Methods for Improving Visual Terrain Relative Navigation for Dynamic Aerial Systems

by

Matthew Castleberry

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
April 30, 2021

Keywords: GPS-Denied Navigation, Image Processing, Terrain Relative Navigation, Remote Sensing, Simulation

Approved by

Thaddeus Roppel, Chair, Associate Professor of Electrical Engineering
Scott Martin, Co-chair, Assistant Research Professor of Mechanical Engineering
David Bevly, Professor of Mechanical Engineering
Stanley Reeves, Professor of Electrical Engineering

Abstract

One popular means of aerial localization and navigation in GPS-denied environments is visual terrain relative navigation. Terrain relative navigation involves performing image registration with sensed aerial camera imagery and georeferenced satellite maps to produce the geographic translation and rotation of the camera. One popular terrain relative navigation technique depends on matching feature descriptors. These features, however, are intolerant to major changes in perspective, light, vegetation, season, and other scene changes and produce excessive amounts of false matches. Alternatively, image correlation can be used for registering a sensed image to a reference image but is extremely intolerant to perspective differences for 6 degree of freedom camera systems.

This research explores the use of a combination of corner detection and normalized cross correlation for aerial vehicles at different altitudes. New methods for using dynamic search windows within reference satellite imagery are explored to constrain the pose estimation and increase image matching accuracy. The algorithm is tested with both simulated aerial imagery and experimentally sensed imagery captured with rigid mounted cameras on unmanned aerial vehicles and high altitude balloons. It is evaluated on its successful match rate and pose estimation error compared to GPS. It has approximately 75% successful match rate in simulation and 20% successful match rate in experimental datasets. The filtered pose estimate error is decreased in simulation in effectively over 95% of the frames and 20% in the experimental cases. Integration of this algorithm with other navigational sensors and algorithms would provide improvements to the overall navigation solution.

Acknowledgments

I am very grateful for the amazing opportunity to study at Auburn University to earn a Bachelors and Masters in Electrical Engineering. I enjoyed the chance to perform research in the GPS and Vehicle Dynamics Lab (GAVLAB). It has proven to be a great environment to study navigational algorithms and set me on a career path to perform research in areas that I am passionate about.

Thank you to the many members of the GAVLAB who have helped me through grad school and the writing of this thesis. Thanks to Sam Douglass and Dr. Chen for encouraging me to start writing my thesis as early as I can. Thanks to Dan Kamrath and Jake Pryor for helping proofread this thesis. Also special thanks to Dan, Jake, Tanner, and Dr. Chen for assisting with the testing and launch of the high altitude balloon. Thanks to Amy Strong for being a great co-worker and encouraging me as I completed this thesis.

I am very grateful for my research sponsor IS4S and the continued support they provided. Thanks to Leo Richard and Rob Daily for the helpful direction, support, and guidance. The research presented in this thesis would not be possible without your help. I would also like to thank Ben Thompson and Dustin Doggett for the amazing summer internship at Dynetics despite the start of a global pandemic. The internship helped provide some valuable insight on the utility of simulation for aerial vision nav.

I would like to give special thanks to the property owner of the landing sight of the high altitude balloon. His generous time and effort to hike through some difficult terrain to help recover the payload made it possible to recover the data for this thesis.

Thank you to Garon Griffiths and his leadership in directing the new Auburn Engineering Makerspace and allowing me to help setup a state-of-the-art electronics workshop that proved to

be instrumental in the construction of the high altitude balloon. Thanks to the fellow maker assistants like Jose Arquitola and Joy Simmons for assisting me in training students and organizing the electronics shop.

In addition, I am very appreciative of all of my friends from the Auburn Wesley Foundation community. I am thankful for all of the love and support as I finished up my last years of undergrad and completed my Masters.

Most notably, I would like to thank my family who has supported me in my education from childhood through my Masters and has always been there for me when I need it. Thank you to my mom for encouraging me to be the best that I could be. Thank you to my dad for inspiring me to become an engineering and providing helpful guidance along the way.

Finally, I would like to thank my committee. Thank you to Dr. Roppel for being a good mentor during undergrad through my involvement in the Student Projects and Research Committee (SPARC) and guidance during my journey into grad school. Thanks to Dr. Reeves for teaching me about signal processing and image processing. Thank you to Dr. Bevly who has directed the GAVLAB and helped secured sponsor funding for my research. Lastly, thank you to Dr. Martin who has helped mentor me and direct my research involvement in the GAVLAB.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3DEP	3D Elevation Program
AFOV	Angular Field of View
APRS	Automatic Packet Reporting System
EKF	Extended Kalman Filter
ESA	European Space Agency
DEM	Digital Elevation Map
DoD	Department of Defense
EDL	Entry, Descent, and Landing
ENU	East North Up
FAA	Federal Aviation Administration
FCC	Federal Communications Commission
FOV	Field of View
FPGA	Field Programmable Gate Array
GAVLAB	GPS and Vehicle Dynamics Laboratory
GIS	Geographic Information System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HAB	High Altitude Balloon
HD	High Definition
I^2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
ITAR	International Traffic in Arms Regulations

JPL	Jet Propulsion Laboratory
MSL	Mean Sea Level
NASA	National Aeronautics and Space Administration
NED	North East Down
P3P	Perspective-3-Point
PnP	Perspective-n-Point
RANSAC	Random Sample Consensus
ROS	Robot Operating System
SIFT	Scale Invariant Feature Transform
TRN	Terrain Relative Navigation
UART	Universal Asynchronous Receiver-Transmitter
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
USGS	United States Geological Survey

Chapter 1: Introduction

Global navigation is a problem humanity has studied for millennia with its origins in astronomy for naval navigation. Today, global navigation is primarily reliant on the Global Navigation Satellite System (GNSS). One part of GNSS is the Global Positioning System (GPS). GPS is increasingly known to be susceptible to jamming, spoofing, and other forms of disruption. As more satellite constellations are being launched by various companies and nations, it is essential that alternative forms of navigation be developed for redundancy and strategic advantage for defense purposes. In addition, as humans are planning to return to the moon and set foot on Mars in the coming future, alternative forms of navigation are essential where navigational satellite constellations do not yet exist. One form of GPS-denied navigation is visual navigation. There are two main types of aerial visual navigation: relative and global. Relative visual navigation such as visual odometry and optical flow compare successive frames to provide an estimated location based on previous movements. These methods are susceptible to drifting, whereas global visual navigation uses maps and other data sources to match the current sensed scene to a geo-referenced source. This produces an absolute position fix that does not drift over time. This method is often called geo-registration or terrain relative navigation (TRN).

1.1. Problem Definition

Terrain relative navigation is easiest in areas rich with distinct features and for aerial platforms with a stabilized camera. For many applications, these luxuries are not available. Lower altitude imagery has less distinct landmarks and is more susceptible to seasonal changes causing large position errors over certain areas. Higher altitude imagery is further from the terrain reference potentially causing more pose uncertainty. By using camera gimbals, the imagery is

considered stabilized and can be pointed orthogonal to the ground. TRN for this type of application is a 3 Degree of Freedom (3DOF) problem. Aerial systems such as missiles, planetary landers, parachute-payloads, and high altitude balloons can be restricted from using gimbals to maintain the camera orthogonal to the ground due to weight, size, cost, or form factor requirements. When cameras are non-orthogonal to the ground, the imagery appears warped compared to the satellite maps. TRN for these applications is a 6DOF problem requiring more complex analysis to accommodate the additional uncertainty.

There are two different scopes for terrain relative navigation: coarse tracking and fine tracking. This research focuses on the latter. A coarse tracking system is used when the pose is only known to be within a given region that could be kilometers wide. Most coarse tracking systems assume a 3DOF system to narrow the results. A fine tracking system takes an input of a coarse pose and an estimated error to produce a pose estimation within a few meters in translation and orientation in a few degrees of truth.

1.2. Related Work

Aerial terrain relative navigation is a well-studied research area in the recent two decades with the improvements in computer vision algorithms, cameras, and processing hardware. NASA, for the landing of the Spirit and Opportunity Mars rovers, used a combination of corner detection and cross-correlation to identify features on the Martian surface [1] for navigation in 2004. More details on algorithms related to NASA's approach is described in Section 2.2.3. Another notable application of terrain relative navigation algorithms was demonstrated on an unmanned aerial vehicle (UAV) in 2008 by Conte and Douherty using edge detection coupled with visual odometry [2]. In 2015, researchers in China used the histogram of oriented gradients of the sensed and reference images along with a particle filter to match the images [3]. In 2016, Venable from the

US Air Force Institute of Technology used feature descriptors to compare the sensed imagery from fixed-wing UAVs to reference map features stored in a database.

When humans look at aerial images the first things recognized are features like roads, buildings, water, and vegetation. It could be useful to train an algorithm to recognize these features to aid in image matching with deep learning and neural networks. In 2010, a university in Sweden produced results of training a neural network to semantically segment the aerial images into "super pixels" that can be matched via a class histogram [4] instead of using feature based detectors. In 2018, Nile University produced some results by using a deep convolutional neural network based framework that classifies and matches buildings for urban geo-localization [5]. While this method works well in urban environments, rural areas where few buildings are present this would be insufficient. In 2019, a university in Brazil compared a method using a convolutional neural network and a multilayer perceptron neural network for edge matching as opposed to using feature-based detectors [6]. While much recent research is in the deep learning approaches to this problem, deep learning is not applicable to all applications due to restrictions in computation hardware and limited training datasets.

1.3. Contributions

The focus of the research presented is the development of terrain relative navigation using visual imagery at a variety of altitudes by using a rigidly-mounted downward facing camera in simulation and various aerial platforms with a translational accuracy that is similar in magnitude to modern GPS systems. To that end, the following contributions are made:

- Global positioning algorithms are developed to match sensed imagery to a georeferenced satellite image map using a combination of corner detection and cross-correlation based on prior art.

- A novel method of dynamic window sizing for cross-correlation is derived to allow the estimated error range of the window to be specified in world coordinates.
- A low-overhead method of simulating camera imagery to test vision-based navigation is implemented to verify the performance and accuracy of the navigational algorithms.

1.4. Thesis Outline

Chapter 2 is a discussion of the necessary background in the development of image matching algorithms. Chapter 3 reviews some of the basics of geographic systems and navigational sensors. Chapter 4 defines the simulation and localization algorithms used. Chapter 5 provides an overview of how the experimental data is collected with a UAV and high altitude balloons. Chapter 6 presents the performance of the algorithms on the simulated data and Chapter 7 presents the performance on the experimental data. Finally, Chapter 8 provides conclusions from the work presented here and provides direction for future work in aerial terrain relative navigation. Additional information on how the high altitude balloon payloads are constructed, tested, and their performance is documented in Appendix A and Appendix B.

Chapter 2: Image Matching with Computer Vision

Map-matching with images involves comparing a "sensed" image from a camera mounted to an aircraft compared with a known "reference" image from a satellite or another aircraft. This chapter discusses the basics of image processing and matching to compare two images.

2.1. Multiple View Geometry and Image Processing

As humans, we have two eyes that can convert electromagnetic waves in the visible spectrum into useful information. This information is based on the light emission and reflection of objects in a set field of view. The brain can interpret this information to understand color, size, distance, orientation, and type of object. A camera serves a similar function to human eyes and is precisely designed to capture light information in a specific way depending on application. To use camera information for navigational purposes, algorithms are designed around the physics of how cameras capture information.

2.1.1. Pin-hole camera model and image representation

Light always travels in a straight line until it hits a new material and refracts, reflects, or is absorbed. An ideal pin-hole camera is a device where the aperture that lets light pass through is infinitesimally small. The light then is projected onto a flat plane, film, or sensor. When observing the projection, the image is upside down to preserve the straight line of travel of light as shown in Figure 2.1. The quantity f represents the focal length, the distance between the imaging plane and the aperture. The larger the focal length, the larger the projection size. In modern cameras, the focal length is typically measured in millimeters (mm).

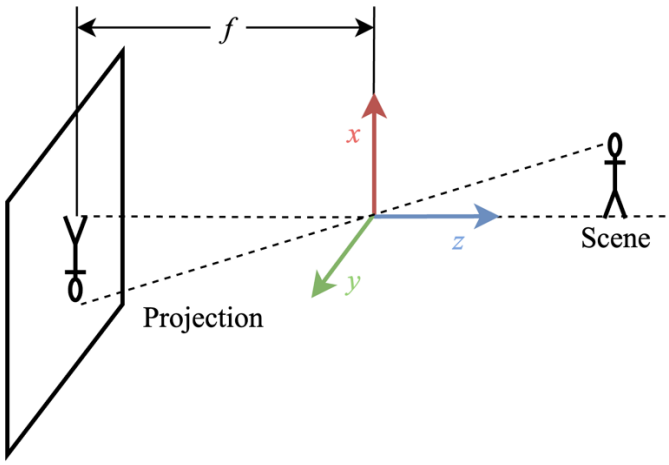

Figure 2.1: A diagram of the pin-hole camera model where an object is projected through a small aperture and thin lens to a flat plane.

Typically, an aperture has a circular hole for light to travel through. This means the image projected if unobstructed would be circular. Conventionally images are square or rectangular because image sensors are square or rectangular. Therefore, an image projection size is based on its focal length and image sensor size. This property is referred to as field of view (FOV). An image sensor is responsible for converting the analog intensities into discrete blocks of values that can be interpreted by a computer or saved onto a digital storage medium for later use. Each of the blocks is called a pixel and has intensity data encoded in one or more color channels as a numerical value scaled relative to a maximum value, usually a power of two. For example, an 8-bit high-definition (HD) grayscale image can be represented as a 1080×1920 matrix where each pixel has a value from 0 to $2^8 = 255$. The coordinate system convention of an image is defined in

Figure 2.2 where an image has M rows and N columns of pixels.

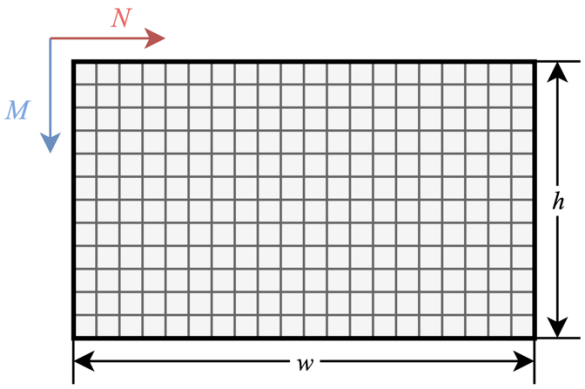

Figure 2.2: A diagram of a typical image sensor. The quantities w and h represent the width and height of the sensor in mm. The axis M and N represent the rows and columns of pixels.

2.1.2. Lenses and focal length

The pin-hole model is a great for understanding camera geometry and basic physics of light, however it has limitations especially with respect to zooming. A lens refracts light from an object in a scene to the image sensor as depicted in Figure 2.3. D_1 is the distance between the camera lens and the scene. D_2 is the distance between the lens center and the image sensor. The focal length, f, is distance between the lens and the focal point. By changing the lens shape or using a combination of lenses, the focal length can be adjusted to provide a different zoom level. This also changes the value of the angular field of view (AFOV), θ.

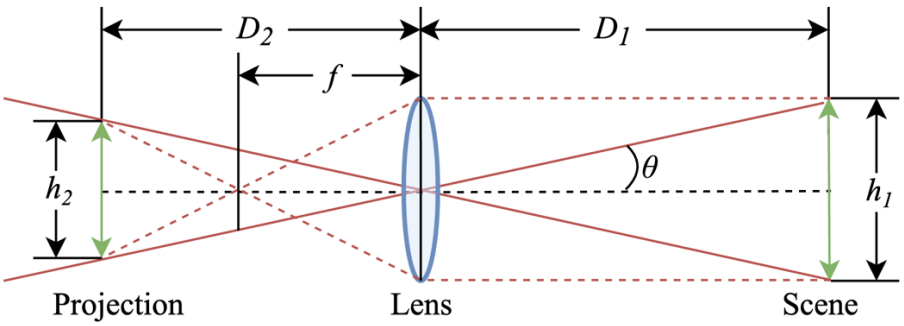

Figure 2.3: Diagram of light travelling from the scene to a projection on image sensor through a lens.

Given a set focal length and sensor size, the AFOV can be determined. For an image to be in focus, $f = D_2$. Using trigonometry, the following relationship can be described.

$$\tan(\theta) = \frac{h_2/2}{f} \tag{2.1}$$

Solving for θ

$$\theta = \tan^{-1}\left(\frac{h_2/2}{f}\right) \tag{2.2}$$

Therefore, the horizontal field of view and vertical field of view are the following, where w_2 and h_2 are the sensor size in mm.

$$\text{HFOV}(^\circ) = \tan^{-1}\left(\frac{w_2/2}{f}\right) \tag{2.3}$$

$$\text{VFOV}(^\circ) = \tan^{-1}\left(\frac{h_2/2}{f}\right) \tag{2.4}$$

2.1.3. Camera Intrinsic and Extrinsic Parameters

To simplify camera calculations, individual parameters can be arranged into matrices. There are two distinct categories of parameter: intrinsic and extrinsic. Intrinsic parameters are based on the internal properties of the camera. Extrinsic parameters are based on the overall position and orientation of the camera. The following is the camera intrinsic matrix.

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

The focal length can be split into the components by dividing by the pixel size in mm like $f_x = f/p_x$ and $f_y = f/p_y$. The parameters c_x and c_y represent the offset of the optical center from the center of the image. The skew, s, is non-zero if the image is not perfectly rectangular or square.

Extrinsic parameters have two components, a rotation R and a translation t in world coordinates. Intuitively, rotation is understood as Euler angles yaw (ψ), pitch (θ), and roll (ϕ) in degrees or radians. These three values do not fully describe orientation unless a rotation sequence is specified. A 3D rotation matrix is a 3×3 matrix that fully describes an orientation as shown in the following equations.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.6}$$

$$= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.7}$$

Translation units are dependent on the application and could be in relative coordinates like East and North or they could be in absolute coordinates like latitude and longitude. Translation is typically represented in a Cartesian 3D coordinate system. More information on translational coordinate systems is presented in Section 3.1. A translation vector for a cartesian coordinate system is represented as the following.

$$t = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.8}$$

2.1.4. Lens Distortion

Lenses are designed in a variety of shapes and sizes that can produce different distortion effects onto images. In addition, the manufacturing process can introduce small defects that can cause non-uniform distortion. To determine an accurate image projection, the image must be calibrated to the selected lens. This process is important especially for comparing images from different cameras and viewpoints. There are a variety of methods for lens calibration, but the most common method is to use a set of images of a checkerboard at different angles. There are algorithms available that can scan the checkerboard images for the points where the squares intersect. These points are compared to an ideal checkerboard. The discrepancies between points are used to select coefficients in a mathematical model of the lens.

There are two main types of distortion: radial and tangential. Radial distortion is due to the shape of the lens(es). There are two types of radial distortion: pincushion and barrel. Pincushion

shrinks the center of the image whereas barrel expands the center of the image. The diagrams in Figure 2.4 shows the effects of radial distortion on a checkerboard.

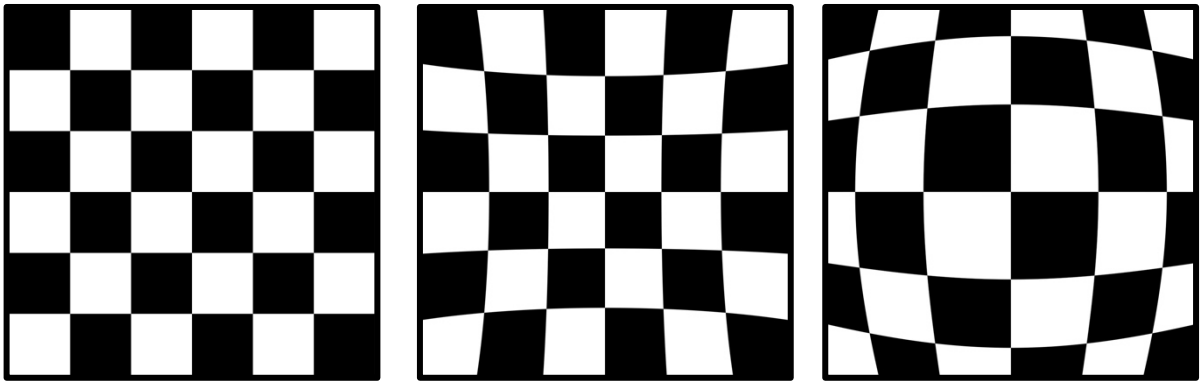

Figure 2.4: (Left) A standard checkerboard grid. (Center) A checkerboard with radial pincushion distortion as vertical lines bow outward like the edges of a striped throw pillow. (Right) A checkerboard with radial barrel distortion as vertical lines bow inward on the edges like the seams of a wooden barrel.

A tutorial by MathWorks® [7] includes a description of the method used in the Camera Calibration Toolbox for MATLAB® [8]. The Python and C++ library, OpenCV, uses a similar process [9]. Given the x and y locations of points in an undistorted and distorted image and $r = \sqrt{x^2 + y^2}$, the following equations can be solved for the radial distortion coefficients k_1, k_2, and k_3. These coefficients are stored for later use with the same equations to de-warp any image.

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.9}$$

$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{2.10}$$

Tangential distortion is due to a slight misalignment between the lens and the image sensor where they are not parallel. Figure 2.5 shows an exaggerated comparison of this misalignment. The tangential distortion is calculated with the same process as radial distortion to find the coefficients p_1 and p_2.

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2) \tag{2.11}$$

$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{2.12}$$

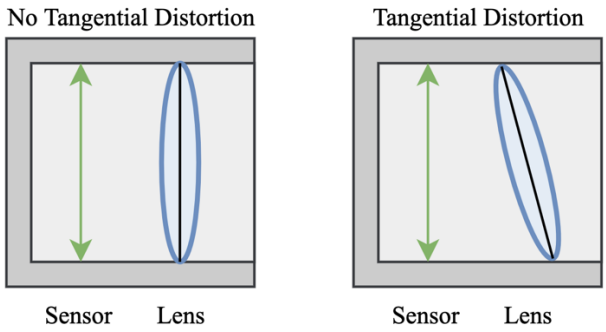

Figure 2.5: (Left) the lens and sensor are perfectly aligned, (Right) the lens is dramatically misaligned from the sensor.

The process for calibrating a camera involves taking ten to twenty pictures of a printed checkerboard taped to a planar surface. The Camera Calibration Toolbox for MATLAB® imports these images and a measurement of a single square width in mm. It then identifies the points on the checkerboard and finds a least square fit of an ideal checkerboard using the equations above for all the images simultaneously. Figure 2.6 shows a visualization of this process.

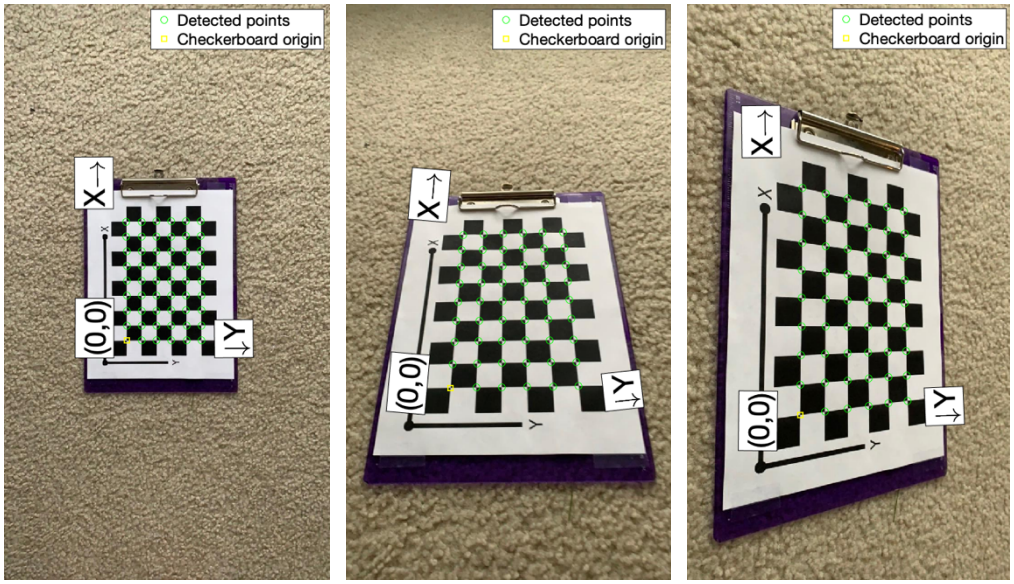

Figure 2.6: Images annotated with the Camera Calibration Toolbox for MATLAB®. The origin of each checkerboard is a yellow circle labeled with (0,0). The respective axes are labeled. Each checkerboard point detected is labeled with a green circle.

