in the center. This circle should be large enough to be seen in an image when taken under the leveling stand, but small enough for the center pixel location to be calculated in said image.

Table 17. Extrinsic Camera Calibration Process

1. Obtain Ideal Image Origin	Place the calibration lens cap onto the camera. Place the extrinsic camera calibration board such that the circle point in the center is directly under the plumb line. Remove the calibration lens cap.	
2. Obtain Actual Image Origin	Ensure the extrinsic calibration board has not been disturbed. Using the image acquisition test process, acquire an image of the extrinsic calibration board.	
3. Perform Extrinsic Calibration Scripts	Run the python calibration scripts on the NUC. These will calculate the difference between the two image origins and use the known height of the leveling stand and the differences in the x and y to determine the roll and pitch correction. These are applied to the ideal rotation matrix and the calibrated matrix saved in the appropriate YAML files used for the VisNav solution.	

Table 18. UAV Take Off

1. Check Control Surfaces	Remove UAV from leveling stand. Safety pilot check for full and correct deflection of all control surfaces using RC radio.	
2. Check Stabilize Mode	Switch transmitter to stabilize mode.	
	Have assistant pitch, yaw, and roll the UAV and ensure that the control surfaces move in the proper direction to stabilize the UAV	
3. Obtain Clearance	Contact the field controller and obtain clearance to launch the UAV.	
4. Launch	Roll aircraft to launch point.	
	Have Assistant 1 stand in front of tail/rudder to prevent forward movement.	
	ENSURE ALL PERSONNEL ARE CLEAR AND WARNED PRIOR TO ENGAGING PROPELLER. Pull ignition pin to apply power to ignition circuit. ENSURE ASSISTANT 2 KNOWS HOW TO KILL ENGINE WITH RC CONTROLLER. Assistant 2 holds Safety Pilot RC controller, while Safety Pilot starts engine. Immediately after engine start, safety pilot takes control of RC controller, and checks throttle and engine operation while Assistant 1 continues to prevent forward motion of the aircraft. When ready to launch, Safety Pilot ensures all test team members are ready for launch. Once all are ready, Assistant 1 steps away from the aircraft, and safety pilot executes a manual take-off.	

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE (DD–MM–YYYY)	2. REPORT TYPE	3. DATES COVERED *(From — To)*
24-03-2016	Master's Thesis	Sept 2014 – March 2016

4. TITLE AND SUBTITLE
Real-Time Implementation of Vision-Aided Monocular Navigation for Small Fixed-Wing Unmanned Aerial Systems

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Machin, Timothy I, 2d Lt

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENG-MS-16-M-035

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

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

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A. Approved for Public Release; Distribution Unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The goal of this project was to develop and implement algorithms to demonstrate real-time positioning of a UAV using a monocular camera combined with previously collected orthorectified imagery. Unlike previous tests, this project did not utilize a full inertial navigation system (INS) for attitude, but instead had to rely on the attitude obtained by inexpensive commercial off-the-shelf (COTS) autopilots. The system consisted of primarily COTS components and open-source software, and was flown over Camp Atterbury, IN for a sequence of flight tests in Fall 2015. The system obtained valid solutions over much of the flight path, identifying features in the flight image, matching those features with a database of features, and then solving both the 6DOF solution, and an attitude-aided 3DOF solution. The tests demonstrated that such attitude aiding is beneficial, since the horizontal DRMS of the 6DOF solution was 59m, whereas the 3DOF solution DRMS was 15m. Post processing was done to improve the algorithm to correct for system errors, obtaining a 3DOF solution DRMS of 8.22 meters. Overall, this project increased our understanding of the capabilities and limitations of real-time vision-aided navigation, and demonstrated that such navigation is possible on a relatively small platform with limited computational power.

15. SUBJECT TERMS
Vision-Aided Navigation, Monocular, UAV, PnP

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. John F. Raquet, AFIT/ENG
U	U	U	UU	122	19b. TELEPHONE NUMBER (Include Area Code) (937) 255-3636 x4580 John.raquet@afit.edu