2.1.5. Histogram Equalization

An image processing technique to help in the comparison of two images from different cameras with very different lighting is histogram equalization. In the case of 8-bit digital grayscale images, there is a set number of pixels each with an intensity integer value 0 to 255. A histogram can be formed where the x-axis is intensity from 0 to 255 and the y-axis is the number of pixels for each corresponding intensity value in the image as depicted in the top row of Figure 2.7. Therefore, brighter images would have more pixels in the right-hand section of the histogram and darker images would have more pixels in the left-hand section of the histogram.

Histogram equalization is the process that adjusts the intensity of the pixels to spread out the histogram curve. This dramatically improves the contrast of an image without requiring specific tuning parameters. Let $h(x_i)$ be the number of pixels with intensity x_i. Therefore, the accumulated histogram should be equal to the total number of pixels or the product of the number of rows M and the number of columns N.

$$\sum_{i=0}^{255} h(x_i) = MN \tag{2.13}$$

The following relationship should be met for a given intensity value of k for an equalized histogram.

$$\sum_{i=0}^{k} \hat{h}(x_i) = \frac{MN(k+1)}{255} \tag{2.14}$$

By defining j as the intensity value in the original histogram, the transformation between the original and equalized histogram is as follows.

$$k = round\left(\frac{255}{MN}\sum_{i=0}^{j} h(x_i) - 1\right) \tag{2.15}$$

This provides the mapping between the intensity j to the intensity k.

As an example, an image from Google Satellite maps is processed with histogram equalization in Figure 2.7. In the histograms the large peak at intensity 100 is widened where the distinct intensity values are separated by small gaps. In the imagery, this shows a high amount of added contrast. While the resulting image does not look as good for human viewing, the details are much more distinct.

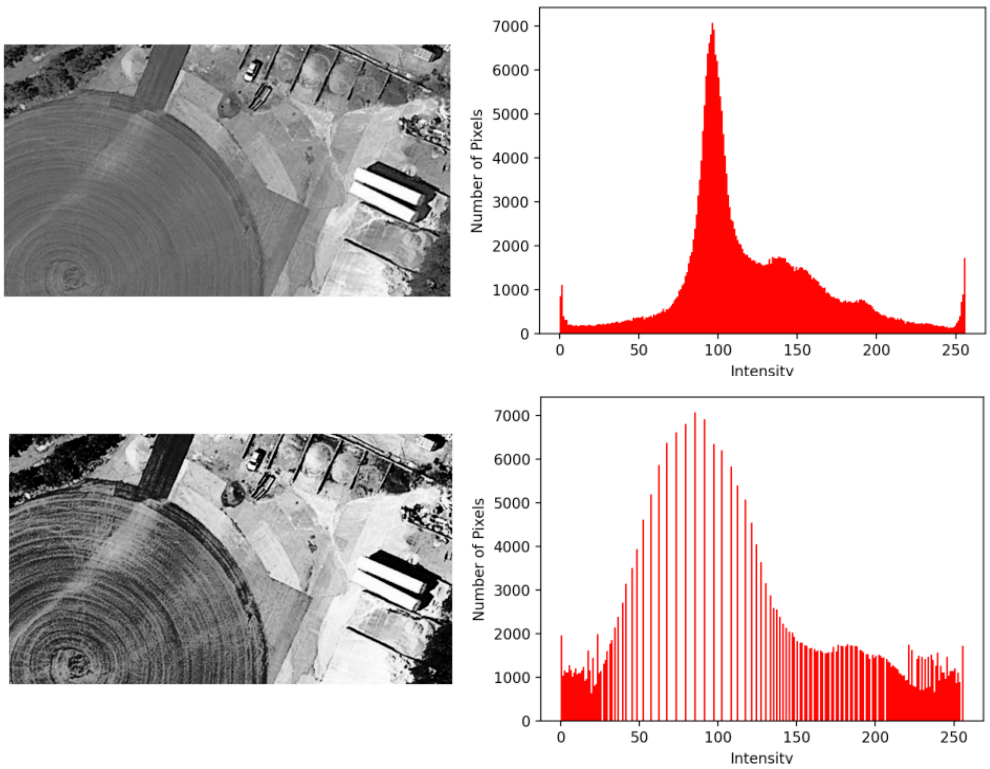

Figure 2.7: (Top row) Original reference image from Google Satellite maps and the corresponding histogram. (Bottom row) Histogram equalized reference image and the corresponding histogram

Performing histogram equalization on a sensed image from a UAV produces a similar result in Figure 2.8. The peak at intensity level 150 is spread out. There are now pixels occupying the entire range from 0 to 255 but with some small gaps. The spike at 255 is unaffected by the process. Comparing the imagery between the equalized sensed image and the equalized reference image

with a visual inspection demonstrates how histogram equalization helps make the images look more similar.

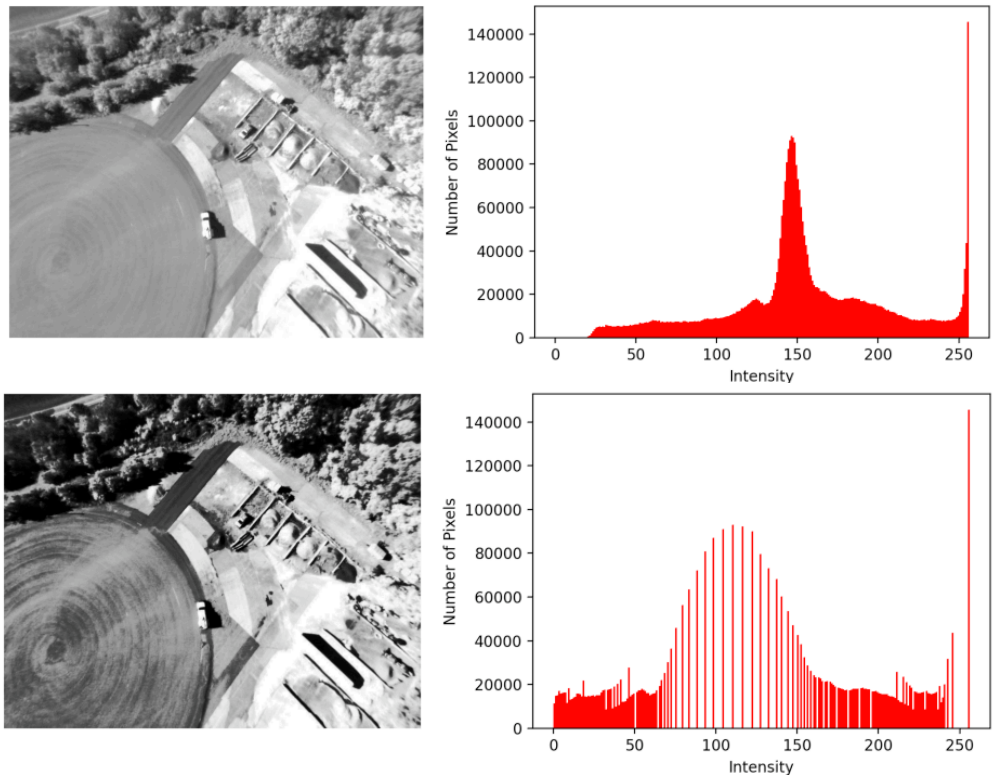

Figure 2.8: (Top row) Original sensed image from Google Satellite maps and its corresponding histogram. (Bottom row) Histogram equalized sensed image and its corresponding histogram

2.2. Image Registration

Image registration is the process of corresponding the coordinate system of one image into another. This could be a smaller image within a larger image or two images of similar size. There are a few different methods of image registration such as corner detection, template matching, and feature detection.

2.2.1. Correlation Template Matching

A way of comparing images is through correlation. Correlation is based on the 2D convolution of a template image $T(x, y)$ with a larger window image $W(x, y)$. 2D convolution

slides the template image throughout the window image and sums the element-based product of the two images for each position of the template.

$$R(x,y) = T(x,y) * W(x,y) = \sum_{x'y'}(T(x',y')W(x+x',y+y')) \quad (2.16)$$

The normalized cross-correlation can be found by dividing the convolution result by the square root of the squared convolution.

$$R(x,y) = \frac{\sum_{x'y'}(T(x',y')W(x+x',y+y'))}{\sqrt{\sum_{x'y'} T(x',y')^2 \sum_{x'y'} W(x+x',y+y')^2}} \quad (2.17)$$

The best correlation occurs at $\max(R(x,y))$. This is depicted in Figure 2.9 where the bottom left of the correlation result shows a vibrant white dot. The coordinates of this pixel represent the translation between the template image and the window. The normalized cross-correlation performs well in some situations but is not tolerant to the 6DOF case where the template and window images are not orthogonal or where the images are rotated. Any rotational difference between two image prevents the corresponding pixels aligning in a cross-correlation across the entire image. For this reason, cross-correlation alone is not useful for this application without additional steps to accommodate a variety of orientations.

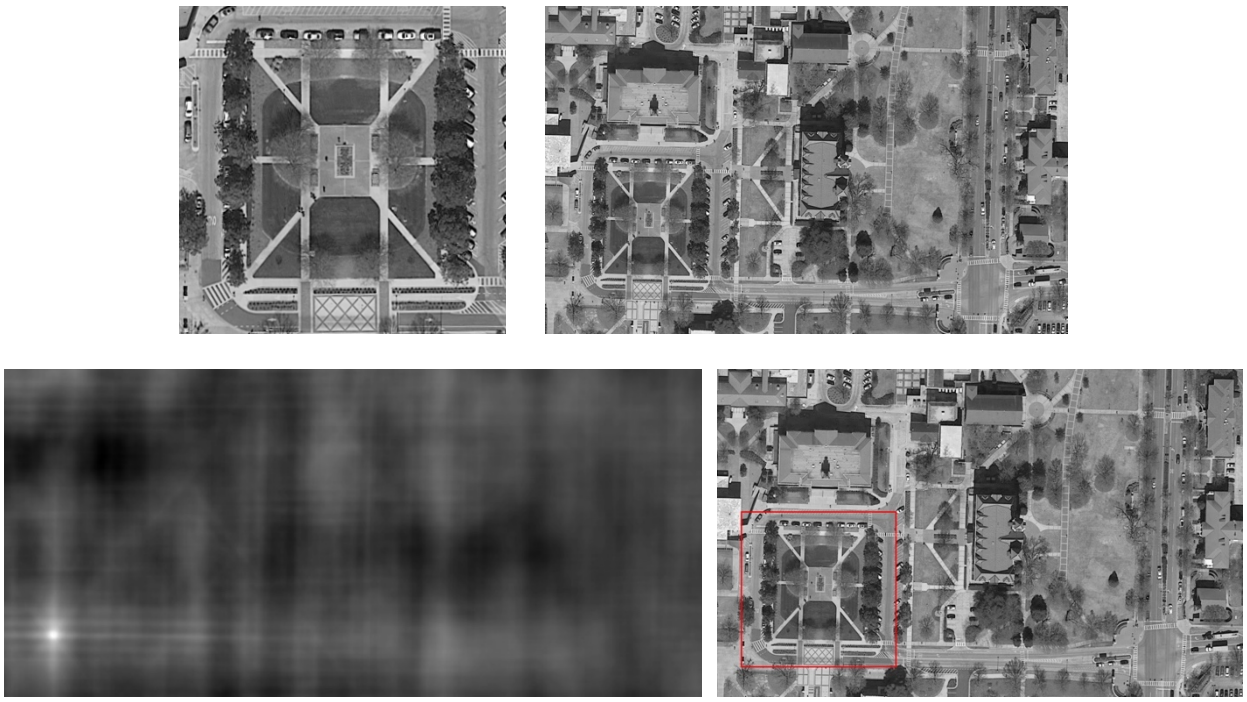

Figure 2.9: Template matching with Google satellite imagery: (Top-left) Template (Top-right) Window (Bottom-left) Correlation result (Bottom-right) Location of template based on correlation result in window.

2.2.2. Feature Matching and Pose Estimation

Instead of correlating each pixel of an image, individual points of interest of the image can be extracted, matched, and filtered to produce a more robust pose estimate result.

2.2.2.1. Corner Detection

Some of the first feature detectors in computer vision were edge and corner detection. Corners are easy to identify due to the distinct changes in image intensity. One of the earliest and most prominent corner detectors was proposed by Chris Harris and Mike Stephens in 1988 now known as the Harris corner [10]. It characterizes the intensity displacement of adjacent pixels to a given pixel based on the equation below where w is a rectangular or Gaussian window of the image I.

$$E(x,y) = \sum_{u,v} w(u,v)[I(x+u,y+v) - I(u,v)]^2 \quad (2.18)$$

With the Taylor expansion, this equation can be approximated as the following where I_x and I_y are the partial derivatives of the image in the respective directions.

$$E(x,y) \approx [x \quad y] M \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.19}$$

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{2.20}$$

To classify which regions are corners, an equation to score the result is formulated.

$$R = \det(M) - k\big(\text{trace}(M)\big)^2 \tag{2.21}$$

When R is large, it indicates a corner. When R is less than zero, the region is an edge. When R is a small number, the selected region is flat. The threshold of R can help tune how many corners are detected.

In 1994, Jianbo Shi and Carlo Tomasi proposed a modification to the Harris corner algorithm to form the Shi Tomasi corners [11]. Their main modification was redefining the method to find R. By taking the eigenvalues of M, the Harris corner R can be represented as the following.

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2) \tag{2.22}$$

Shi and Tomasi instead defined R as a minimum of the two eigenvalues of M to improve the performance of corner detection.

$$R = \min(\lambda_1, \lambda_2) \tag{2.23}$$

Figure 2.10 shows an example of both corner detection algorithms in action. There are some obvious corners that are detected, but many that did not pass through the threshold. One of the disadvantages of corner detection is that it only uses a single strength factor. A single metric does not allow for accurate matches between two related points in multiple distinct images. Multiple metrics are needed to have acceptable matching.

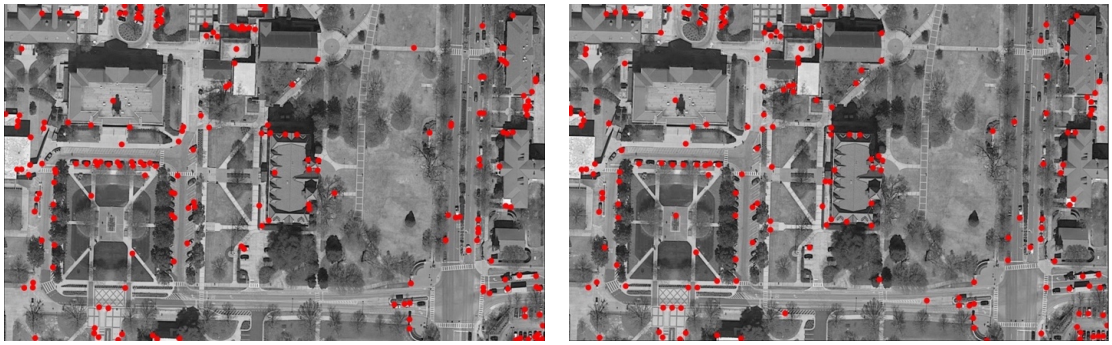

Figure 2.10: Corner detection on Google satellite imagery over Auburn University (Left) Harris corners (Right) Shi Tomasi corners

2.2.2.2. SIFT

Feature detectors can detect many features in each image assuming it is sufficiently detailed. The challenge is detecting a decent number of accurate matches. The simplest matching algorithm is the brute-force matcher. It selects a feature in one image and finds a distance metric to every feature in the second image. This distance metric varies for each type of feature detector but is more closely related to the strength and orientation of the feature rather than the pixel distance in the overlapped images. The result with the nearest neighbor is the selected match. Figure 2.11 shows an example of matched SIFT features where a small section of the image is cropped and rotated. The detected features are represented with red dots and the matches are represented as yellow lines. From a visual inspection many of the matches are correct, but there are two outliers that match the clipped image to areas way outside of the corresponding area.

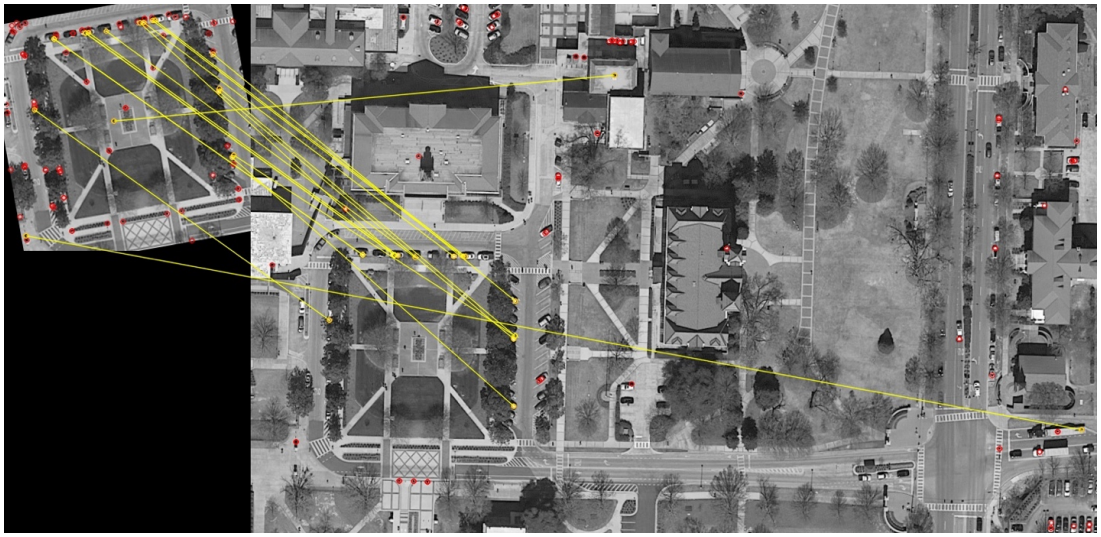

Figure 2.11: SIFT matches of cropped and rotated Google satellite imagery.

2.2.2.3. Match Filtering and Pose Estimation

When performing feature matching between real images and satellite images, there are a variety of concerns to take into consideration. Figure 2.12 highlights these differences due to changes in time, perspective, and environment. While human eyes are very capable of understanding these images were captured over the same area, training computer algorithms to perform this same comparison takes some refinement. Using SIFT with the full Google satellite reference image and a UAV sensed image produces the image matches in Figure 2.13, but a visual inspection identifies numerous false matches.

Figure 2.12: Difference between Google satellite imagery (left) and UAV imagery (right). (1) Shadows moving due to change in time and season. (2) Perspective changes due to different altitudes of camera. (3) Environmental changes such as moving or absent vehicles.

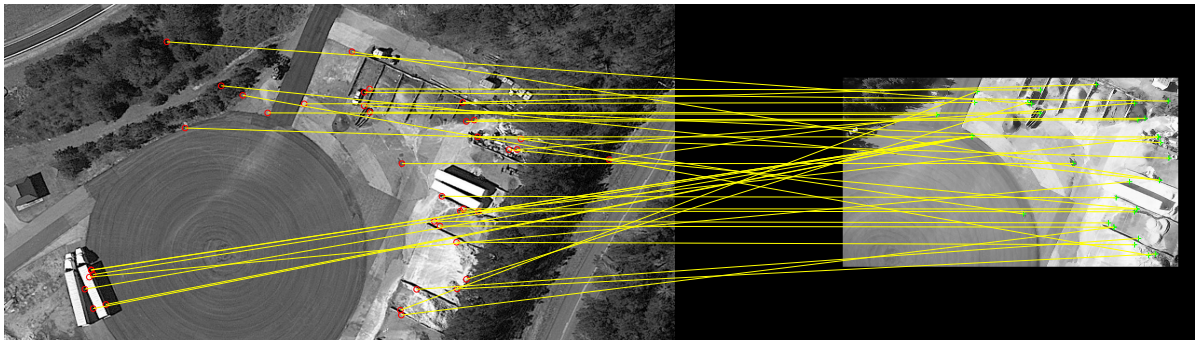

Figure 2.13: Example of feature matching with real imagery

2.2.2.4. Random Sample Consensus (RANSAC)

Feature matching often produces numerous bad matches. Typically matches should follow a uniform geometric pattern. If the images are simply translated from one another, all the matches should be parallel. If the images are rotated, the matches should form a radial pattern. A mathematical model can best fit these matches to remove outliers. One common algorithm to form this optimization is called Random Sample Consensus (RANSAC) [12]. It accomplishes the best fit by starting with a randomly selected subset of the data. It then fits the best model to that subset. It repeats this process for a set number of trials. The trial with the highest number of inliers is selected as the best fit. The plots in Figure 2.14 show an example of this process on scatter plot of points where RANSAC can produce a line that better fits the trend of data than least squares. RANSAC is typically paired with a method of pose estimation.

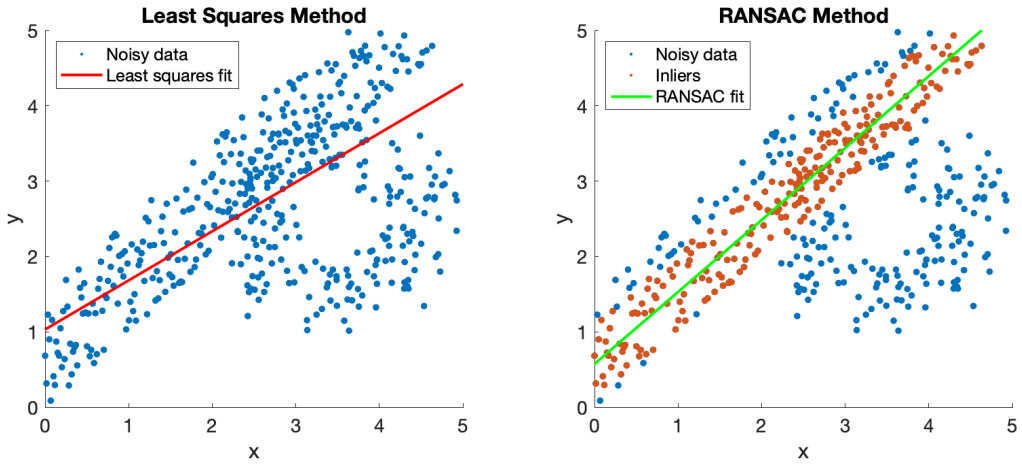

Figure 2.14: A comparison of using the least squares method to fit a line to a set of noisy data versus using the RANSAC method. It also can select inliers indicated in orange and reject outliers indicated in blue.

2.2.2.5. Projective Transformation and Homography

Once feature matches are determined and filtered, additional processing is needed to extract the camera pose between the two images. The problem is related to image registration where there exists a mathematical model to transform one image to match another geometrically. An affine transformation, represented by a 3×3 matrix with the last row identical to the last row of an identity matrix, is capable of translating, reflecting, scaling, rotating, and shearing an image for a total of six degrees of freedom. A projective transform is similar to an affine transformation but uses all nine values of the transformation matrix for a total of eight degrees of freedom. This matrix is referred to as a homography matrix. The following equation and Figure 2.15 represent this transformation with (x, y) as image 1's pixel coordinates and (x', y') as the new coordinates of the transformed pixel.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.24}$$

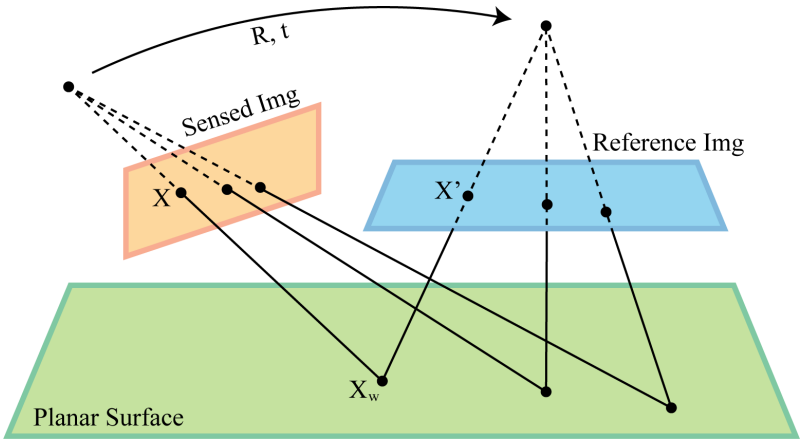

Figure 2.15: Projective transformation of reference and aerial sensed image of planar surface described by a rotation matrix R and translation matrix t.

The homography matrix is determined using the following equation.

$$H_1^2 = R_1^2 + \frac{t_1^2 n^T}{d} \tag{2.25}$$

The rotation matrix from image 1 to 2 is represented as R_1^2. The tranlation vector is t_1^2. The normal vector is labelled as n. The normal distance, or the distance of the camera to the ground, is d. This equation allows the homography matrix to be determined for any extrinsic parameters of the camera when it is pointed towards the ground.

2.2.2.6. Perspective-n-Point (PnP)

One of the limits of homography, is the assumption the world is a planar surface. While this works for some applications, the terrain in the real world is not planar even in relatively flat regions. Perspective-n-Point (PnP) estimates an object's pose from a 3D-2D point correspondence using world coordinates. A minimum of three points are required to produce a closed solution. This configuration is called P3P. The following equations from [12] define the relationship between the world points and the image points as depicted in Figure 2.16.

$$(\mathrm{R_{ab}})^2 = \mathrm{a}^2 + \mathrm{b}^2 - 2\mathrm{ab}[\cos(\theta_{\mathrm{ab}})] \tag{2.26}$$

$$(\mathrm{R_{ac}})^2 = \mathrm{a}^2 + \mathrm{c}^2 - 2\mathrm{ac}[\cos(\theta_{\mathrm{ac}})] \tag{2.27}$$

$$(R_{bc})^2 = b^2 + c^2 - 2bc[\cos(\theta_{bc})] \tag{2.28}$$

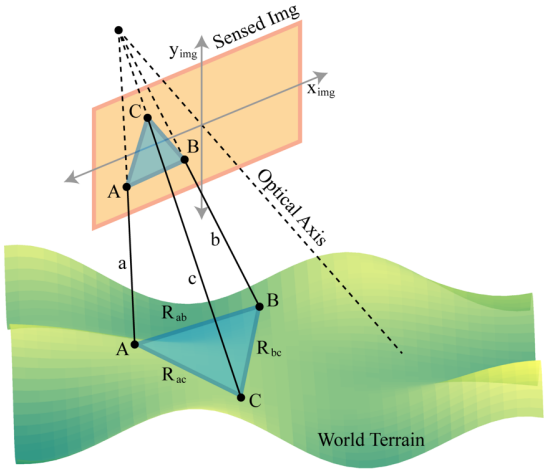

Figure 2.16: Diagram depicting geometry of P3P algorithm.

PNP can be described by a mathematical model similar to the projective transformation with accommodation for world units.

$$\begin{bmatrix} x_{img} \\ y_{img} \\ 1 \end{bmatrix} = K[I_{3\times3} \quad 0_{3\times1}] \begin{bmatrix} R & t_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \\ 1 \end{bmatrix} \tag{2.29}$$

In practice, P3P alone is not reliable for aerial map-matching because it is rare to have a perfect set of matches. Instead RANSAC paired with PnP produces many solutions for the pose, selecting the best feature matches as inliers. The authors of the RANSAC algorithm proposed aerial cartography with RANSAC PnP [12]. Each solution is checked by reprojecting the image points into the world frame as visualized in Figure 2.17. The reprojection error is the distance in pixels between the reprojected point and the world point. The RANSAC PnP algorithm attempts to minimize this reprojection error over numerous iterations to determine the best fit rotation and translation matrix. The variant of PnP used in this thesis is based on the improvements made in [13] by Terzakis and Lourakis in 2020 that efficiently improves the chance PnP will find a global minimum.

Figure 2.17: Reprojected sensed image points on Google satellite imagery reference map. The corresponding matches are indicated by yellow lines.

2.2.3. Corner Selected Windows with Normalized Cross-Correlation Matching

One method to improve the successful match rate between two images is by using a search window-based fine-tracking method. The method assumes a coarse pose estimate with some uncertainty is known from either a previously determined state or a coarse search method. The Vision-Aided Inertial Navigation (VISINAV) [14] system proposed by Mourikis et al. is based on the Mars Exploration Rover (MER) Descent Image Motion Estimation System (DIMES) [1] used for the descent stage of the Spirit and Opportunity rovers built by NASA's Jet Propulsion Laboratory (JPL). The "marked landmark" algorithm described performs a fine adjustment of the projection of a sensed image on a satellite reference image and digital elevation map (DEM) to determine the latitude, longitude, and altitude of marked landmarks. The following steps will outline the implementation of Algorithm 2 specified in [14] on a UAV imagery case.

Step 1: Select multiple templates in descent image

This step is performed using the Shi Tomasi corner detection algorithm to identify corners in the sensed image. Additional image processing was done to narrow down the corners such as, dilating and eroding corners. It then sorts by corner strength and removes close duplicates. Only

the top 30 corners are considered as shown in Figure 2.18. One of the corners, highlighted in red, is studied for this example due to its distinct visual representation in both the sensed and reference image.

Figure 2.18: (Left) Corners detected in simulated sensed image from Google satellite maps with Shi Tomasi corner detection. Selected point for analysis is indicated by a red dot. (Right) Corner in a zoomed-in view of sensed image

Step 2: Compute homography that rectifies descent image to map using lander position and attitude estimation.

For this example, the homography correlation between the sensed and reference image were determined with manual point correspondence. In a full navigational system, this homography is determined with the coarse pose from another computer vision algorithm or with sensor fusion using the equations in Section 2.2.2.5. Figure 2.19 shows this correspondence between the sensed and reference image. Usually, the images would show signs of pixel shifting as the coarse pose is not guaranteed to be the same as truth.

Figure 2.19: (Lighter foreground) Google satellite maps simulated sensed image overlapped with (Darker background) Bing aerial maps reference image with homography

Step 3: Warp each template using homography

A template is a cropped section of the sensed image with a height and width as a tunable number of pixels around each corner. The template is warped based on the coarse homography matrix as shown in Figure 2.20. This warped template is later used in a correlation search in a window. Because the match for the corner should be close to the reprojection of the corner, only the surrounding area should be searched. The homography matrix is used to reproject an expanded version of the template outline onto the reference image. A window is defined by a given area around the reprojected corner is demonstrated in Figure 2.21. The window size is also set to a constant parameter as a number of pixels in width and height. See Section 2.2.4 for a method to dynamically set the window size.

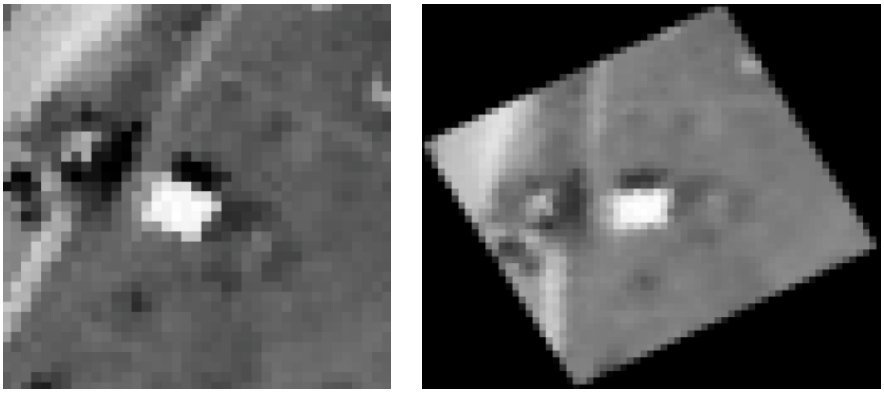

Figure 2.20: (Left) Selected corner template from sensed image. (Right) Warped template to reference frame with homography

Figure 2.21: Window around selected corner

Step 4: Spatially correlate each warped template with map.

The warped template is matched with respect to the window in the reference image using the normalized cross-correlation algorithm described in Section 2.2.1. This returns a matrix slightly smaller than the reference image representing the correlation in the frequency domain. The correlation result is displayed in Figure 2.22. Even though the window has some repeated features, the template matching does a good job of finding a peak value.

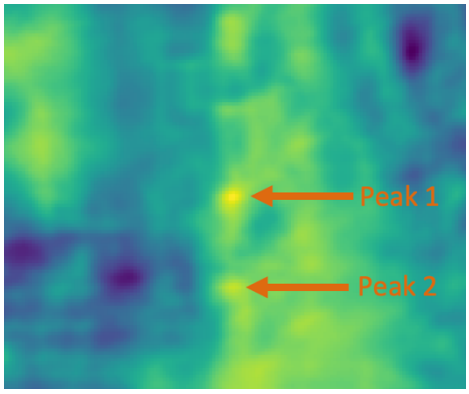

Figure 2.22: Correlation of template in search window. Higher intensities in yellow represent a stronger correlation and lower intensities in blue represent a weaker correlation. The two peaks are indicated with arrows.

Step 5: Mark match as valid if correlation peak passes checks on height, width, and ratio to second-highest peak

According to [14] the following expression should be used to fit a biquadratic surface to the peak correlation 3x3 neighborhood:

$$au^2 + bv^2 + cuv + du + ev + f \quad (2.30)$$

The next step is to determine whether the match is valid. This is done with a few different tests. The simplest, is determining whether the first highest correlation match max height is above a threshold. This removes any low correlation results. The next test is to determine whether the peak width exceeds a threshold. This helps remove matches with a low frequency correlation in a certain direction such as along an edge where the match would be unreliable. Another test finds the ratio between the highest correlation peak to the second highest peak to help remove matches with repetitive terrain elements. A subpixel correction is used to improve the match accuracy and removes any matches that have a correction greater than 1.5 pixels determined with the following two equations based on the bicubic surface expression.

$$u_p = \frac{-2bd+ce}{4ab-c^2} \quad (2.31)$$

$$v_p = \frac{-2ae+cd}{4ab-c^2} \quad (2.32)$$

Step 6: If match is valid, generate ML.

For each valid match, the feature is stored into a database with the corresponding latitude, longitude, and elevation. Figure 2.23 shows the result of the selected corner reprojected on the reference image. The identified match is close to where the corner was in the sensed image. Figure 2.24 displays the template matches for all 30 corners demonstrating all good matches.

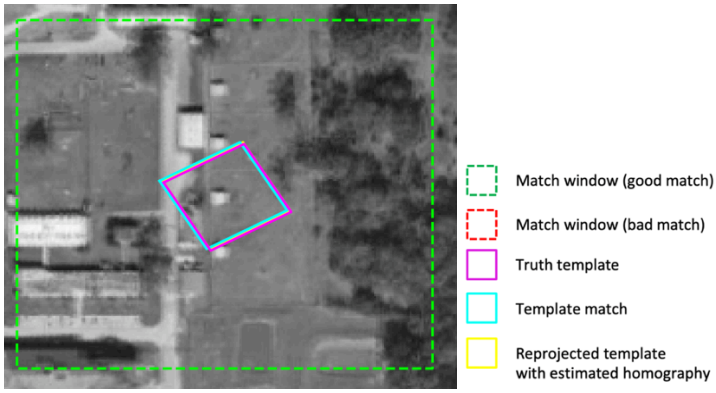

Figure 2.23: Projection of selected corner on the reference image and its template match.

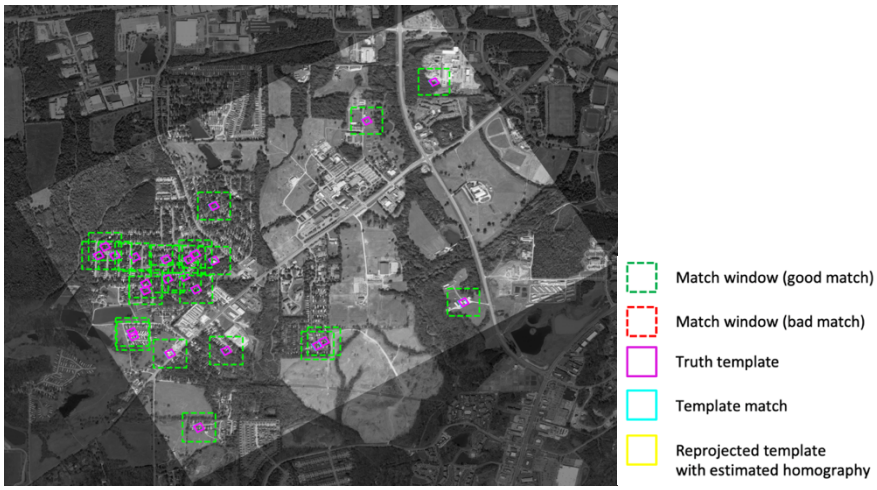

Figure 2.24: Projection of all valid corners and matches on reference image.

The window is represented with a dashed box and colored as green if the match is good and red if it is bad. A match is good if the template match (cyan box) is within a set number of pixels of the truth template (magenta box). The reprojected template (yellow box) represents where the

coarse pose indicates the match should occur. If the coarse pose and the truth pose are the same, then the reprojected and truth template boxes should exactly overlap.

2.2.4. Dynamic Window Matching

With the corner normalized cross-correlation algorithm, there are cases where the truth template match is outside of the search window and it is impossible for the algorithm to find the correct match. These cases typically occur where there is a significant pose estimate error in rotation (a few degrees) and with matches near the edges of the sensed image. Figure 2.25 demonstrates an example of this issue.

Figure 2.25: Matches with a sensed Google Satellite to reference Google Satellite image with rotation error of 3 degrees in pitch, roll, and yaw. Matches where truth template is outside of window are indicated with orange arrows.

In addition, the larger the window size, the more processing time that is required to find a match and the lower the match rate. A simple test shown in Figure 2.26, demonstrates this process on the normalized cross-correlation described in Section 2.2.1. A template from Google Satellite maps with a size of 100×100 pixels was matched in a window from Bing Aerial maps of increasingly larger width and heights. A 10% salt and pepper noise filter was added to the template image to increase the probability of a bad match simulating the difference in imagery from two

cameras. The match error remains zero until the window size reaches the critical threshold at 2500 pixel width, where the correlation algorithm is unable to find the correct match. The processing time increases at approximately $O(N^2)$ showing that even small changes in window size can cause large increases in processing time.

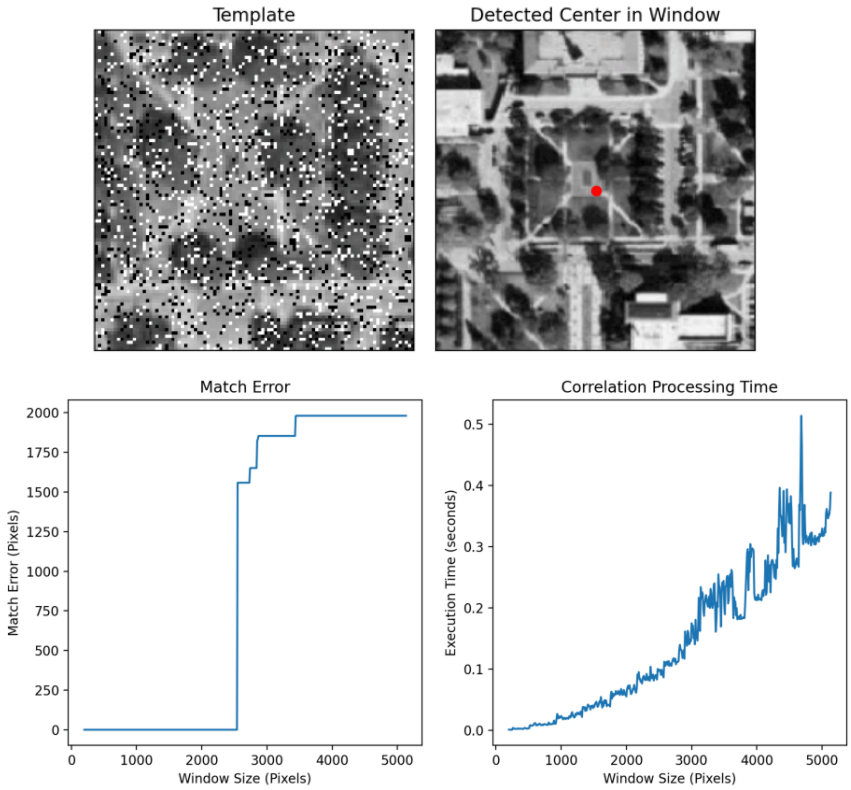

Figure 2.26: (Top-row) Template with added noise and its corresponding match in a window with a size of 200 pixels. (Bottom-row) Match error and processing time with normalized cross-correlation as a function of window size.

The requirements of a successful match rate and low processing time indicate the need for dynamic window sizing. A simple solution to implement dynamic window sizing is to "nudge" the coarse pose template to each extreme in the modelled error. This is done in a brute force method by generating homography matrices representing the extremes in translation and rotation. Three scalar values are used to set the dynamic window size for translation, altitude, and Euler rotation errors respectively: t_{error}, z_{error}, and r_{error}. These values are set as tunable parameters.

In this thesis, the values are set as constant for each run. They could be modelled off of the known error ranges of sensor such as an IMU or a coarse pose algorithm. Alternatively, the parameters could be adjusted in real time with the covariance values from a Kalman Filter. The values from the following matrices are used independently in the homography to extrinsic equation.

$$\mathrm{t}_1^2 = \begin{bmatrix} t_{error} & 0 & 0 \\ 0 & t_{error} & 0 \\ t_{error} & t_{error} & 0 \\ -t_{error} & 0 & 0 \\ 0 & -t_{error} & 0 \\ -t_{error} & -t_{error} & 0 \\ 0 & 0 & z_{error} \\ 0 & 0 & -z_{error} \end{bmatrix} \tag{2.33}$$

$$\mathrm{r}_1^2 = \begin{bmatrix} r_{error} & 0 & 0 \\ 0 & r_{error} & 0 \\ 0 & 0 & r_{error} \\ r_{error} & r_{error} & r_{error} \\ -r_{error} & 0 & 0 \\ 0 & -r_{error} & 0 \\ 0 & 0 & -r_{error} \\ -r_{error} & -r_{error} & -r_{error} \end{bmatrix} \tag{2.34}$$

Each row of the matrix r_1^2 is converted from Euler angles to a rotation matrix R_1^2 for the calculation of homography. The homography calculation uses equations described in Section 2.2.2.5. The result from these calculations is a collection of points clustered around the course pose template. A simple rectangular box is drawn around the points with some margin signifying the cropped window used in the correlation step as shown in the depiction in Figure 2.27.

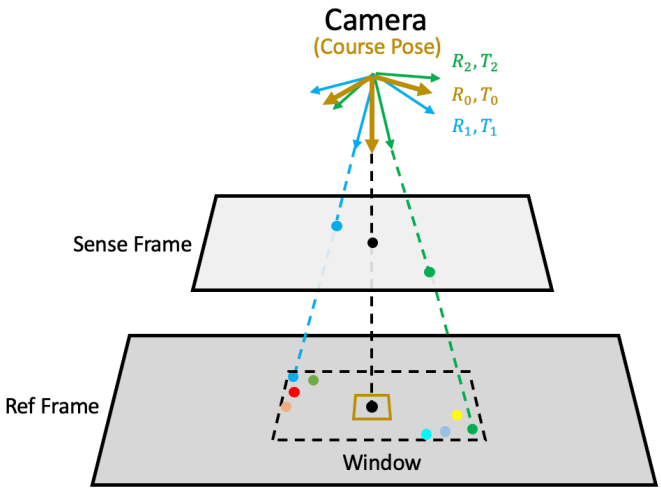

Figure 2.27: Dynamic window size concept. Course pose is represented in dark yellow with R_0 and T_0. The window is sized to fit all of the reprojected points.

The effect on window matching in an image with significant rotation error is shown in Figure 2.28. The red dots are the nudged homography templates. None of the truth templates appear outside of the selected window.

Figure 2.28: Google sensed to Google reference image with 3 degrees error in pitch, roll, and yaw with dynamic window sizing implemented.

2.3. Conclusion

This concludes the discussion on image processing and computer vision techniques as a foundation of terrain relative navigation. The basic foundational algorithms like correlation, corner detection, and feature matching are useful for comparing two images. A more advanced algorithm combining corner detection and correlation was outlined with the addition of dynamic window sizing that makes it a more useful for integration with other navigation algorithms.

Chapter 3: Geographic State Estimation

In order to use information from imagery to find the camera pose, it must be related to world coordinate systems and fused with data from other sensors.

3.1. Geographic Information System (GIS)

An important aspect of map-matching is defining a frame of reference for the map. A map must be geo-referenced where given a specific pixel, the latitude, longitude, and elevation can be determined. Due to working with 2D images and maps in a 3D world, it is important to establish geographic conventions.

3.1.1. Geographic Systems

Any location in the world can be described by a latitude (ϕ) and longitude (λ). Latitude and longitude are expressed in degrees with higher precision values represented as a decimal degree or in minutes (‘) and seconds (“). Decimal degrees are simpler to work with computationally and will be the convention for this thesis. Latitude is referenced as the angle between the selected point and the equator with values ranging from -90° to 90°. Longitude is referenced as the angle between the selected point and the Prime Meridian through the Royal Observatory in Greenwich London, UK with values range from -180° to 180°. The geographic datum, World Geodetic System of 1984 (WGS 84), approximates the Earth as an ellipsoid where the center of mass is defined including the oceans and atmosphere [15]. The Department of Defense (DoD) GPS satellites use this datum. WGS 84 is also known as the EPSG:4326 projection. A geodetic system differs from a geocentric system. The distinctive element is the latitude of a geodetic system is not referenced at the center of the ellipsoid except at the equator and the poles. Figure 3.1 shows an example of where the line from point P to the normal line does not intersect the center in a Geodetic coordinate system.

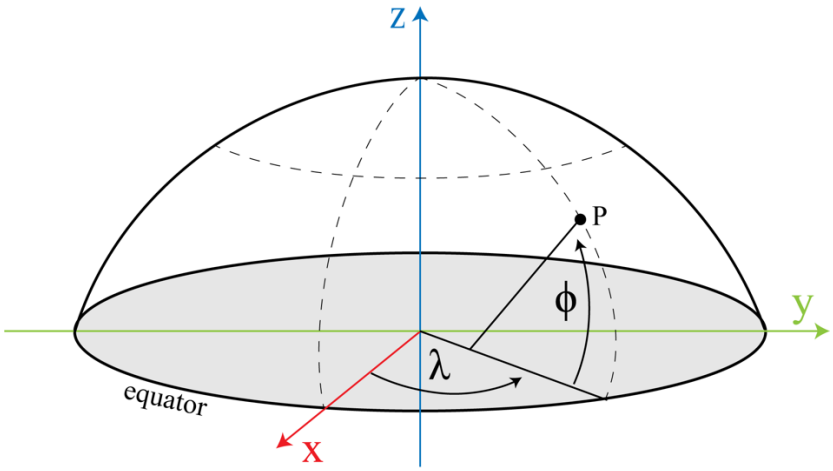

Figure 3.1: An ellipsoid approximation of the shape of the Earth with three-dimensional Cartesian coordinate system.

The Earth is not a perfect ellipsoid. The WGS 84 standard defines a geoid based on a gravitational model for the local Mean Sea Level (MSL) [15]. Figure 3.2 demonstrates an example of the difference between the ellipsoid, geoid, and topography. The distance between the ellipsoid and the geoid (geoid undulation) is labeled. The difference between the geoid and the topography (orthometric height) is labeled h. The geodetic height (H) represents the distance between the topography and ellipsoid. The following equation describes this relationship.

$$h = H + N \tag{3.1}$$

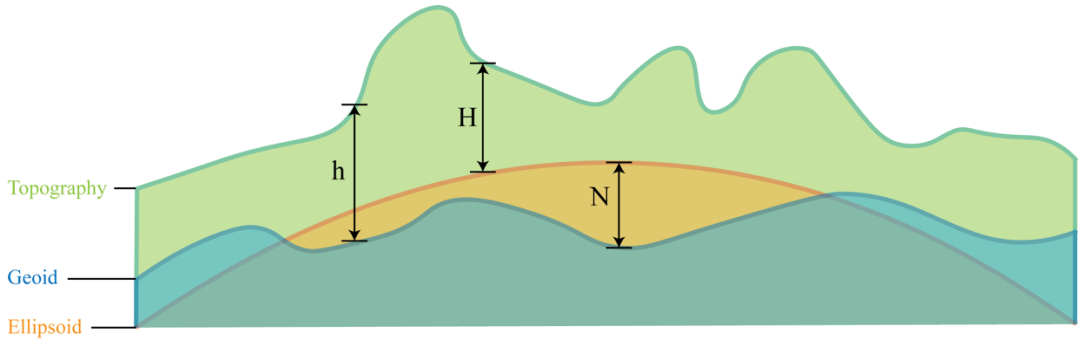

Figure 3.2: Diagram comparing the relationship between a WGS-84 ellipsoid, geoid, and topological elevation.

3.1.2. Local Coordinate Frames

Latitude and longitude resemble spherical coordinates. The issue with elliptical coordinates is the distance of one degree latitude lines changes based on the location. One degree of longitude around the equator is a much longer distance than one degree of longitude near the north pole. Because of this, it is not easy to have an intuitive understanding of distances in geodetic coordinates. Cartesian coordinates, normally expressed in meters, provide an easier reference frame for relative distances. There are two commonly used Cartesian local coordinate frames: East, North Up (ENU) and North, East, Down (NED). ENU is practical for defining coordinates on map planes in reference to the ground. NED is useful for defining relative distances to aircraft or other vehicles. These coordinate systems assume the world is flat because an infinitesimally small area of an ellipsoid is approximately flat.

A straightforward way to convert between geodetic coordinates and local coordinates is to approximate the Earth as a sphere and use the Haversine formula. This formula allows for the computation of distance across any two points on a sphere. It is more accurate for shorter distances across the Earth rather than longer distances due to the spherical assumption. The haversine is defined as the following.

$$hav(A) = \sin^2\left(\frac{A}{2}\right) = \frac{1-\cos\ A}{2} \tag{3.2}$$

By defining point 1 as (λ_1, ϕ_1), point 2 as (λ_2, ϕ_2), and the radius of earth as $r = 6{,}378km$, the distance in kilometers is found with the following equation:

$$\mathrm{d} = 2\mathrm{r}\sin^{-1}\left(\sqrt{\mathrm{hav}\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\,\mathrm{hav}\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{3.3}$$

3.1.3. Satellite and Ortho-Imagery

To find the pose of a camera, it must be compared to georeferenced imagery referenced to absolute world coordinates. Ortho-imagery is a dataset of georeferenced images that is processed to be planar and orthogonal to the ground captured by either an aircraft or satellite. The files are typically stored in the geotiff (.tif) or JPEG 2000 (.jp2) filetypes. These filetypes support georeferencing metadata such as projection type and map extent. Libraries like GDAL can interpret this data and can extrapolate the coordinates for any given pixel. Software programs like QGIS and ArcGIS can easily view, crop, merge, and reproject these files. Each source of imagery has a different resolution. If using lower altitude sensed imagery, a higher resolution reference map is required. Frequently updated imagery like the European Space Agency's (ESA) Sentinel 2 satellite imagery has a resolution of about thirty meters per pixel. Less frequently updated imagery like United States Geological Survey ortho-imagery has a resolution of about half a meter per pixel. Table 3.1 and Figure 3.3 compares Sentinel 2, USGS HR ortho-imagery, and Google Maps imagery. The ESA and USGS data are available from USGS's Earth Explorer web service: https://earthexplorer.usgs.gov/. The Google and Bing map imagery is available through the program QGIS as XYZ Tiles.

Table 3.1: Common ortho-imagery sources

Organization and Name	Spatial Resolution (pixel/m)	File Type	EPSG Projection
ESA Sentinel 2	~ 30	.jp2	32616
USGS HR ortho-imagery	~ 0.5	.tif	26916
Google Static Maps via QGIS	< 0.5	.tif	3857
Bing Aerial Maps via QGIS	< 0.5	.tif	3857

Figure 3.3: (Left) ESA Sentinel 2 satellite imagery, (Middle) USGS high resolution ortho-imagery, (Right) Google Static Map API satellite imagery.

3.1.4. Digital Elevation Maps (DEM)

Ortho-imagery provides the terrain imagery of an area but includes no information about the elevation. A digital elevation map (DEM) is a georeferenced image where each pixel represents the elevation of the corresponding location. DEMs are captured from satellites or aircraft using RADAR or LIDAR. One widely used DEM is the Shuttle Radar Topography Mission (SRTM) that was captured with radar on the Endeavour space shuttle in 2000. The spatial resolution of a DEM is typically measured in arc-seconds. For instance, the SRTM is rated as one arc-second in the US, approximately thirty meters/pixel around the equator. One latitude or longitude degree is composed of sixty arcminutes. Each arcminute is composed of sixty arc-seconds. Most globally available DEMs are satellite based due to the impracticality of capturing global aircraft LIDAR data. Figure 3.4 shows a DEM map and a 3D colorized depiction of it. 3D visualization services like Google Earth use DEMs with an ortho-image texture to produce a realistic aerial view of the world.

Table 3.2: Common elevation map comparison

Organization and Name	Horizontal Datum	Spatial Resolution (arc-second)	File Type
USGS Shuttle Radar Topography Mission [16]	WGS 84	1 (US) 3 (Globally)	.tif
USGS 3DEP	NAD 83	1/3	.tif
JAXA Advanced Land Observation Satellite	WGS 84	1	.tif

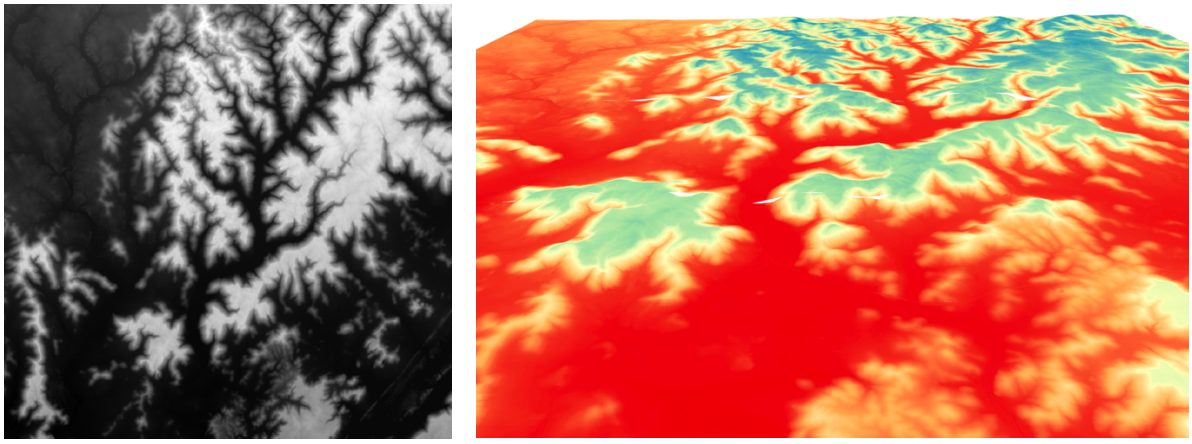

Figure 3.4: Digital elevation map of north-east Alabama from USGS's 3DEP dataset. (Left) Raw 2D image where lighter values indicate higher elevation. (Right) 3D view of elevation map.

3.2. Sensors

For GPS denied applications, additional sensors are helpful in constraining the localization problem. For instance, a barometer is used to estimate altitude and an IMU is used to estimate orientation.

3.2.1. Barometer

A passive way to estimate altitude is using a barometer to measure air pressure. This method is widely used in aviation. Air pressure changes due to several factors like altitude, temperature, and weather conditions. The fluctuation of weather conditions can be accounted for by determining the pressure at mean sea level (MSL) at the time data is collected. This can be

calculated by using a GPS altitude measurement at the start of a data run along with temperature and pressure. The formula below represents the height above MSL based on the temperature at MSL (T_b), the temperature lapse rate (L_b), the universal gas constant (R), gravitational acceleration (g_0), and the molar mass of Earth's air (M) for altitudes less than 11km [17].

$$h = \frac{T_b}{L_b} \cdot \left[\left(\frac{P}{P_b} \right)^{-\frac{R \cdot L_b}{g_0 \cdot M}} - 1 \right] \tag{3.4}$$

For finding the temperature and pressure at MSL with the temperature and pressure at a known height, the following equations can be used.

$$T_b = T - L_b \cdot h \tag{3.5}$$

$$P_b = \frac{P}{\left[1 + \frac{L_b}{T_b} h \right]^{\frac{-g_0 \cdot M}{R \cdot L_b}}} \tag{3.6}$$

For measurements in the stratosphere, a different equation should be used where h_b, T_b, and P_b are relative to the bottom of the stratosphere.

$$h = h_b + \frac{R \cdot T_b \cdot \ln\left(\frac{P}{P_b} \right)}{-g_0 \cdot M} \tag{3.7}$$

To demonstrate this process, a low altitude UAV test was conducted with a barometric altimeter and a GPS. See Section 5.1 for more information on how the UAV data is collected. The original calculated altitude with a standard pressure at MSL was about 50m offset from the GPS as shown in Figure 3.5. By using the process listed above, the calibrated altimeter altitude much more closely resembles the GPS truth and remains less than a 10m offset during most of the flight but starts to drift over time.

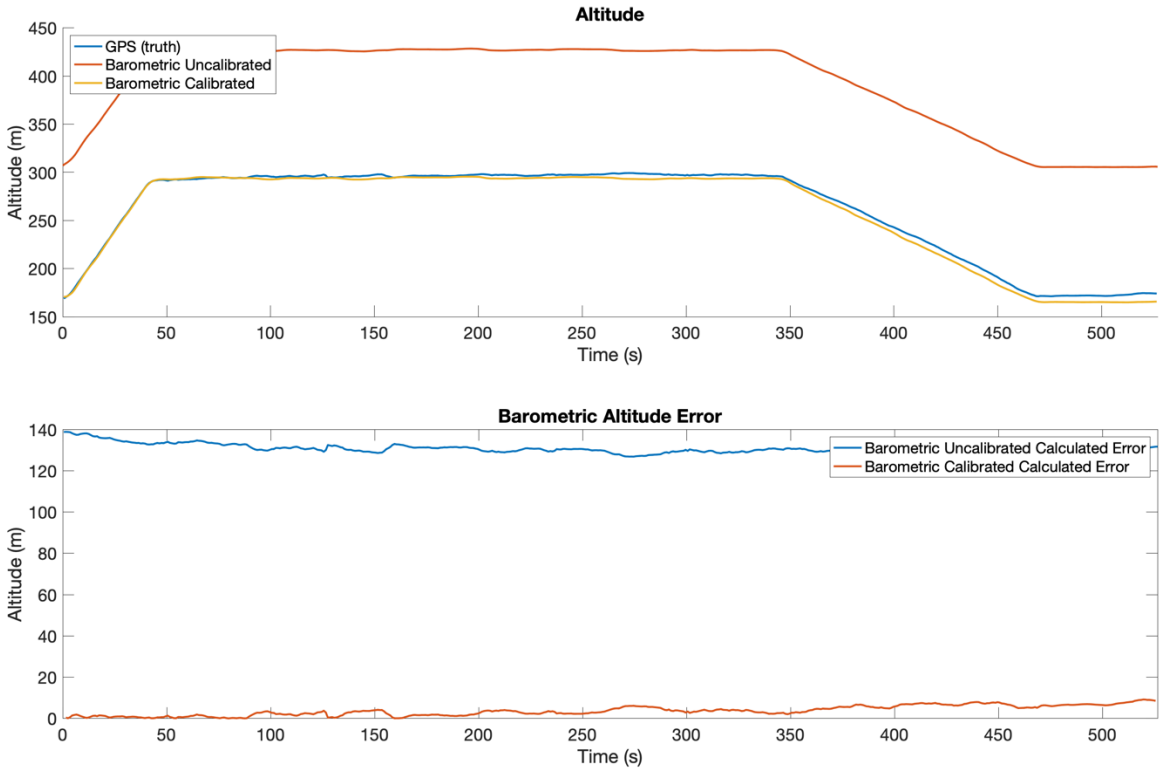

Figure 3.5 Low altitude UAV test of altitude uncalibrated and calibrated calculations compared to GPS truth. Calibration occurred just before takeoff.

For high altitude balloons where the altimeter is in the stratosphere, the alternate equation should be used. The results in Figure 3.6 show the calibration reduces the error when GPS is present. The overall error is higher than the UAV test because of the longer flight duration and distance. This drifting in error can be reduced by using the last known GPS altitude instead of the initial value. A few meters of inaccuracy should be acceptable for a coarse pose for a low altitude UAV under 100m but more than 10m would corrupt the state estimate. Likewise for a high altitude balloon at a few thousand meters of altitude, an error of up to 100m should be acceptable, but significantly more error than that could cause state estimate inaccuracies.

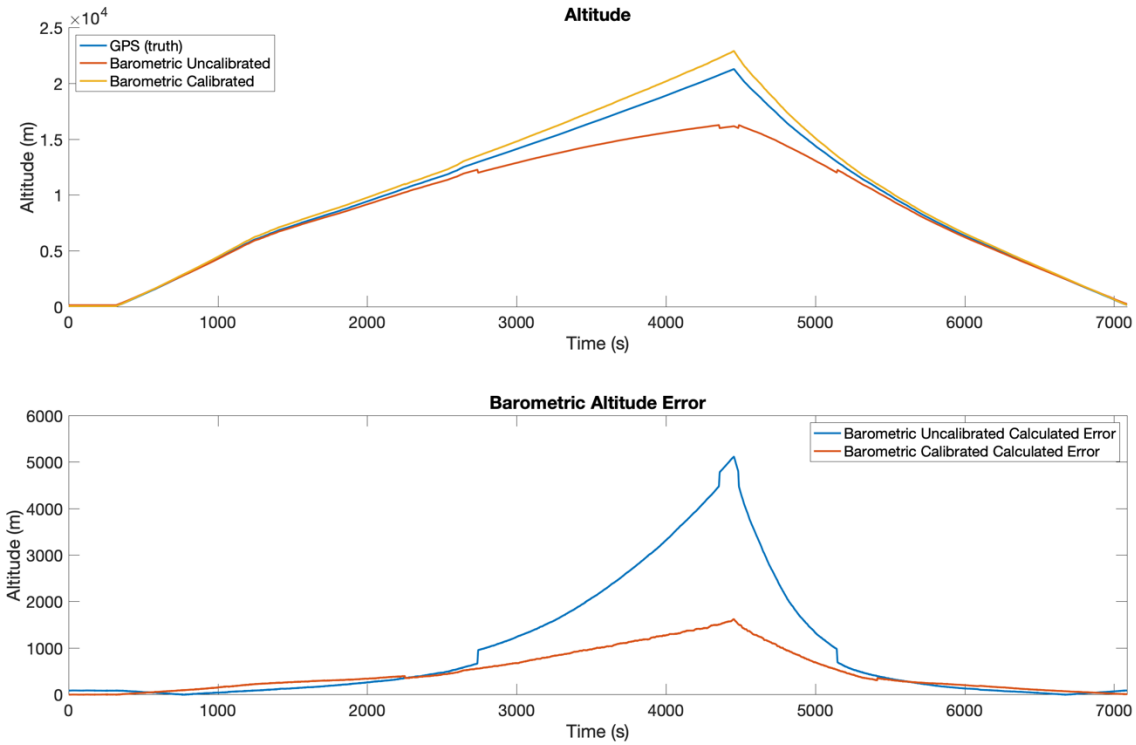

Figure 3.6: High altitude balloon test of altitude uncalibrated and calibrated calculations compared to GPS truth. Calibration occurred just before launch.

3.2.2. Inertial Measurement Unit (IMU) and Magnetometer

An inertial measurement unit (IMU) usually consists of a gyroscope and accelerometer. Many IMU boards also contain a magnetometer and temperature sensor. Except for the temperature sensor, each sensor measures in three axes. A gyroscope typically measures angular rotation rate in radians/s. An accelerometer measures linear acceleration in m/s. A magnetometer measures the magnetic field in teslas. By using the data from these three sensors, the orientation of the platform can be determined. For instance, on Earth the acceleration due to gravity is $9.8 m/s^2$. Due to a constant downward acceleration, the accelerometer shows this acceleration vector distributed across its three axes corresponding to the direction of down. This helps determine the pitch and roll of a vehicle or object. Another characteristic of Earth useful for determining orientation is magnetic north. Like the accelerometer, the magnetometer represents

the direction of the magnetic north across its three axes. This helps determine the yaw of a vehicle or object.

3.3. Conclusion

To conclude, this chapter discusses various ways of representing geographic coordinates and using georeferenced maps. It also describes two different sensors that could easily be integrated with vision-based algorithms for determining the absolute world location and orientation.

Chapter 4: Terrain Simulation and Navigation Algorithm Formulation

With the basics of image processing and geographic coordinate systems discussed in the previous two chapters, these techniques can be combined to simulate imagery and form a terrain relative navigation algorithm.

4.1. Simulating Aerial Imagery

As an initial test of the map matching algorithms, an image simulator can be used. Modelling the extrinsic and intrinsic parameters of a camera with known truth poses provides a robust test of the algorithms eliminating the variables of sensor errors. A simulator also allows the use of multiple sets of imagery. This can highlight the difference between how the algorithm works on comparing images from the same source versus comparing imagery from different sources. Several low-cost 3D visualization programs were tested with satellite imagery and digital elevation maps. However, the performance cost of these programs is high due to the overhead of the unneeded features of these software systems. This problem assumes the terrain is relatively flat and does not need elevation of every pixel to be different. Also, no additional 3D objects need to be placed into the scene, simplifying the simulation technique needed.

Instead, the simulation strategy for this thesis uses a ray tracing technique based on [18] and [19]. Given the extrinsic and intrinsic parameters of the camera, it projects a ray to the world coordinate system as shown in Figure 4.1.

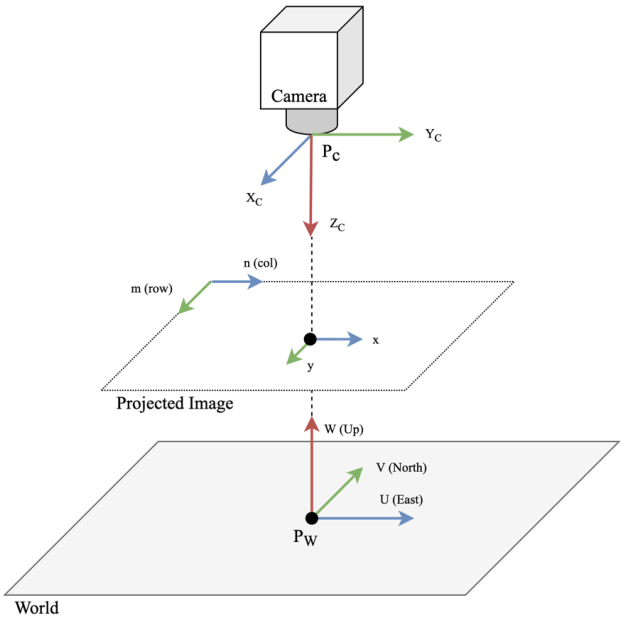

Figure 4.1: Depiction of coordinate systems of the camera with relation to the world.

This ray projection is accomplished through a series of coordinate transformations defined in Table 4.1. The first step is to transform the world coordinate frame to the camera frame. This is split into two steps to allow for the camera placed on a location or orientation that is not the center of gravity of the vehicle. The displacement of the vehicle in vehicle coordinates is simply the negative of the displacement of world coordinates in NED coordinates. The rotation is defined as the identity matrix for simplicity but could be reconfigured as needed.

$$d_w^v = \begin{bmatrix} -y_w^v \\ -x_w^v \\ -z_w^v \end{bmatrix} \tag{4.1}$$

$$R_w^v = I_3 \tag{4.2}$$

$$T_w^v = \begin{bmatrix} R_w^v & d_w^v \\ 0_{1\times3} & 1 \end{bmatrix} \tag{4.3}$$

The orientation of the camera is defined in the vehicle to camera transformation. Additional transformations could be used if the camera is mounted on a gimbal.

$$t_v^c = 0_{3\times1} \tag{4.4}$$

$$R_v^c = R_x(\theta)R_y(\phi)R_z(\psi) \tag{4.5}$$

$$T_w^v = \begin{bmatrix} R_v^c & t_v^c \\ 0_{1\times3} & 1 \end{bmatrix} \tag{4.6}$$

Table 4.1: Coordinate frame notation

Transformation	Description
$\boldsymbol{T_w^v}$	World Frame to Vehicle Frame
$\boldsymbol{T_v^c}$	Vehicle Frame to Camera Frame
$\boldsymbol{P_{sense}}$	Points of Object or Terrain in Sensed Image Frame
$\boldsymbol{P_w^p}$	Projected of Object or Terrain in Projected Image Frame
$\boldsymbol{P_w^{obj}}$	Points of Object or Terrain in World Frame

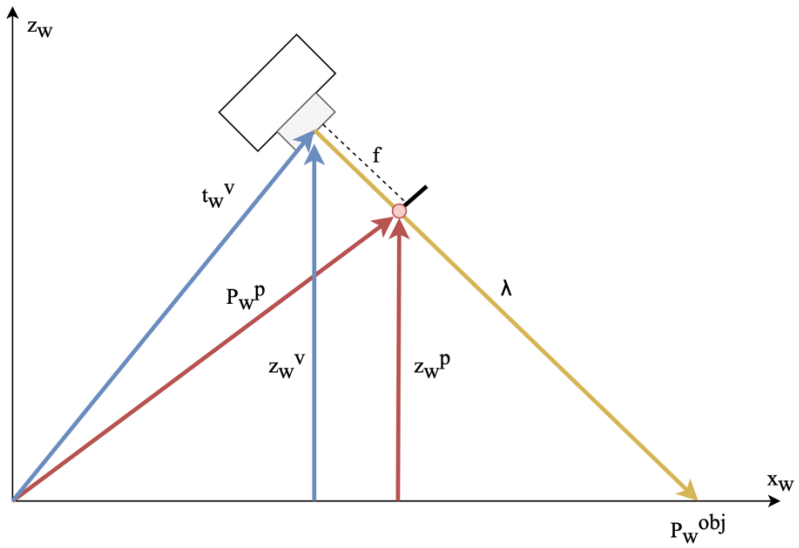

Figure 4.2: A side view of the perspective camera model

To produce a full image, these transformations should occur on each pixel in the sensed image to the world frame as depicted in Figure 4.2. A mesh grid of pixel locations is represented as P_{sense} in a $4 \times N$ matrix where N is the total amount of pixels in the sensed image, the first column is the x pixels, the second column is the y pixels, and the last two columns are filled with ones. The camera matrix K can be reformatted into a 4×4 matrix C.

$$C = \begin{bmatrix} 0 & f_x & c_x & 0 \\ f_y & 0 & c_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

The projected coordinates of the pixels is

$$P_w^p = (C\ T_w^v\ T_v^c)^{-1} P_{sense} \tag{4.8}$$

The image depth λ is defined as the following where z_p^v is the z coordinate of P_w^p.

$$\lambda = \left| \frac{z_w^v}{z_w^v - z_w^p} \right| \tag{4.9}$$

Now the projected pixels in the world frame (NED) can be determined assuming the terrain plane is flat.

$$q_{L_x} = \lambda P_{w\,x}^p \tag{4.10}$$

$$q_{L_y} = \lambda P_{w\,y}^p \tag{4.11}$$

$$q_{L_z} = \lambda \tag{4.12}$$

$$P_w^{obj} = (C\ T_w^v\ T_v^c)^{-1} q_L \tag{4.13}$$

Once in the world frame in NED coordinates, they are converted into geodetic. Through interpolation, the geodetic coordinates are converted to pixel coordinates in the map resulting in P_{obj}^{ref}. Each pixel in P_{sense} is assigned the intensity in the corresponding pixel in P_{obj}^{ref}. This process is vectorized for computational efficiency and summarized in Figure 4.3.

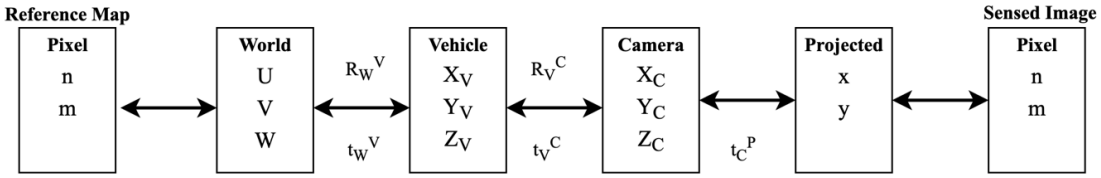

Figure 4.3: Transformation to pixel coordinates from world coordinates.

As a sanity check to verify the simulator is producing the correct image, Figure 4.4 shows the comparison between a real image and a simulated image. The homography between the sensed

image and reference map is found with the same process used for the simulator. Instead of passing through all pixel locations, only the reference map pixels locations of the four corners of the sensed image are calculated. With the correspondence of four locations in the sensed and reference image, the homography matrix is calculated with the process outlined in Section 2.2.2.5. A visual inspection is performed to verify the homography and simulator produce the correct correspondence as shown in Figure 4.5.

Figure 4.4: (Left) Simulated image with Google reference imagery (Right) Aerial image captured from High Altitude Balloon

Figure 4.5: Correspondence between simulated sensed image and reference map with homography. Lighter shaded section is sensed image and darker shaded background is reference map.

4.2. Terrain Relative Navigation (TRN) Approach

The general algorithm to find the pose from image correspondence is listed below:

1. Receive an input of a coarse pose from a prior state, coarse search algorithm, or a testing dataset.
2. Detect Shi Tomasi corners in the sensed image and rank them by strength. Pick the N strongest corners that are outside of a set pixel distance from other selected corners.
3. For each corner, construct a square template around the corner.
4. Transform the template into the reference image frame using the coarse pose estimate.
5. Generate a dynamically sized window in the reference image centered on the transformed corner.
6. Sweep the template over the search window and correlate the template to each section of the search window using normalized cross-correlation.
7. Check the quality of the correlation using several criteria to determine if each match is valid or invalid.
8. Perform RANSAC with homography to eliminate outlier matches.
9. Use the georeferenced position of each matched point on the reference image along with the corner location in the sensed image to compute the pose of the camera with PnP.

4.3. Software Architecture and Design

A full software package is developed in Python to test these algorithms outlined in Figure 4.6. It can either use a simulated trajectory and simulate imagery using Google or Bing satellite imagery or import data from a Robot Operating System (ROS) “bag” file. This is compiled with one of the “dataset_generator” modules to save the preprocessed data to file. Optionally for the ROS bag files, a “point_picker” script and “pose_adjuster” script are used to determine the truth

pose for frames where reliable GPS and IMU data is not present. The "point_picker" allows the user to manually pick the point correspondence in the sensed and reference image and the "pose_adjuster" uses PnP to determine the truth pose. The "map_matcher" module communicates with various submodules to perform the TRN algorithm outlined in the previous section. The resulting analysis is plotted using Matplotlib and saved to csv files. Majority of the image processing is performed with OpenCV. Numpy is used to process the matrix calculations. Various configurations are stored in Python configuration classes and YAML files.

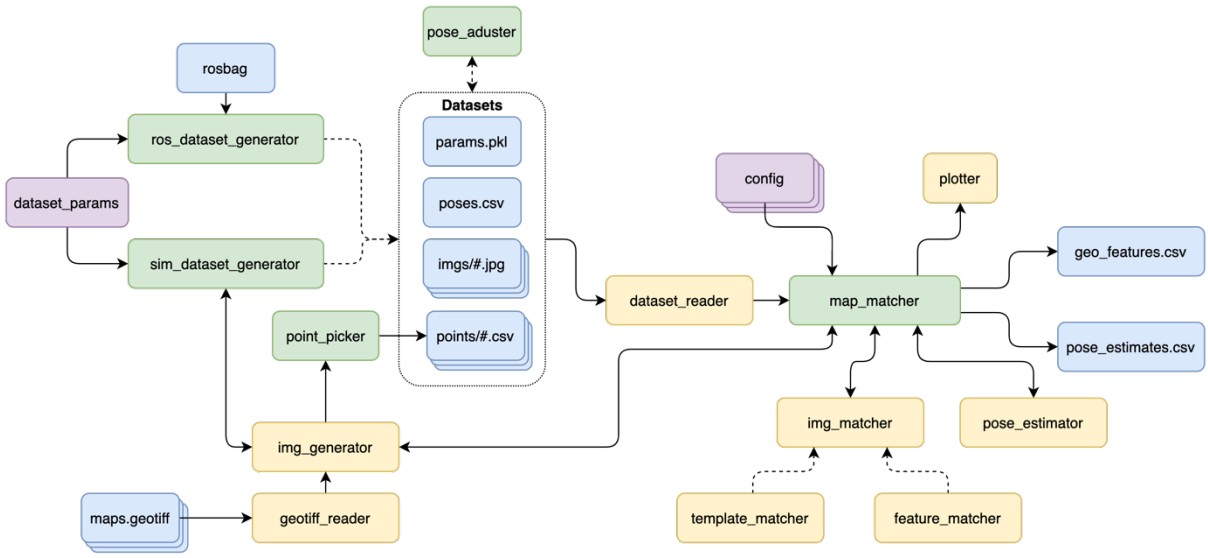

Figure 4.6: Full software architecture where the green blocks are Python scripts the user runs. The yellow blocks are modules inherited by the main scripts. The blue blocks are data files and the purple blocks are configuration files.

4.4. Conclusion

In conclusion, this chapter details the use of a 6DOF aerial imagery simulator and the overall terrain relative navigation algorithm architecture tested in this thesis.

Chapter 5: Data Acquisition and Evaluation

In order to analyze the performance of the terrain relative navigation algorithm on real world data, Unmanned Aerial Vehicles and High Altitude Balloons were used. This chapter outlines the basics of the aerial platforms chosen and some data evaluation metrics to analyze the data.

5.1. Unmanned Aerial Vehicles

Unmanned Aerial Vehicles (UAVs) or Unmanned Aerial Systems (UASs) describes a wide variety of remote controlled or autonomous aircraft. They are used for a variety of purposes like surveying, data collection, or short-range lightweight cargo transportation. Many are highly dependent upon GPS systems for navigation. Using a vision-based navigation system is a way to offer redundancy in situations where GPS signals are denied. UAVs can be found in many different forms. The two most common categories are fixed wing and rotary wing. Fixed wing UAVs fly like traditional airplanes and are typically outfitted with a single propeller or jet engine. Rotary wing UAVs are classified by the number of vertical rotors. The most common are 1, 4, 6, and 8.

Figure 5.1: (Left) Tarot octocopter outfitted with data collection system. (Right) Example image captured by down-look camera.

UAVs also come in a variety of sizes, but civilian UAVs are limited to 55lbs according to the Federal Aviation Administration (FAA) [20]. Another notable FAA regulation for civilian

UAVs pertains the maximum flight altitude of 400ft above ground level without a waiver. The larger DoD fixed-wing UAVs can reach a wider variety of altitudes upwards of 30,000 to 50,000ft depending on the capabilities of the UAV. For the purposes of this thesis, all UAV data is captured by an octocopter (Figure 5.1) limited to the altitude of 400ft. The data collection system is the same as used in the high altitude balloon discussed in more detail in the next section. SN01 refers to the first high altitude balloon electronics system and SN02 refers to the second high altitude balloon electronics system. See Table 5.1 and Table 5.2 for specifications of these two systems.

Table 5.1: SN01 Collection System Specs

Characteristic	Value	Characteristic	Value
Processor	Raspberry Pi 4B 2GB (ARM Cortex-A72)	Camera Sensor	OV5647
IMU 1	BNO055	Lens Focal Length	4mm
IMU 2	ICM20948	Image Resolution	2592 x 1944
GPS	MT3339	Sensor Area	3673.6µm x 2738.4µm
Barometer	MPL3115A2	Shutter Type	Rolling

Table 5.2: SN02 Collection System Specs

Characteristic	Value	Characteristic	Value
Processor	Raspberry Pi 4B 4GB (ARM Cortex-A72)	Camera Sensor	OV5647
IMU 1	BNO080	Lens Focal Length	4mm
IMU 2	ICM20948	Image Resolution	2592 x 1944
GPS	uBLOX MAX-M8Q-10	Sensor Area	3673.6µm x 2738.4µm
Barometer (External)	MPL3115A2	Shutter Type	Rolling
Barometer (Internal)	LPS22HB		

5.2. High Altitude Balloons

While UAVs are limited to 400ft according to the FAA, uncontrolled high altitude balloons (HABs) can be operated at unlimited altitudes. By constructing a lightweight payload box and

filling a specialized latex helium balloon, altitudes of 70,000 – 110,000ft are achievable before the balloon bursts due to low air pressure. The payload then travels by parachute back down to the surface allowing the data to be recovered. This presents the opportunity to collect aerial imagery and navigational data at a wide variety of altitudes. Figure 5.2 shows the payload suspended from the balloon. See Appendix A and Appendix B for more detailed information on the payload design and flight characteristics.

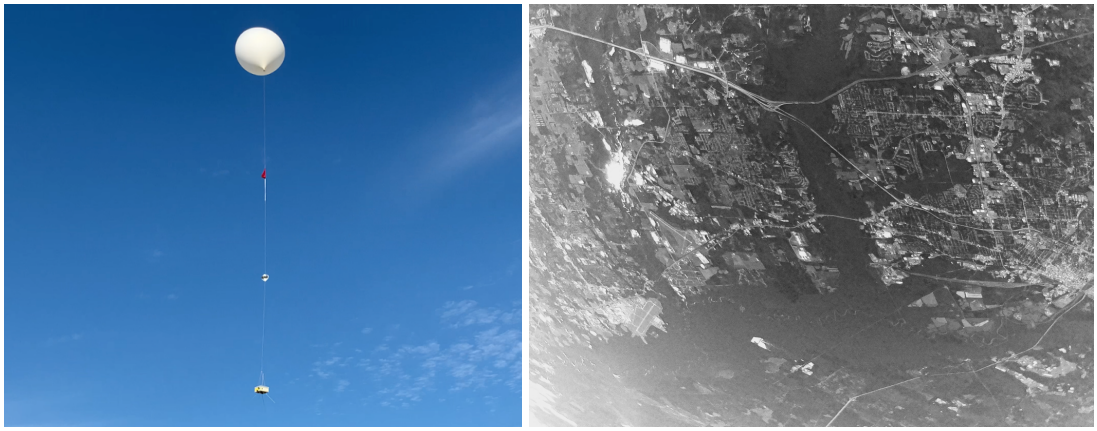

Figure 5.2: (Left) High altitude balloon data collection system (Right) Example image captured by down-look camera during parachute descent over Macon, GA.

5.3. Data Evaluation Metrics

The results from testing on simulated and collected datasets is evaluated in two main ways: matching accuracy and estimation error. Ultimately low matching accuracy can contribute to high estimation error. For this thesis, a good match is defined as one that is within 100 pixels of the true match in the reference image frame. A valid match satisfies the conditions stated in Section 2.2.3, Step 5. Invalid matches are rejected from being used in the PnP estimation calculations. Therefore, in an ideal case, all valid matches are good, and all invalid matches are bad. To produce the best pose estimate, the matching algorithm should be tuned to maximize good, valid matches and minimize good, invalid matches. Conversely, bad valid matches should be minimized, and bad, invalid matches should be maximized as needed. To quantify this, a match score is proposed

by weighing all of these scenarios. Let P_{gv} represent the percentage of good and valid matches and P_{bi} represent the percentage of bad and invalid matches. The match score is defined as Q where 100 is a perfect score.

$$Q = 100P_{gv} + 25P_{bi} - 25P_{gi} - 100P_{bv} \tag{5.1}$$

The desired output of a localization algorithm is the pose estimate. PnP is used to determine the translation vector and rotation matrix between the sensed image and the reference map using the point correspondence determined with the image matching algorithm. The estimation error is defined as the difference in the fine estimated pose and the truth pose. The truth pose is determined in simulation, with a point matching GUI (Figure 5.3), or with GPS and IMU data.

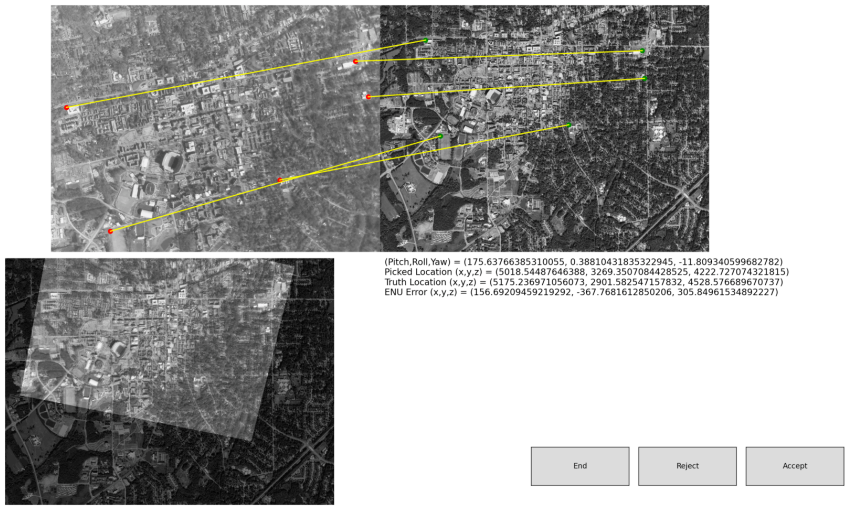

Figure 5.3: Point matching GUI for determining translation and rotation of imagery to map.

5.4. Conclusion

This chapter presented how the data is collected an evaluated. A UAV is used to collect data at lower altitudes, whereas, a HAB is used to collect data at higher altitudes. The aerial imagery and estimated coarse pose are used as inputs to the terrain relative navigation algorithm and evaluated on its successful match rate and its ability to estimate the true pose.

Chapter 6: Simulated Results and Analysis

This chapter presents the results from running the terrain relative navigation algorithm on simulated imagery.

6.1. Simulated Test #1: Google Sensed to Google Reference Map for Orthogonal Case

This simulation represents the control case. The simulated images are down sampled from Google Satellite Maps and are compared to a Google reference map. See Table 6.1 for the information regarding the setup of this test. The coarse pose error is zero mean Gaussian with a standard deviation of 50m in x and y, 25m in z, and 3˚ in pitch, roll, and yaw. This is modelled after the uncertainty in the z axis being lower than the x and y with the use of a barometric altimeter. A constant velocity model is used to move the camera in a downward movement from 5000m to 1000m AGL for 100 frames. Figure 6.1 and Figure 6.2 depict the difference between the truth and coarse pose estimate.

Table 6.1: Parameters for Simulated Test #1

Parameter	**Value**
Sensed Imagery Source	Google Satellite Maps
Reference Map Imagery Source	Google Satellite Maps
Elevation Map Source	USGS 3DEP
Location	Auburn, AL
Simulated Coarse Error σ (ENU)	(50m,50m,25m)
Simulated Coarse Error σ (Pitch, Roll, Yaw)	(0˚, 0˚, 0˚)
Translation and Altitude Error Range (ENU)	(100m,100m,75m)
Attitude Error Range (Pitch, Roll, Yaw)	(0˚, 0˚, 0˚)
Number of Corners Selected	100
Template Size (Pixels)	200

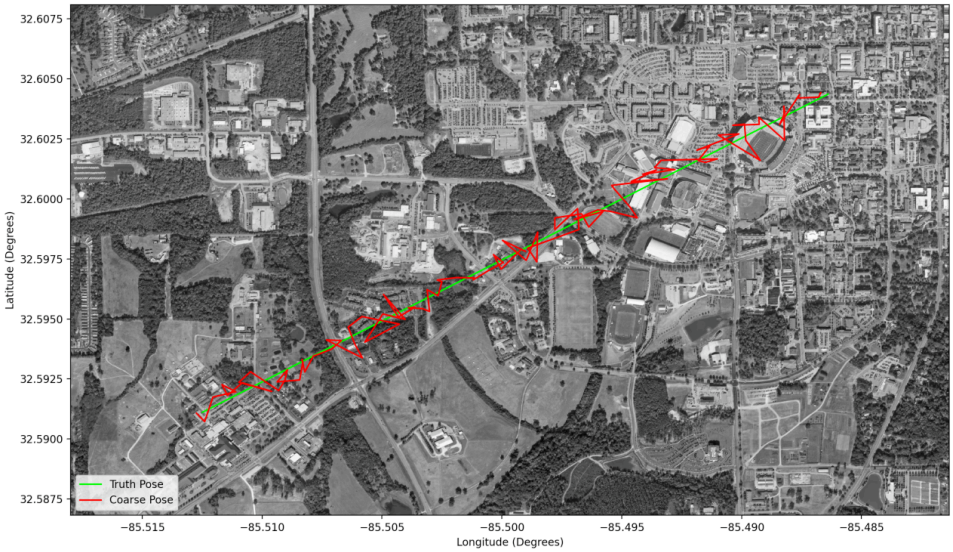

Figure 6.1: Path of camera on Bing reference map.

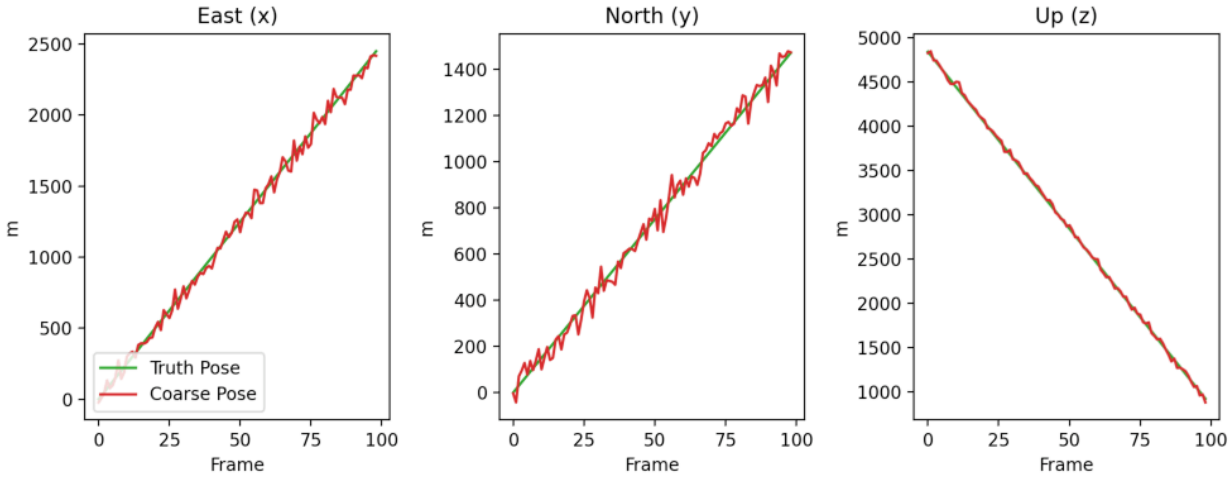

Figure 6.2: ENU translation plot of truth and coarse pose estimates for simulated dataset.

For this dataset, almost all of the matches are considered good and valid as expected since the imagery source is the same (Figure 6.3). Even with some coarse pose error, the algorithm is very successful in finding good matches. The match score is perfect for most of the frames (Figure 6.4). The estimated pose from PnP, therefore, is within the error ranges for the entire dataset (Figure 6.5).

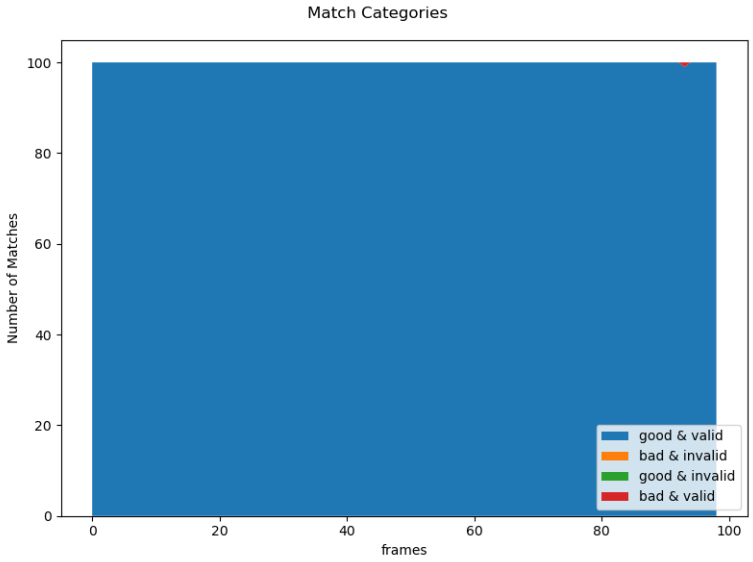

Figure 6.3: Stacked plot of matched category distribution for each frame. A good match is one that is within 50 pixels of the truth. A match is valid if its correlation passes the height and width criteria. In this dataset all the matches are good and invalid except for one near frame 95.

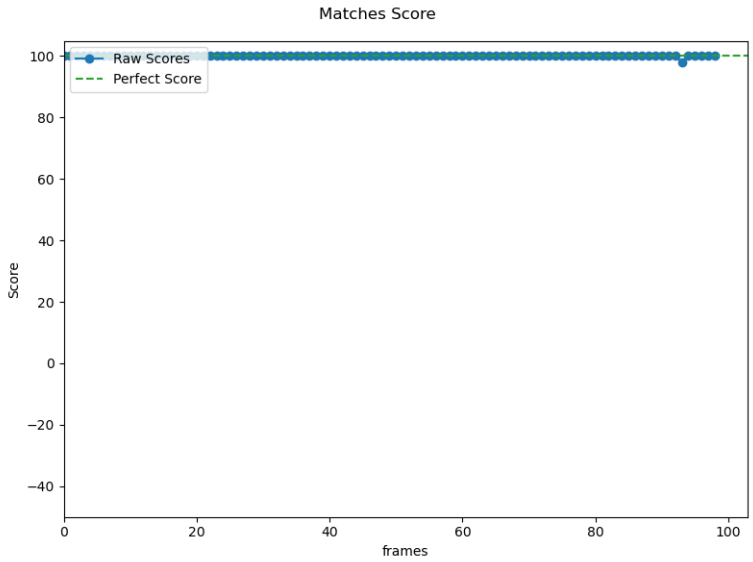

Figure 6.4: Match score as defined in Section 5.3. In this simulation all frames have a perfect score except for one.

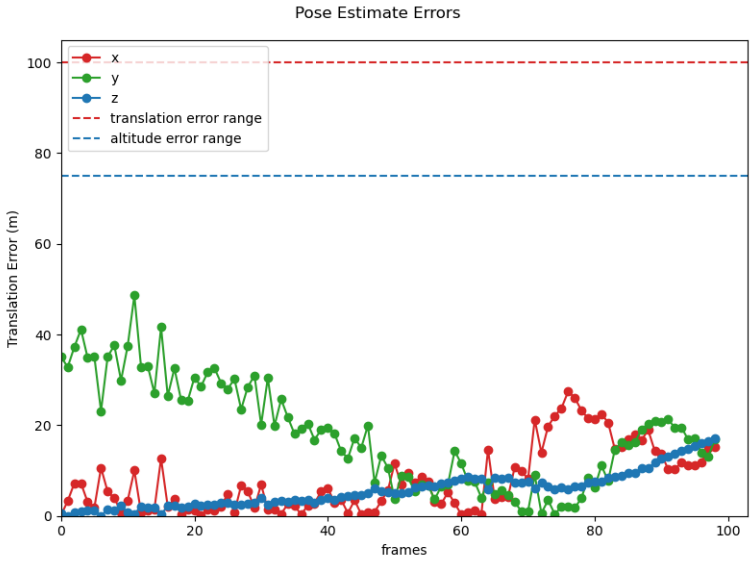

Figure 6.5: Raw fine estimation error in translation and altitude. The dashed lines represent the error range the algorithm was using to set the dynamic windows.

6.2. Simulated Test #2: Google Sensed to Bing Reference Map for Orthogonal Case

This simulation tests the robustness of the algorithm against imagery from another source. The same parameters are used to generate a path from the previous simulation. The simulated images are down sampled from Google Satellite Maps and are compared to a Bing reference map. See Table 6.2 for the information regarding the setup of this test.

Table 6.2: Parameters for Simulated Test #2

Parameter	Value
Sensed Imagery Source	Google Satellite Maps
Reference Map Imagery Source	Bing Aerial Maps
Elevation Map Source	USGS 3DEP
Location	Auburn, AL
Simulated Coarse Error σ (ENU)	(50m,50m,25m)
Simulated Coarse Error σ (Pitch, Roll, Yaw)	(0˚, 0˚, 0˚)
Translation and Altitude Error Range (ENU)	(100m,100m,75m)
Attitude Error Range (Pitch, Roll, Yaw)	(0˚, 0˚, 0˚)
Number of Corners Selected	100
Template Size (Pixels)	200

For this test, the matching performance is slightly degraded as shown in Figure 6.6 and Figure 6.7. There are some false positives where matches are marked valid even though they are bad, especially as the camera decreases in altitude. This increase in bad matches might be due to the difference in imagery where new building construction in the area has degraded the performance. There is also some false negatives where matches are marked invalid even though the matches are good. Overall, the match score decreases as altitude decreases. The pose estimates are slightly worse but of a similar magnitude to the control case.

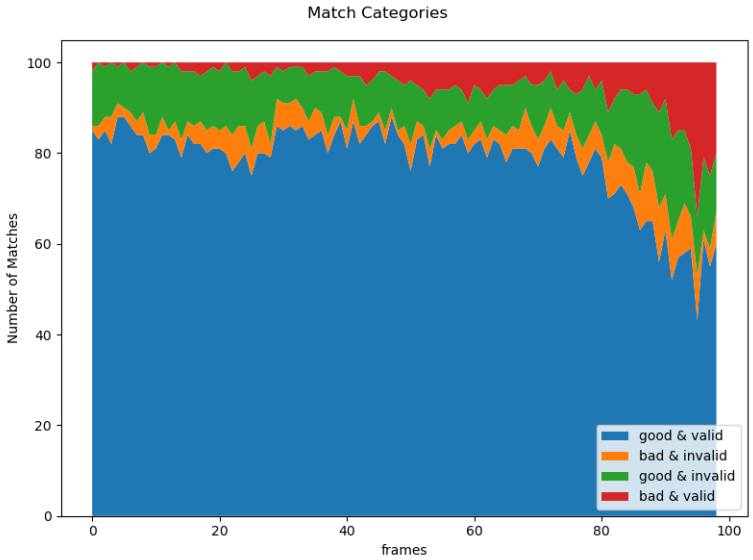

Figure 6.6: Stacked plot of matched category distribution for each frame. A good match is one that is within 50 pixels of the truth. A match is valid if its correlation passes the height and width criteria. In this dataset all the matches are good and invalid except for one near frame 95.

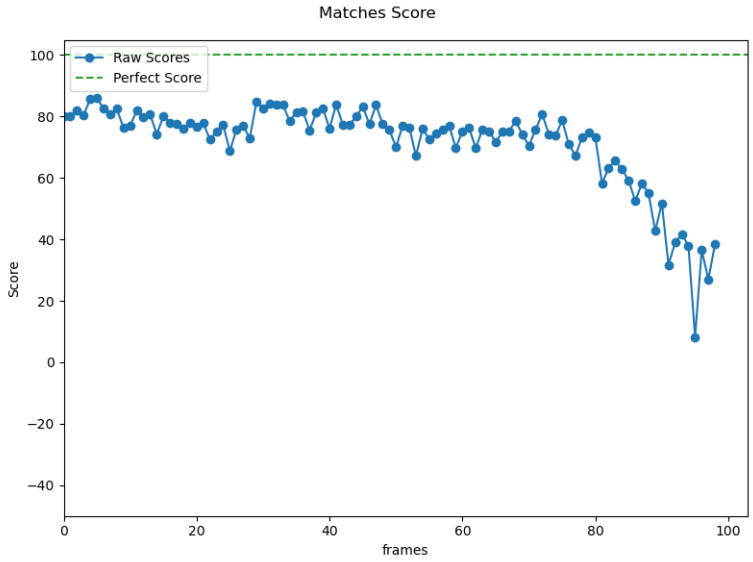

Figure 6.7: Match score as defined in Section 5.3.

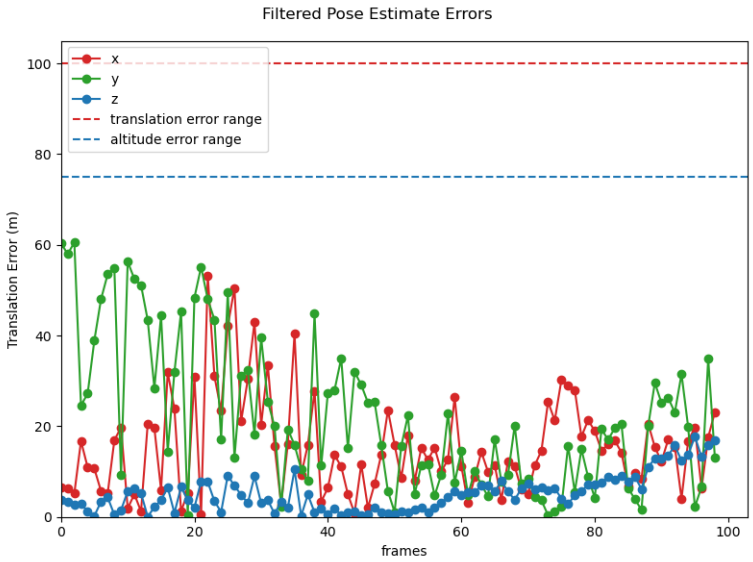

Figure 6.8: Filtered fine estimation error in translation and altitude. The dashed lines represent the error range the algorithm was using to set the dynamic windows.

6.3. Simulated Test #3: Google Sensed to Bing Reference Map for Non-Orthogonal Case

This simulation tests the robustness of the algorithm against imagery from another source when the coarse pose is non-orthogonal to the reference map. See the parameters used in Table 6.3. The translation path is the same as the previous two tests except the coarse orientation is different than truth as shown in Figure 6.9.

Table 6.3: Parameters for Simulated Test #3

Parameter	Value
Sensed Imagery Source	Google Satellite Maps
Reference Map Imagery Source	Bing Aerial Maps
Elevation Map Source	USGS 3DEP
Location	Auburn, AL
Simulated Coarse Error σ (ENU)	(50m,50m,25m)
Simulated Coarse Error σ (Pitch, Roll, Yaw)	(3˚, 3˚, 3˚)
Translation and Altitude Error Range (ENU)	(100m,100m,75m)
Attitude Error Range (Pitch, Roll, Yaw)	(5˚, 5˚, 5˚)
Number of Corners Selected	100
Template Size (Pixels)	200

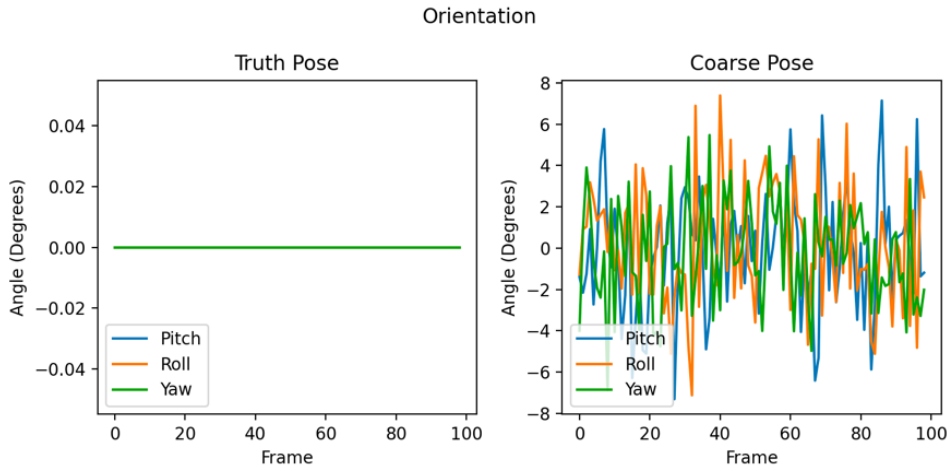

Figure 6.9: Orientation of camera for truth and coarse pose

The matching performance is shown in Figure 6.10 and Figure 6.11. In many frames, it finds many good matches, but outliers exist where the algorithm is unable to find good matches. This is likely due to the larger rotation differences for those frames.

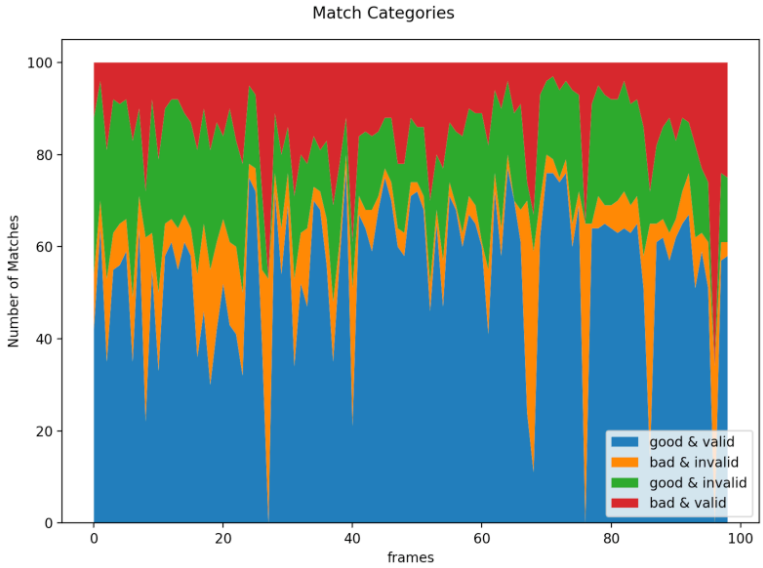

Figure 6.10: Stacked plot of matched category distribution for each frame. A good match is one that is within 50 pixels of the truth. A match is valid if its correlation passes the height and width criteria.

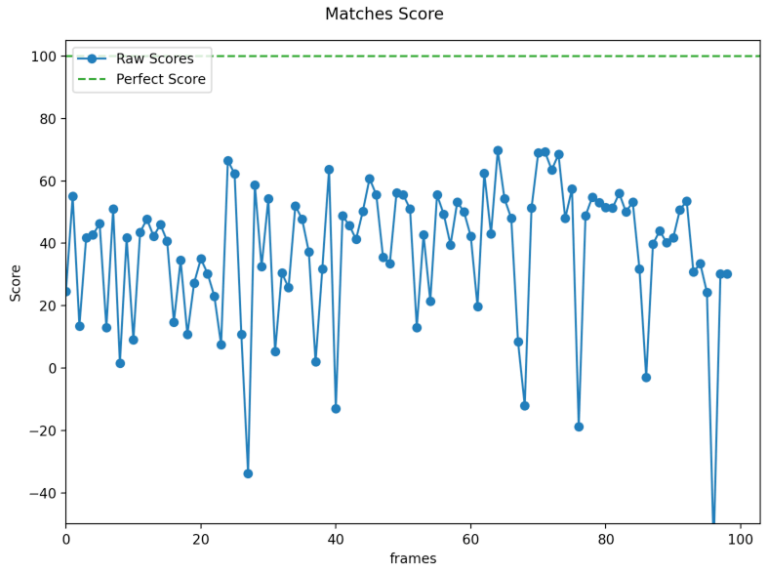

Figure 6.11: Match score as defined in Section 5.3.

RANSAC is used to eliminate many of the bad matches but fails in three of the frames as demonstrated in Figure 6.12. In many frames, the PnP is able to estimate the pose within the translation and altitude error ranges (Figure 6.13). High pose estimate errors exist in certain frames, but these are due to the low number of good matches and corresponds to the frames which

pass through bad valid RANSAC inliers. By comparing the estimated pose to 3σ of the coarse pose range, these pose estimate outliers are eliminated (Figure 6.14). The resulting fine pose estimates narrow the coarse estimate closer to truth fairly consistently in this simulated case. As the camera approaches the ground in the later frames, the error in all three axes stays below 25m.

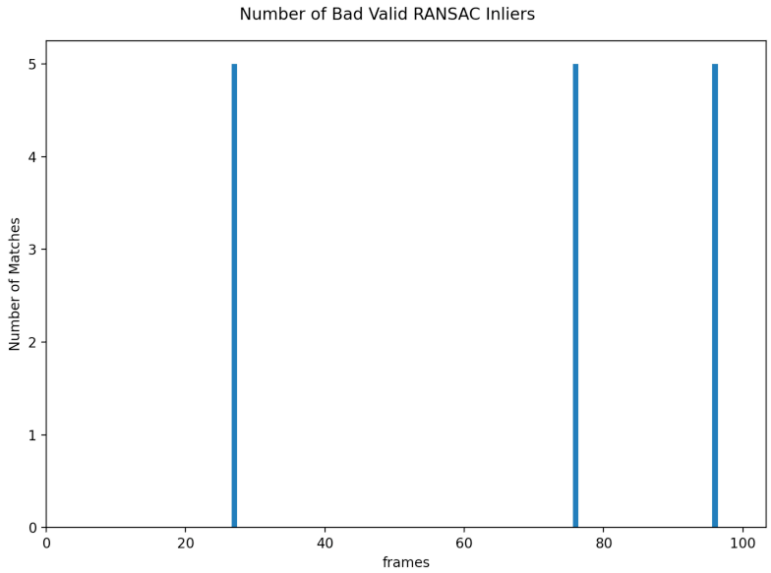

Figure 6.12: Bar graph of number of bad valid RANSAC inlier matches for each frame passed through to the PnP estimation algorithm

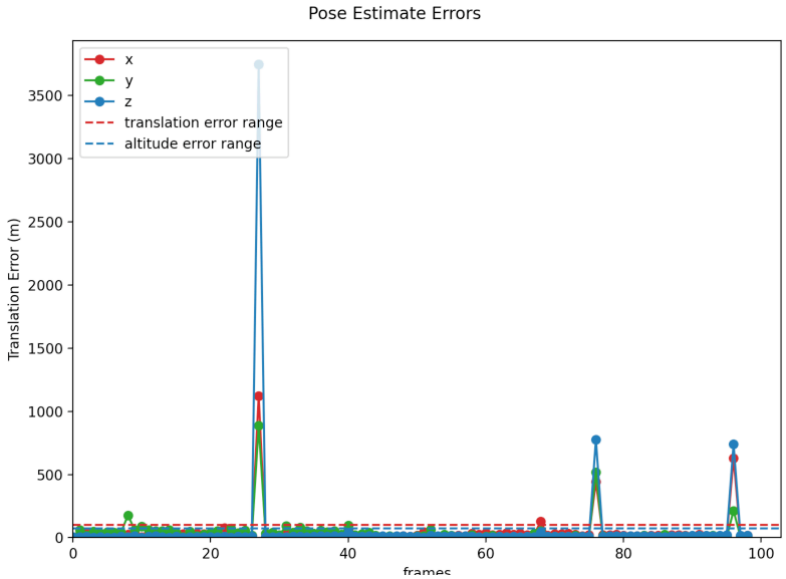

Figure 6.13: Raw fine estimation error in translation and altitude. The dashed lines represent the error range the algorithm was using to set the dynamic windows.

Figure 6.14: Pose estimates within 3σ of coarse error range.

6.4. Conclusion

Overall, the algorithm performed very well on the simulated imagery. In the control case, the match rate approximated 100%. In the other simulation cases, the match rate is very strong. With the highly successful match rate, the pose estimates errors were very low. The slight errors are most likely due to slight pixels shifts caused by sampling differences and ability of PnP to fit the best pose for a given set of matches.

Chapter 7: Experimental Results and Analysis

This chapter presents the results from running the terrain relative navigation algorithm on experimentally collected imagery.

7.1. Experimental Test #1: UAV sensed imagery to Bing Reference Map over NCAT Test Track near Opelika, AL

This dataset tests the robustness of the algorithm at low altitudes. See Table 7.1 for the information regarding the setup of this test. The images were captured by the SN02 payload mounted on the bottom of the UAV described in Section 5.1. It was flown at Auburn University's National Center of Asphalt Technology (NCAT) test track. The coarse pose is determined with the position of the GPS and the orientation of the IMU for simplicity. Figure 7.1, Figure 7.2, and Figure 7.3 show the path the UAV took and the orientation of the camera during the flight. The pitch and roll remain relatively stable during the course of the flight and the yaw shows some distinct turns.

Table 7.1: Parameters for Experimental Test #1

Parameter	**Value**
Sensed Imagery Source	UAV SN02
Reference Map Imagery Source	Bing Satellite
Elevation Map Source	USGS 3DEP
Location	Auburn, AL
Simulated Coarse Error σ (ENU)	(0m,0m,0m)
Simulated Coarse Error σ (Pitch, Roll, Yaw)	(0˚, 0˚, 0˚)
Translation and Altitude Error Range (ENU)	(100m,100m,75m)
Attitude Error Range (Pitch, Roll, Yaw)	(5˚, 5˚, 5˚)
Number of Corners Selected	100
Template Size (Pixels)	200

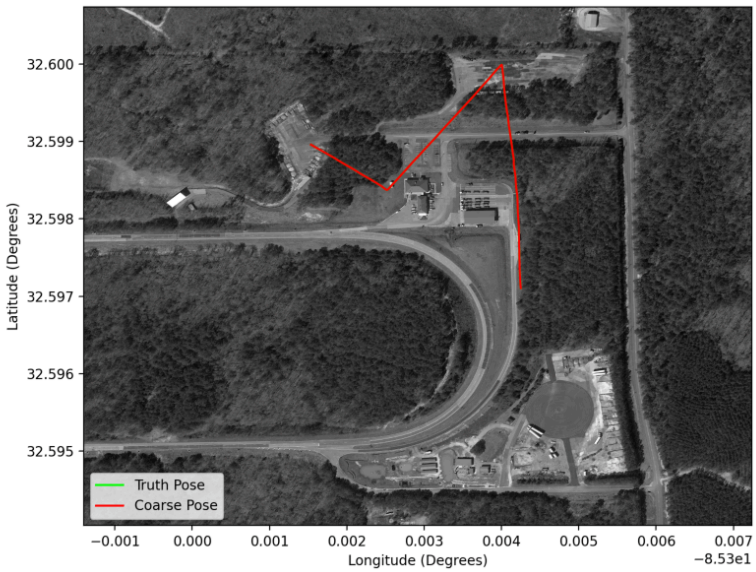

Figure 7.1: Path of camera on Google reference map.

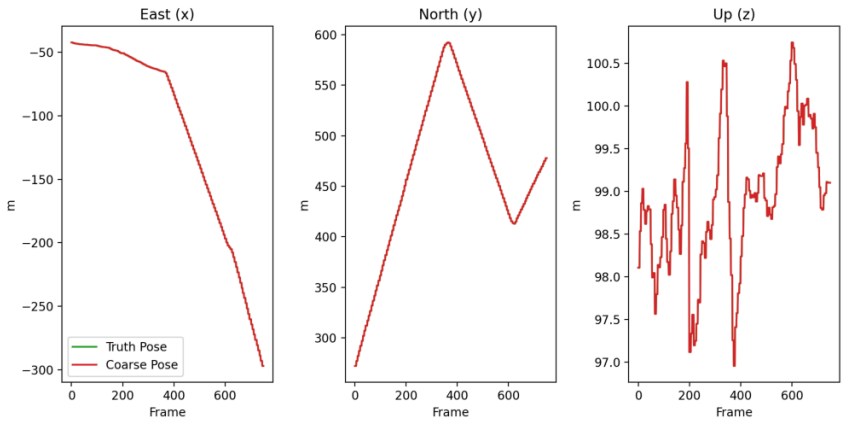

Figure 7.2: ENU translation plot of truth and coarse pose estimates for experimentally collected dataset.

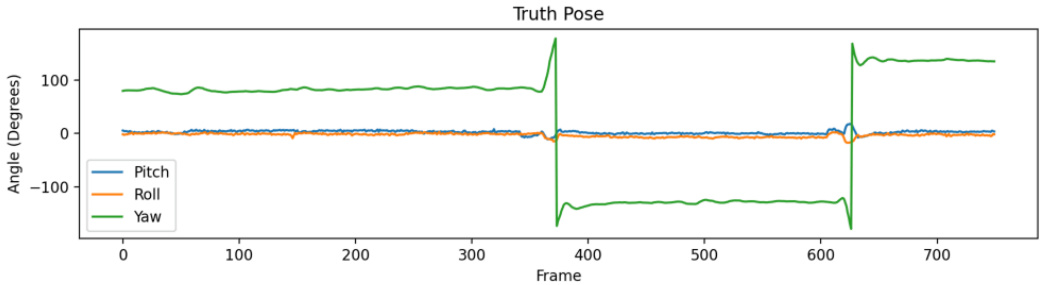

Figure 7.3: Orientation of camera for truth pose

Figure 7.4 shows the matching results and Figure 7.5 shows the match score for a snippet of the flight from frame number 400 to 500. This snippet is chosen due to the most consistent

and distinct features between the sensed and reference imagery. The successful match rate is about 15% and the match score is negative due to the high amount of bad and valid matches. The reasons for this low performance are due to the differences between the sensed imagery and reference map due to environmental, illumination, human caused, and perspective changes.

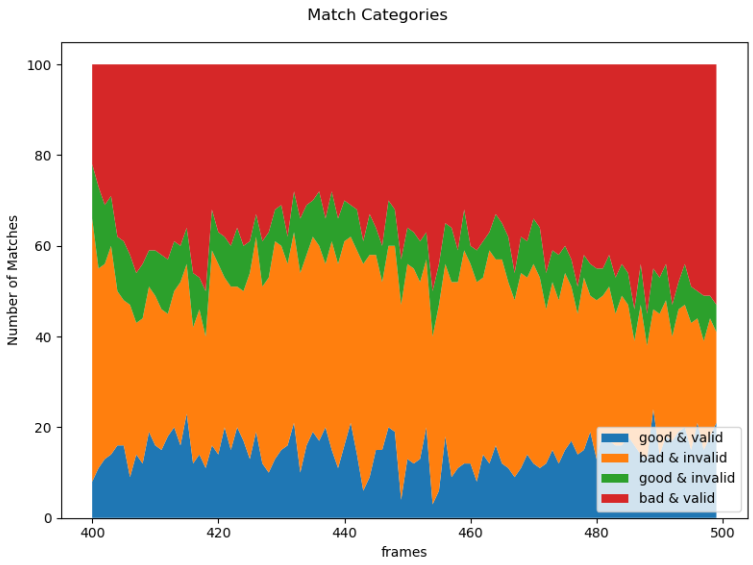

Figure 7.4: Stacked plot of matched category distribution for each frame. A good match is one that is within 50 pixels of the truth. A match is valid if its correlation passes the height and width criteria. In this dataset all the matches are good and invalid except for one near frame 95.

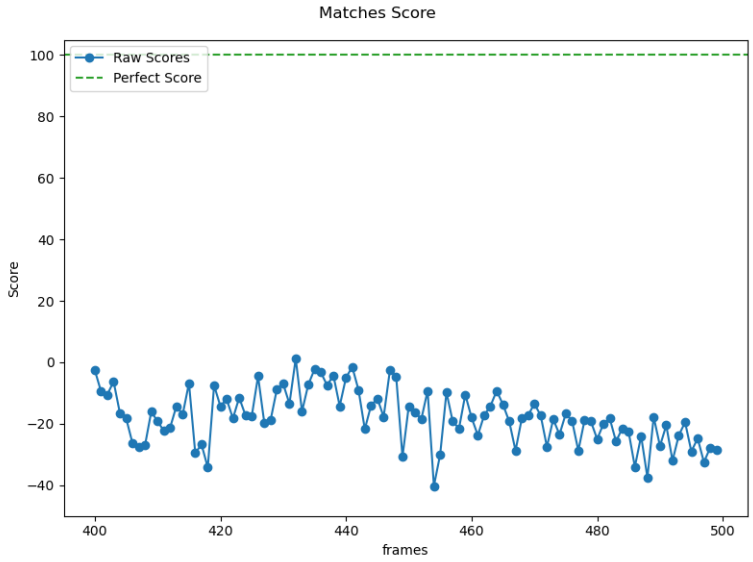

Figure 7.5: Match score as defined in Section 5.3.

Figure 7.6 shows the bad valid RANSAC inliers. Some frames have a high number of bad valid inliers but others show zero. This indicates that RANSAC is sometimes able to filter out most of the bad and valid matches when there is a sufficiently high number of good and valid matches. According to the pose estimate plots in Figure 7.7 and Figure 7.8, there is a sequence of good estimates between frame 410 and 450. In these frames, the translational error is within 10m and the altitude error is within 20m. This low error is in a similar magnitude to the 2.5m error of standard GPS systems.

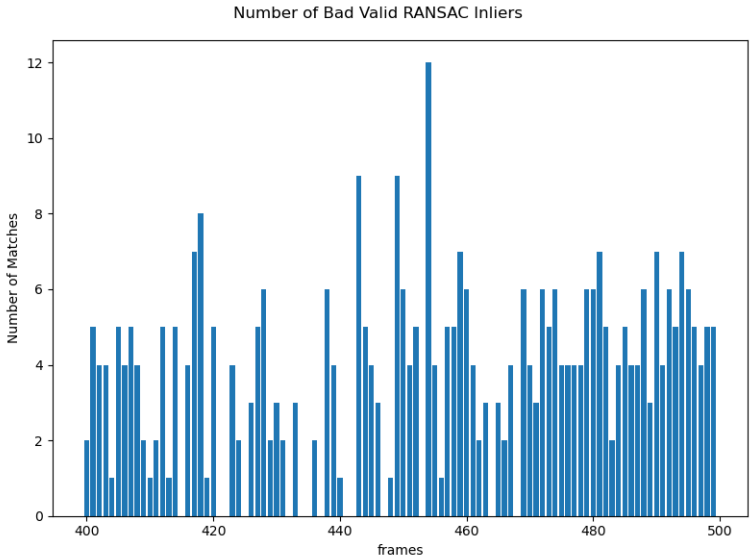

Figure 7.6: Bar graph of number of bad valid RANSAC inlier matches for each frame passed through to the PnP estimation algorithm

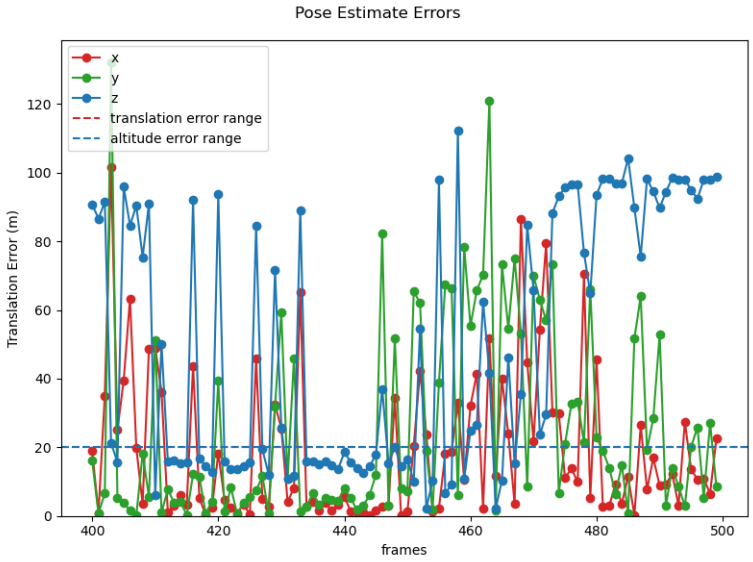

Figure 7.7: Raw fine estimation error in translation and altitude. The dashed lines represent the error range the algorithm was using to set the dynamic windows.

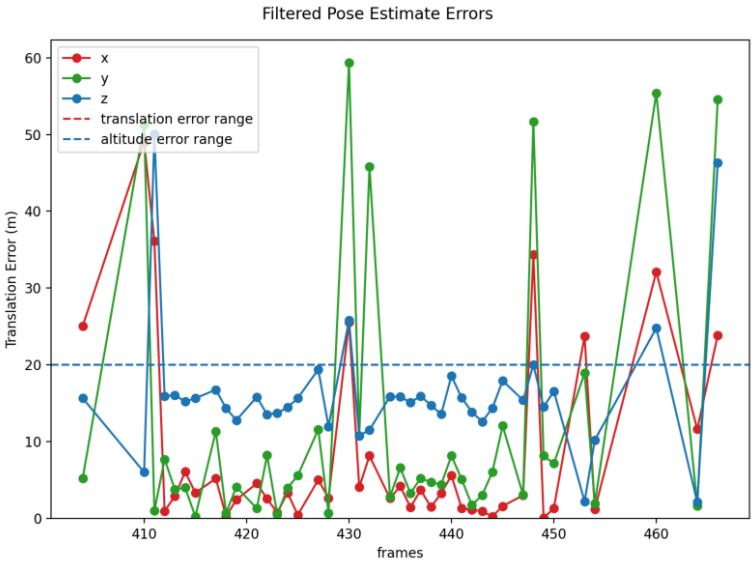

Figure 7.8: Pose estimates within 3σ of coarse error range.

7.2. Experimental Test #2: HAB sensed imagery to Bing Reference Map over Auburn, AL with SN01

This dataset tests the robustness of the algorithm at higher altitudes. See Table 7.2 for the information regarding the setup of this test.

Table 7.2: Parameters for Experimental Test #2

Parameter	Value
Sensed Imagery Source	HAB SN01
Reference Map Imagery Source	Bing Satellite
Elevation Map Source	USGS 3DEP
Simulated Coarse Error σ (ENU)	(0m,0m,0m)
Simulated Coarse Error σ (Pitch, Roll, Yaw)	(0˚, 0˚, 0˚)
Translation and Altitude Error Range (ENU)	(100m,100m,75m)
Attitude Error Range (Pitch, Roll, Yaw)	(5˚, 5˚, 5˚)
Simulated Coarse Error σ (ENU)	(50m,50m,25m)
Number of Corners Selected	100
Template Size (Pixels)	200

To collect experimental imagery, a high altitude balloon (HAB) was launched from just west of Auburn, AL and flown up to about 24km (80,000ft). The truth is determined by manually picking point correspondences and using the PnP algorithm to determine the pose. Because the truth pose is not orthogonal to the ground, the coarse pose is identical to the truth for simplicity in testing. Like the simulated case, the dynamic windows are based on a translation error range of 100m, an altitude error range of 75m, and a rotational error range of 5°. This dataset uses 10 images during ascent from 3270m (10,728ft) to 4181m (13717ft) above MSL where the elevation of the terrain is approximately 200m.

Analyzing the results of processing this dataset with the algorithm, Figure 7.9 and Figure 7.10 demonstrate how in the real imagery case, the number of good matches drops significantly compared to simulation. The number of bad valid RANSAC inliers (Figure 7.11) is nonzero on most of the frames. One particularly high amount of bad valid RANSAC inliers occurs on Frame 4 which is an example where half of the sensed image is outside of the reference map showing a decrease in good matches.

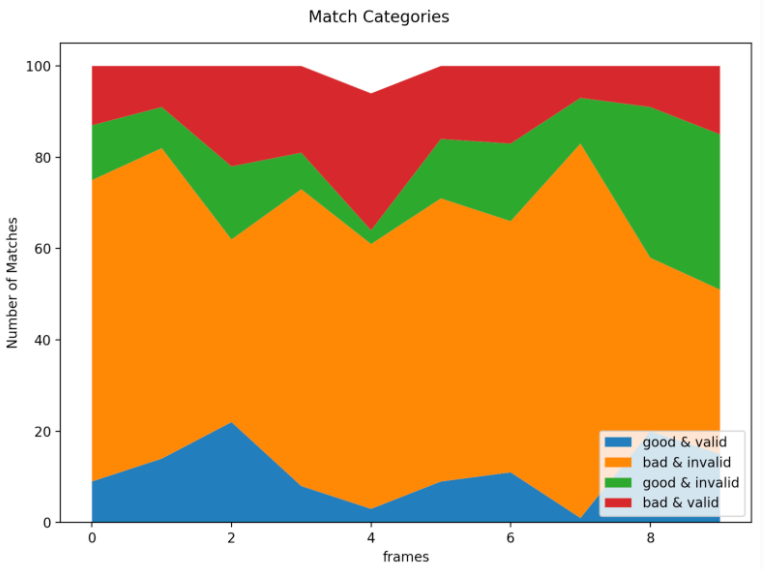

Figure 7.9: Stacked plot of matched category distribution for each frame. A good match is one that is within 50 pixels of the truth. A match is valid if its correlation passes the height and width criteria.

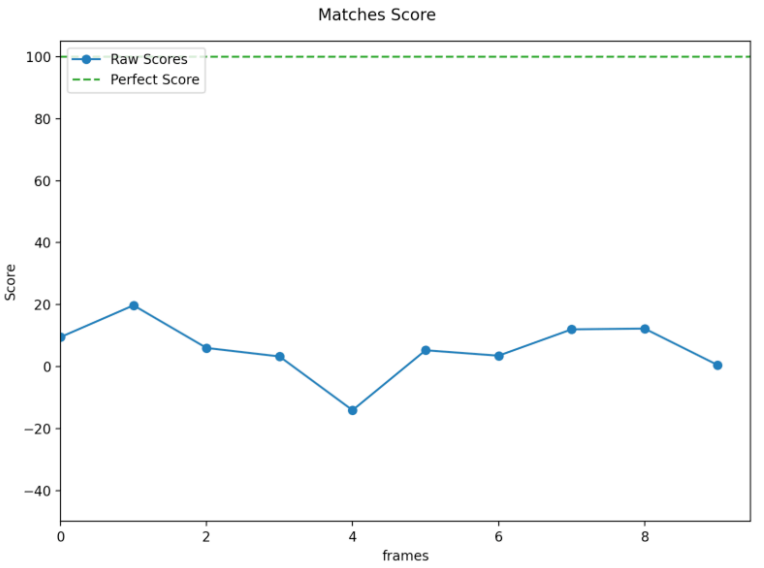

Figure 7.10: Match score as defined in Section 5.3.

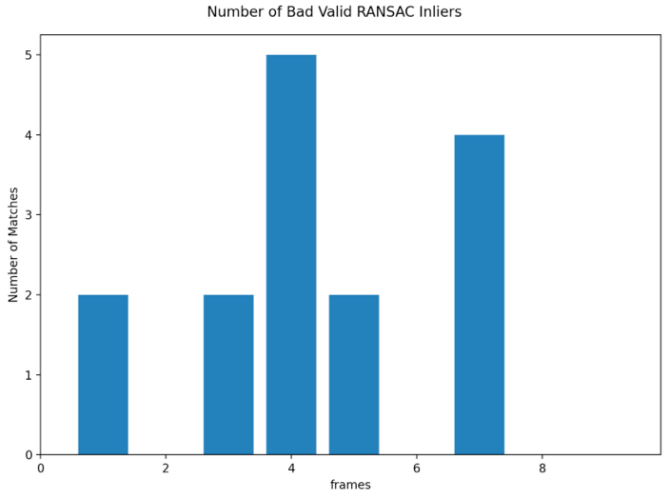

Figure 7.11: Bar graph of number of bad valid RANSAC inlier matches for each frame passed through to the PnP estimation algorithm

Figure 7.12 and Figure 7.13 plot the raw and filtered estimate errors. There are numerous frames with high levels of pose estimate errors. Filtering the errors outside of 3σ of the respective error range shows three frames with reasonable pose estimates: 2, 6, and 8. Figure 7.11 indicates these three frames have no bad valid RANSAC inliers. As expected, if a set of good matches are produced, the pose can be estimated more accurately.

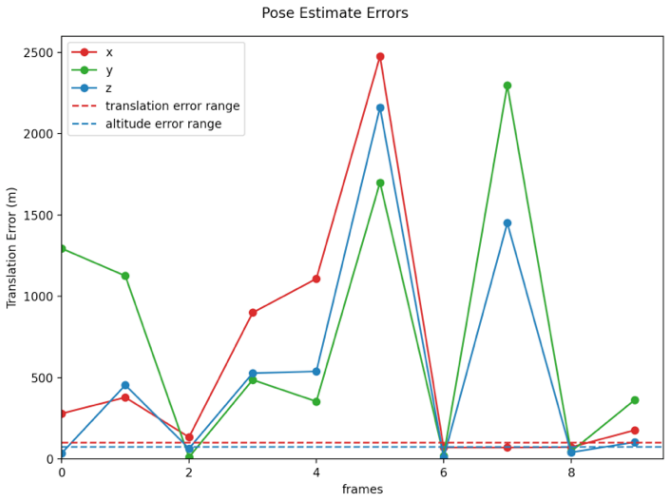

Figure 7.12: Raw fine estimation error in translation and altitude. The dashed lines represent the error range the algorithm was using to set the dynamic windows.

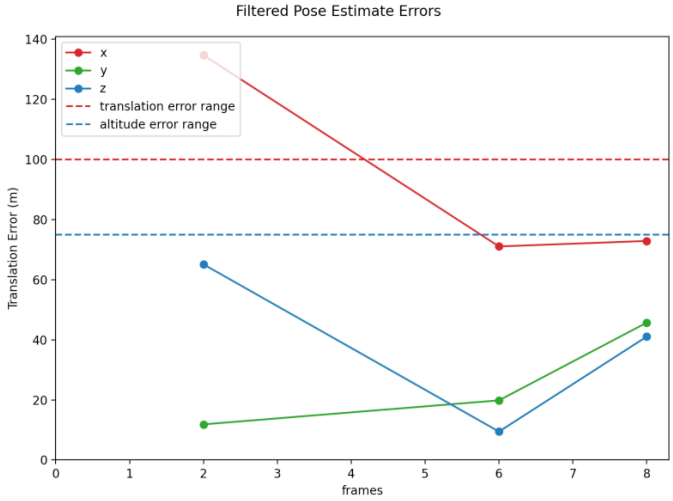

Figure 7.13: Pose estimates within 3σ of coarse error range.

Frame 6 shows the best pose estimate and contains a high number of good matches. Figure 7.14 displays the visualization of some of the correlation matches where the darker background is the reference map and the lighter foreground is the sensed image reprojected in the reference frame. All 100 of the correlation matches are not plotted to reduce clutter. On the right, the valid RANSAC inlier matches from the 100 corner case are reprojected into the sensed frame. This demonstrates the low reprojection error in cases where there is sufficiently high number of matches.

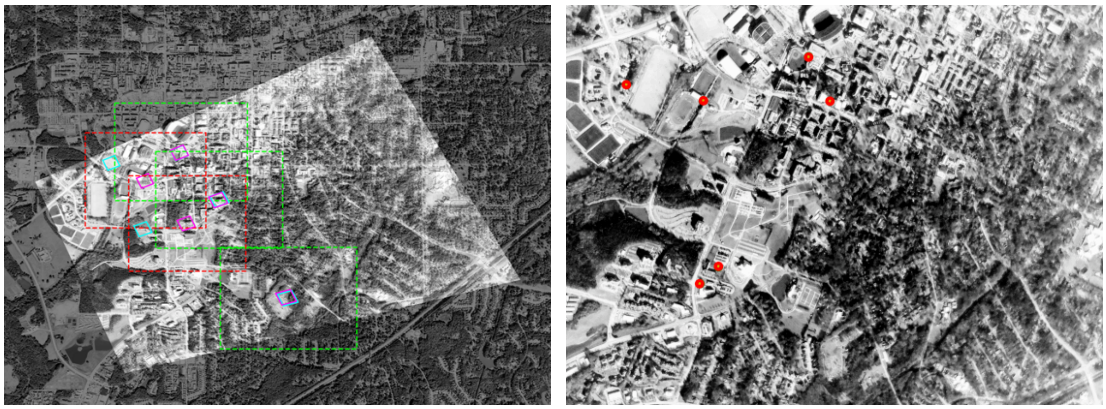

Figure 7.14: (Left) A few correlation matches between HAB sensed and Bing Aerial maps reference. (Right) Reprojected valid RANSAC inlier matches on sensed image.

7.3. Experimental Test #3: HAB sensed imagery to Google Reference Map over Uniontown, AL with SN02

This dataset tests the robustness of the algorithm with a high altitude balloon over a more rural area. See Table 7.3 for the information regarding the setup of this test. Due to the lack of consistent IMU orientation data like the previous experiment, the coarse and truth pose is manually determined with the point picker GUI and PnP described in Section 5.3.

Table 7.3: Parameters for Experimental Test #3

Parameter	Value
Sensed Imagery Source	HAB SN02
Reference Map Imagery Source	Google Satellite
Elevation Map Source	USGS 3DEP
Simulated Coarse Error σ (ENU)	(0m,0m,0m)
Simulated Coarse Error σ (Pitch, Roll, Yaw)	(0˚, 0˚, 0˚)
Translation and Altitude Error Range (ENU)	(100m,100m,75m)
Attitude Error Range (Pitch, Roll, Yaw)	(5˚, 5˚, 5˚)
Simulated Coarse Error σ (ENU)	(50m,50m,25m)
Number of Corners Selected	100
Template Size (Pixels)	125

The matching results in Figure 7.15 and Figure 7.16 are similar in performance to the previous experiment. The number of bad and valid RANSAC inliers and the pose estimate plots also show similar results to the previous experiment. This test demonstrates that the algorithm works similarly in rural environments at high altitudes as urban environments in the previous experiment.

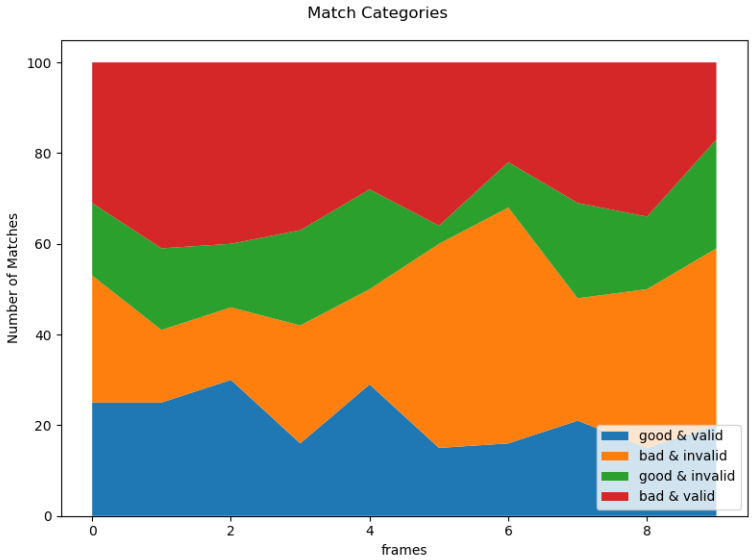

Figure 7.15: Stacked plot of matched category distribution for each frame. A good match is one that is within 50 pixels of the truth. A match is valid if its correlation passes the height and width criteria. In this dataset all the matches are good and invalid except for one near frame 95.

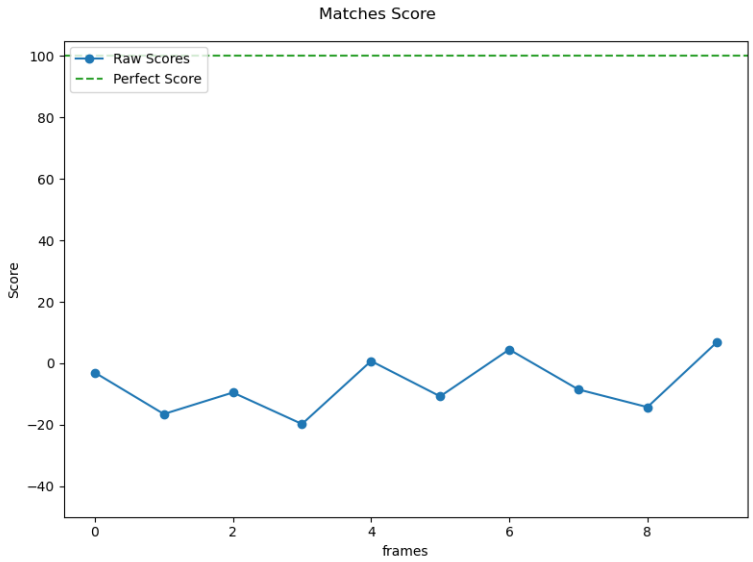

Figure 7.16: Match score as defined in Section 5.3.

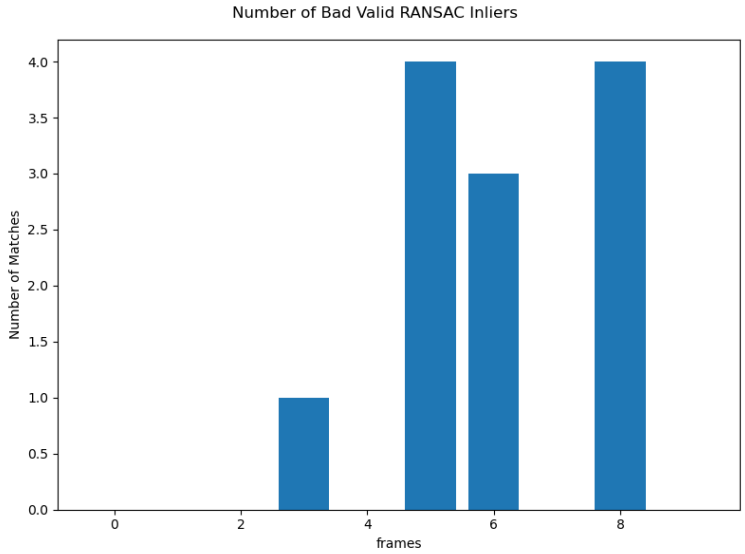

Figure 7.17: Bar graph of number of bad valid RANSAC inlier matches for each frame passed through to the PnP estimation algorithm

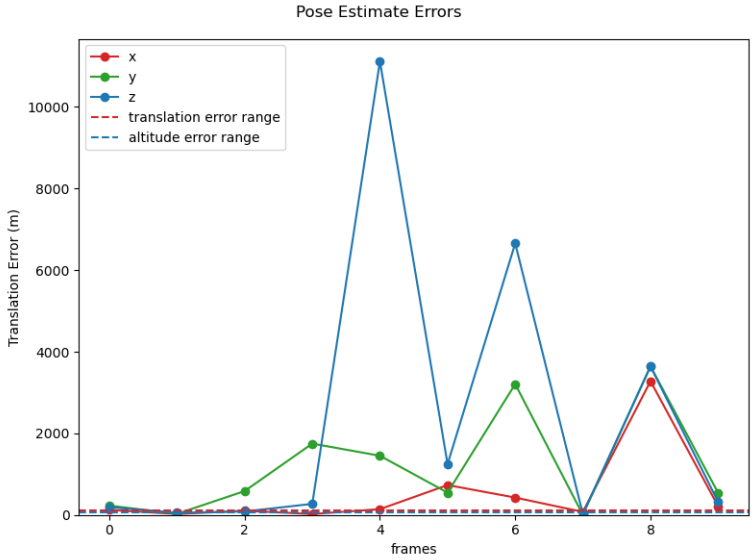

Figure 7.18: Raw fine estimation error in translation and altitude. The dashed lines represent the error range the algorithm was using to set the dynamic windows.

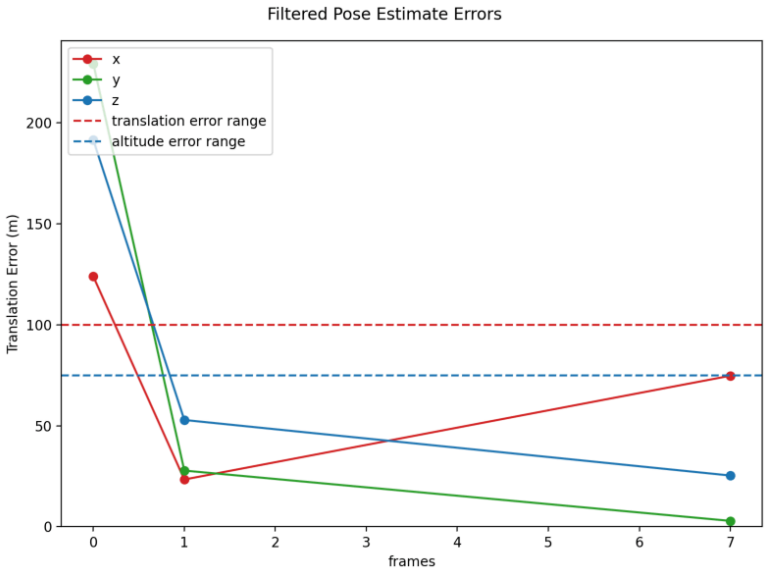

Figure 7.19: Pose estimates within 3σ of coarse error range.

7.4. Conclusion

As expected, the experimental results show degraded performance compared to the simulator. This is due to the higher uncertainty in truth pose and the drastic difference between simulated and experimental imagery. It is also possible the rolling shutter effect of the camera could cause imperceptible pixel shifting in sections of the image due to camera movement. These pixel shifts could increase the number of bad matches. Despite the lower pose estimate accuracy, these experiments demonstrate the algorithm is successful in certain cases where the terrain has distinct, consistent features between the sensed and reference images.

Chapter 8: Conclusion

8.1. Summary and Conclusion

The dynamic window template matching algorithm is demonstrated to estimate the pose within the error range on both simulated and experimental datasets. It performs well in simulation both in terms of matching and pose estimation. In the real imagery case, it struggles to find sufficiently high enough number of good matches in most of the frames, but it does succeed in some frames producing pose estimates within the error range.

8.2. Future Work

Further optimization and additional features are needed before this method is ready for real-time application. There are several avenues of continuing the research presented. These options are detailed in the following subsections.

8.2.1. Dynamic object removal

One of the major downfalls of image matching between satellite and sensed imagery is interference from new objects. Dynamic objects like vehicles in lower altitude imagery and clouds in higher altitude imagery can cause false matches as these objects are not present in the same location in the satellite maps. Deep learning and computer vision techniques could be used to classify aerial imagery and remove these objects by setting the pixel values to zero or fill in the gaps with predicted pixels. The matching algorithms ignore these zero pixels or would simply match the predicted pixels to the satellite maps. If the object removal algorithms performed well, the successful match rate would likely increase. Ammour et al. in [21] demonstrated an approach to detect cars in UAV imagery using a deep convolutional neural network. Chen et. al. in [22] successfully integrated a convolutional neural network to remove thick clouds.

8.2.2. Use Feature Matching within windows

The dynamic window template matching works well when the coarse pose is relatively close to the truth pose, but this is not true in all cases. Instead of using template matching, features within the template and within the windows could be matched to create sub-matches. RANSAC would eliminate the outlier sub-matches. Then the strongest sub-match would be selected as a match. This process can be performed on all window and template pairs to create a robust set of matches. A feature matching method would allow the use of storing all the reference features in a database like in [23] which is more space and computationally efficient than storing full image maps.

8.2.3. Performance enhancements and real-time hardware implementation

For this research, all of the navigation algorithms are run after the data is collected. This is not due to the setup of the algorithms, but the specific implementation. Further optimization of the performance is needed before they can run on real-time hardware. A robust and highly computationally efficient solution is to convert portions of the algorithms to an FPGA coprocessor or to use GPU resources. Additionally, the creation of a map server could be used to optimize the size of the reference map used for analysis. This is a major source of the computation cost for the algorithms discussed. The capability of using a reference map for any location within a larger region of maps saved onto a hard drive would help make it possible to use the algorithms on an aerial system in real time.

8.2.4. Simulator Improvements

The simulator currently generates images assuming the Earth is flat. The ray projection technique is capable of accommodating terrains with a large variance in elevation but would need further derivation. PnP does consider the elevation difference of each of the points it selects.

Improving the simulator would allow testing and optimization in mountainous terrains. Additionally, the simulator is the most computationally expensive part of the solution proposed in this research thus performance enhancements could increase the ease of use.

8.2.5. Kalman filter or other estimation techniques

On the sensor fusion and filtering aspect of the problem, implementing a Kalman filter or particle filter could improve the pose estimates and matching. In [24] and [25] respectively, the Kalman filter and particle filter are implemented in the image matching process. These methods could provide some improvements over RANSAC by fitting a dynamic model to the matching instead of minimizing reprojection error with brute force. Using sensors like an IMU and barometric altimeter could help constrain the pose and form a closed-loop solution. Additionally, terrain relative navigation can be coupled with algorithms like visual inertial odometry or optical flow to fill in the gaps between pose estimates. Solutions from other types of non-visual imaging sensors like multi-channel LIDAR and RADAR systems could be fused with visual imagery using algorithms like the Kalman Filter.

Bibliography

[1] Yang Cheng, A. Johnson, and L. Matthies, "MER-DIMES: a planetary landing application of computer vision," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Jun. 2005, vol. 1, pp. 806–813 vol. 1, doi: 10.1109/CVPR.2005.222.

[2] G. Conte and P. Doherty, "An Integrated UAV Navigation System Based on Aerial Image Matching," in *2008 IEEE Aerospace Conference*, Mar. 2008, pp. 1–10, doi: 10.1109/AERO.2008.4526556.

[3] M. Shan, F. Wang, F. Lin, Z. Gao, Y. Z. Tang, and B. M. Chen, "Google map aided visual navigation for UAVs in GPS-denied environment," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2015, pp. 114–119, doi: 10.1109/ROBIO.2015.7418753.

[4] F. Lindsten, J. Callmer, H. Ohlsson, D. Törnqvist, T. B. Schön, and F. Gustafsson, "Geo-referencing for UAV navigation using environmental classification," in *2010 IEEE International Conference on Robotics and Automation*, Anchorage, AK, May 2010, pp. 1420–1425, doi: 10.1109/ROBOT.2010.5509424.

[5] A. Nassar, K. Amer, R. ElHakim, and M. ElHelw, "A Deep CNN-Based Framework For Enhanced Aerial Imagery Registration with Applications to UAV Geolocalization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2018, pp. 1594–159410, doi: 10.1109/CVPRW.2018.00201.

[6] E. M. Moreira, O. A. M. Camargo, J. C. Duarte, and P. F. F. Rosa, "Scene Matching in GPS Denied Environments: A Comparison of Methods for Orthophoto Registration," in *2019*

IEEE International Conference on Mechatronics (ICM), Mar. 2019, vol. 1, pp. 205–210, doi: 10.1109/ICMECH.2019.8722872.

[7] "What Is Camera Calibration? - MATLAB & Simulink." https://www.mathworks.com/help/vision/ug/camera-calibration.html (accessed Jun. 13, 2020).

[8] "Camera Calibration Toolbox for Matlab." http://www.vision.caltech.edu/bouguetj/calib_doc/ (accessed Jun. 14, 2020).

[9] "OpenCV: Camera calibration With OpenCV." https://docs.opencv.org/3.4/d4/d94/tutorial_camera_calibration.html (accessed Jun. 14, 2020).

[10] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[11] Jianbo Shi and Tomasi, "Good features to track," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, Seattle, WA, USA, 1994, pp. 593–600, doi: 10.1109/CVPR.1994.323794.

[12] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981, doi: 10.1145/358669.358692.

[13] G. Terzakis and M. Lourakis, "A Consistently Fast and Globally Optimal Solution to the Perspective-n-Point Problem," in *Computer Vision – ECCV 2020*, vol. 12346, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 478–494.

[14] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johnson, A. Ansar, and L. Matthies, "Vision-Aided Inertial Navigation for Spacecraft Entry, Descent, and Landing," *IEEE Trans. Robot.*, vol. 25, no. 2, pp. 264–280, Apr. 2009, doi: 10.1109/TRO.2009.2012342.

[15] "Department of Defense World Geodetic System 1984," National Geospatial Agency, National Geospatial-Intelligence Agency (NGA), NIMA TR8350.2, Jan. 2000. [Online]. Available: https://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf.

[16] M. Mukul, V. Srivastava, S. Jade, and M. Mukul, "Uncertainties in the Shuttle Radar Topography Mission (SRTM) Heights: Insights from the Indian Himalaya and Peninsula," *Sci Rep*, vol. 7, Feb. 2017, doi: 10.1038/srep41672.

[17] Mide Technology Corporation, "Air Pressure at Altitude Calculator." https://www.mide.com/air-pressure-at-altitude-calculator (accessed Feb. 27, 2021).

[18] Srikanth A, L. Krishnamurthy, L. P. Prathyusha, and V. Naidu, "Synthetic aerial image generation for miniature aerial system," in *2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15)*, Dec. 2015, pp. 1–6, doi: 10.1109/ITACT.2015.7492685.

[19] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, and C. N. Taylor, "Vision-based Target Geo-location using a Fixed-wing Miniature Air Vehicle," *J Intell Robot Syst*, vol. 47, no. 4, pp. 361–382, Nov. 2006, doi: 10.1007/s10846-006-9088-7.

[20] *SMALL UNMANNED AIRCRAFT SYSTEMS*. .

[21] N. Ammour, H. Alhichri, Y. Bazi, B. Benjdira, N. Alajlan, and M. Zuair, "Deep Learning Approach for Car Detection in UAV Imagery," *Remote Sensing*, vol. 9, pp. 1–15, Mar. 2017, doi: 10.3390/rs9040312.

[22] Y. Chen, L. Tang, X. Yang, R. Fan, M. Bilal, and Q. Li, “Thick Clouds Removal From Multitemporal ZY-3 Satellite Images Using Deep Learning,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 143–153, 2020, doi: 10.1109/JSTARS.2019.2954130.

[23] D. T. Venable, “Improving Real-World Performance of Vision Aided Navigation in a Flight Environment,” Air Force Institute of Technology.

[24] N. Q. Ann, D. Pebrianti, Z. Ibrahim, L. Bayuaji, and M. F. M. Jusoh, “Image Template Matching Based on Simulated Kalman Filter (SKF) Algorithm,” vol. 10, no. 2, p. 5.

[25] E. Arce-Santana, D. U. Campos Delgado, and A. Alba, “Image Registration Guided by Particle Filter,” Dec. 2009, vol. 5875, pp. 554–563, doi: 10.1007/978-3-642-10331-5_52.

Appendix A: Details of High-Altitude Balloon SN01 Design, Launch, and Recovery

A.1. Electrical System

The payload system (Figure A.1) uses a Raspberry Pi 4B to record the data to a microSD card. One of the IMUs and the barometer use I^2C to communicate with the Pi. The other IMU and the GPS module communicate with the Pi via UART. The downward facing camera uses the Pi's CSI cable. A Pi zero is operated as a separate system to capture data from the side facing camera to remove some of the load off of the main processor.

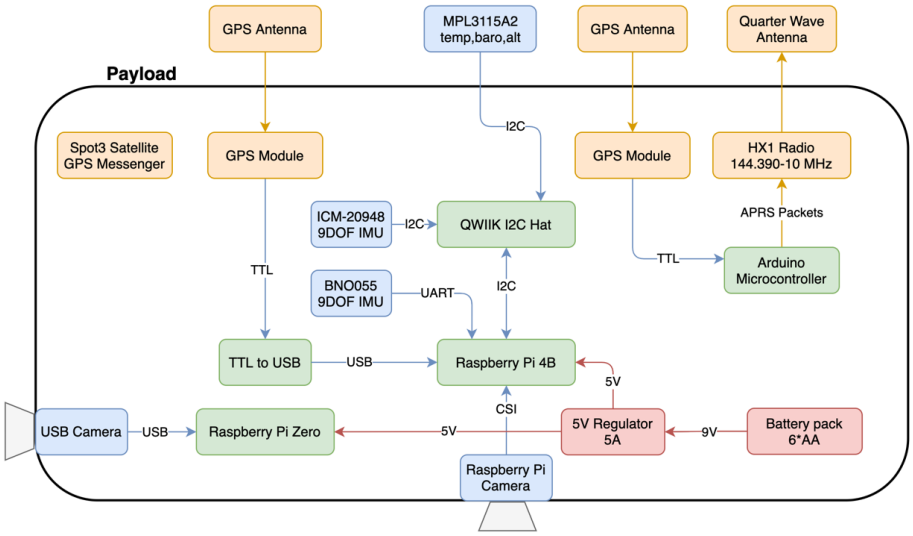

Figure A.1: System Diagram for SN01

The primary tracking system uses an Arduino Nano connected to a 144.39MHz transmitter and a secondary GPS module as shown in the schematic in Figure A.2. It transmits automatic packet reporting system (APRS) messages containing GPS coordinate information to a network of Digipeaters and iGates. A Digipeater is a HAM radio operated radio system the repeats APRS packets. An iGate is a HAM radio operated system that receives APRS packets and uploads them to the internet. This system requires the use of a Federal Communications Commission (FCC) Amateur Technician class license. The antenna used is a 2m quarter wave monopole antenna with

the radiating element pointed upward and the four radials pointed downward. This allows for an omnidirectional radiation pattern without obstructing the camera imagery. The secondary tracker is a Spot3 satellite GPS messaging service device that does not require an FCC license but a monthly subscription fee. It communicates with the Globalstar satellite network to relay the data back to the user.

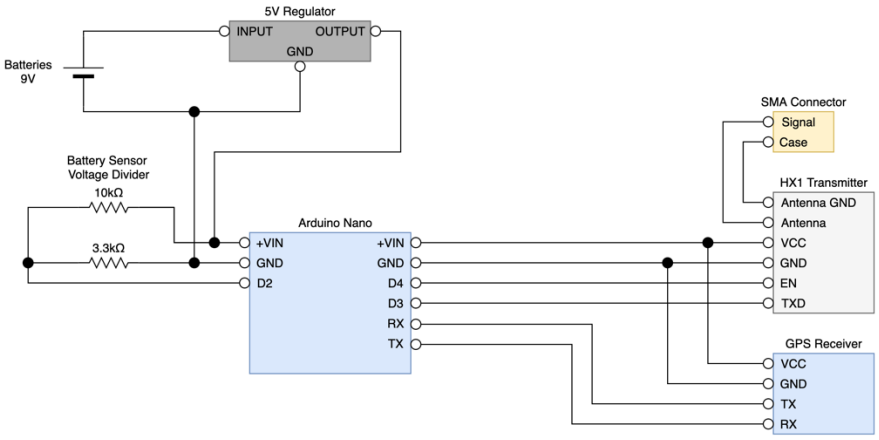

Figure A.2: Transmitter board schematic

A.2. Mechanical System

The payload is constructed with home insulation polystyrene foam glued together and covered with reflective and packing tape for increased visibility and to withstand moisture upon landing (Figure A.3). Insulation foam was chosen due to its low weight and temperature insulating properties to allow the payload electronics to withstand large temperature variations. The electronics were mounted with 3D printed parts printed with eSun PLA Pro filament. The Spot GPS tracking system is mounted on a custom designed passive gyro to ensure its antenna will point towards the sky upon landing.

Figure A.3: (Left) Exterior of polystyrene payload container with quarter wave monopole antenna. (Right) Interior of payload box showing electronics.

A.3. Software System

The payload runs on Ubuntu Mate 20.04LTS with ROS Noetic. ROS is a framework that helps with the interface of numerous sensors. It can record time synced data in a file called a ROS bag. The data from the bag file is capable of being extracted iteratively or in real time after the flight. The APRS tracking system is based on SparkFun's Trackuino library. It transmits an APRS packet once a minute when the GPS module has a satellite fix. The software is publicly available at: https://github.com/GAVLab/ros-hab-dcs.

A.4. Balloon System

A 600g latex weather balloon is inflated with industrial grade helium of 99.996% purity. The required amount of helium is based on the payload weight and desired positive lift as determined using this calculator: http://tools.highaltitudescience.com/ As the balloon is inflated, the positive lift is measured with a pull scale as shown in Figure A.4.

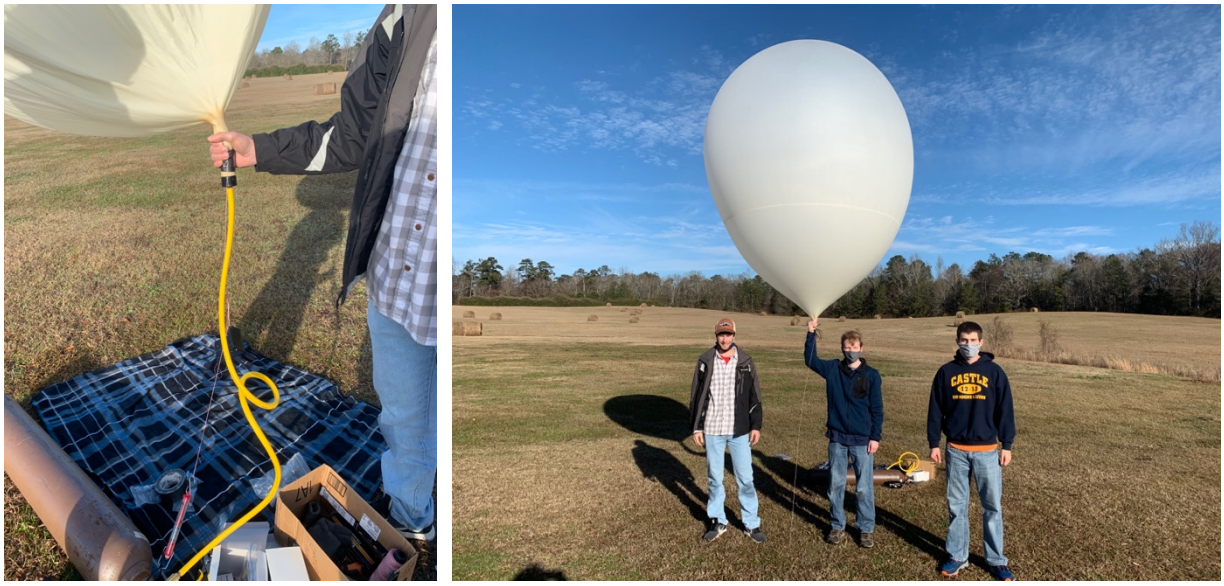

Figure A.4: (Left) Helium inflation process of balloon with pull scale. (Right) Fully inflated balloon

A.5. Telemetry and Tracking

High altitude balloon flights can be predicted up to 5 days in advance by using online tools like: https://predict.habhub.org/. In the first flight, the payload travelled from the west side of Auburn to just east of Macon, GA as shown in the prediction in Figure A.5. The payload landed just under 20 mi to the east of the predicted landing site. The predicted highest altitude was 90,000ft but the highest measured altitude according to the barometer was approximately 80,000ft. This difference is due to inaccuracies in the filling process.

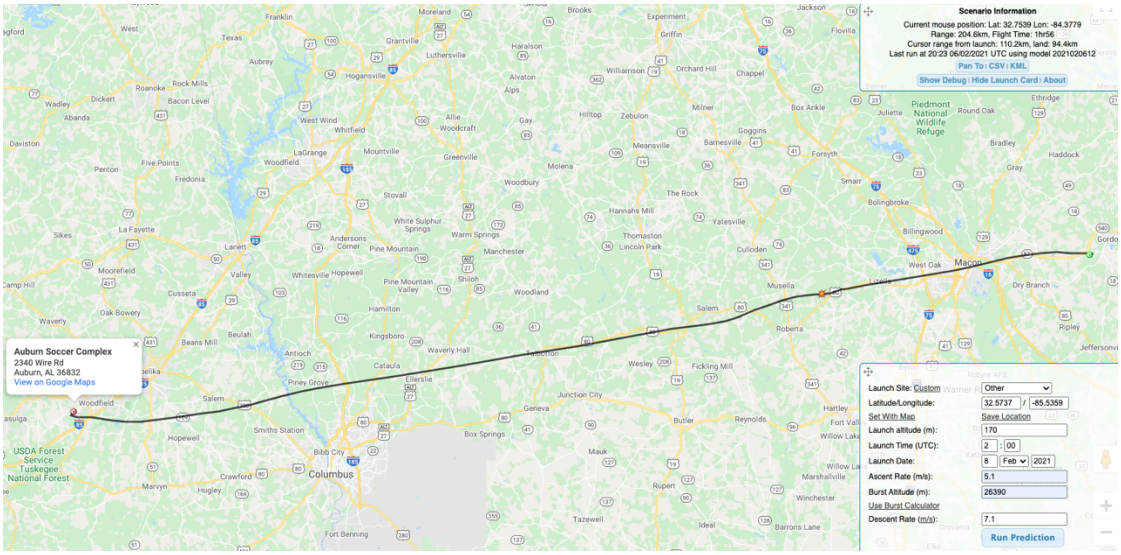

Figure A.5: Predicted flight path from east of Auburn, AL to west of Macon, GA

Many of the APRS packets were received by the vehicle mounted antenna connected to a Windows laptop running the APRS decoding software, Direwolf. No GPS packets were received above 12km or 40,000ft due to the maximum height software limit of the chosen GPS. In addition, no GPS packets were received below 3000ft due to the lack of nearby Digipeaters and iGates. The precise landing location was determined with the Spot3 tracker. It provided updates periodically during the three weeks after it landed.

A.6. Recovery

Recovery of this payload was the most difficult aspect of the project. The landing location was approximately three miles away from the nearest public road. The terrain was low-lying forested swamp land (Figure A.6). Hiking out to it was exceedingly difficult due to the thick vegetation and water. After contacting the property owner, the payload was recovered when the water from recent rains subsided.


Figure A.6: (Left) Payload landing site with string and parachute suspended in tree (Right) Recovered payload

The payload condition was better than expected as all sensors were intact, and the box only suffered minor damage. The interior of the box was coated with moisture, but no significant amount of water remained in the box. The exterior showed signs of water seeping between the packing tape. The antennas were bent due to impact and the recovery process. The parachute, radar reflector, and the shards of the balloon were unable to be recovered.

A.7. Collected Data

Despite the interior moisture, both microSD cards were still recoverable. The main Pi filled up all 128GB of SD storage preventing future boots. By using a Linux VM, the ROS bag files were extracted. From analysis of the data, the SD card filled up three-quarters through the flight. Figure A.7 shows the resulting altitude of the payload from the GPS and barometer. This GPS module also cut out but at the slightly higher altitude of 16,743.60m or 54,933.07ft. Using the process described in Section 3.2.1, the highest barometric altitude record is 24,508.61m or 80,408.83ft. This is below the estimated altitude, but high enough to collect interesting data. The GPS module also reported the velocity of the payload. The maximum speed recorded is 65.38m/s

or 146.25mph. The map in Figure A.8 shows a similar path to the predicted path excluding the missing last quarter of the flight.

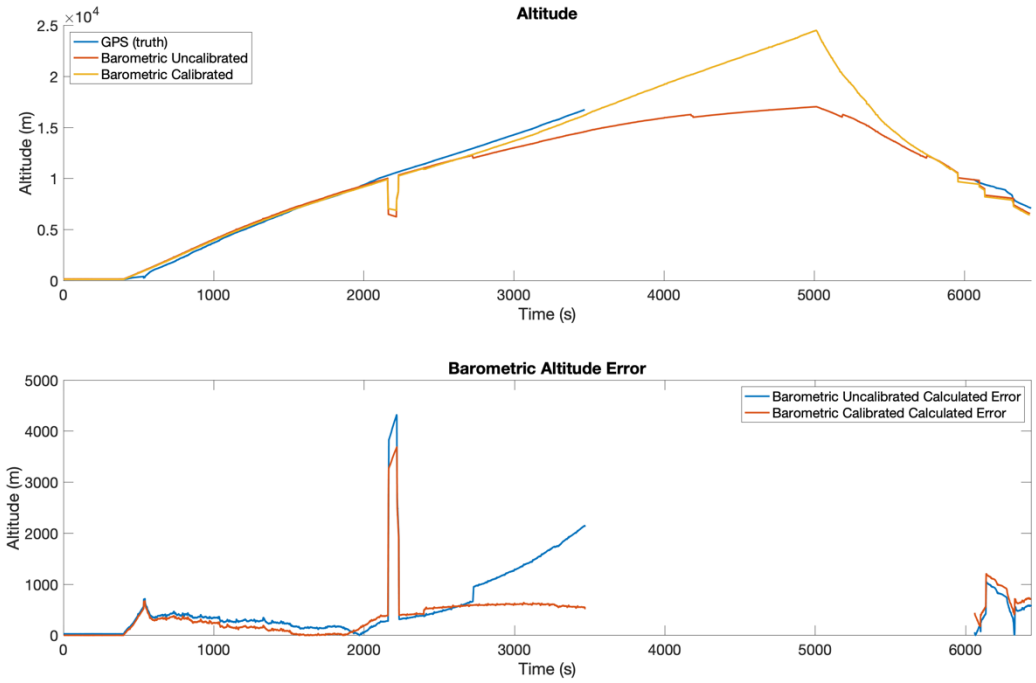

Figure A.7: Comparison of barometric altitude and GPS altitude. GPS cuts out during higher altitude portion of flight.

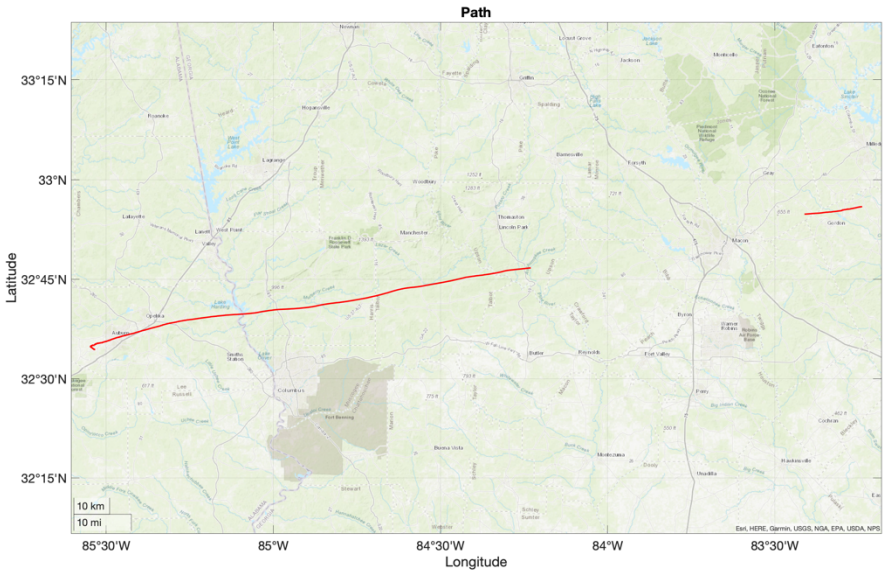

Figure A.8: Map of path of payload

Another interesting aspect of the flight are the extreme temperatures experienced. Figure A.9 plots the internal, external, and main CPU temperature during the flight. The external temperature dropped down to -42.75˚C or -44.95˚F, but the internal temperature only dropped to -12.00˚C or 10.40˚F. This is due to the chemical handwarmer placed in the box, the heat produced from the Pi, and the insulation foam used for the sides of the box.

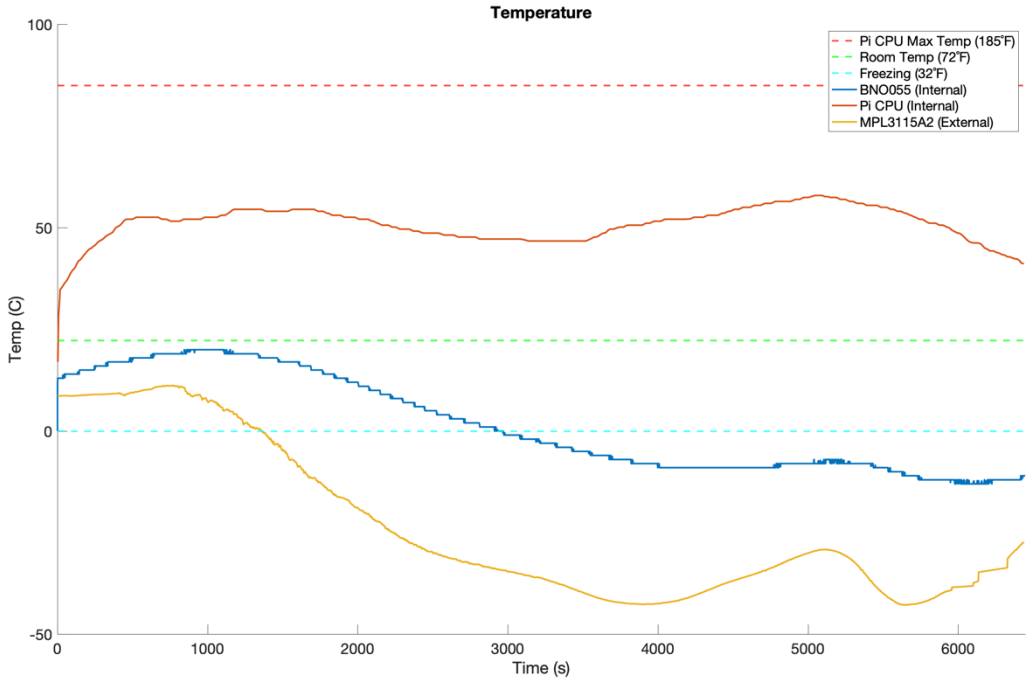

Figure A.9: Temperature of the payload during flight

The orientation is filtered and plotted in Figure A.10. For most of the flight, the pitch and roll remain fairly close to zero with some minor fluctuations due to swinging. The yaw tends to vary widely as reflected in the imagery where the payload spins from until tension builds in the string and starts spinning the other way. The balloon pops at about 5000 seconds showing a significantly higher amount of fluctuation in pitch and roll during descent.

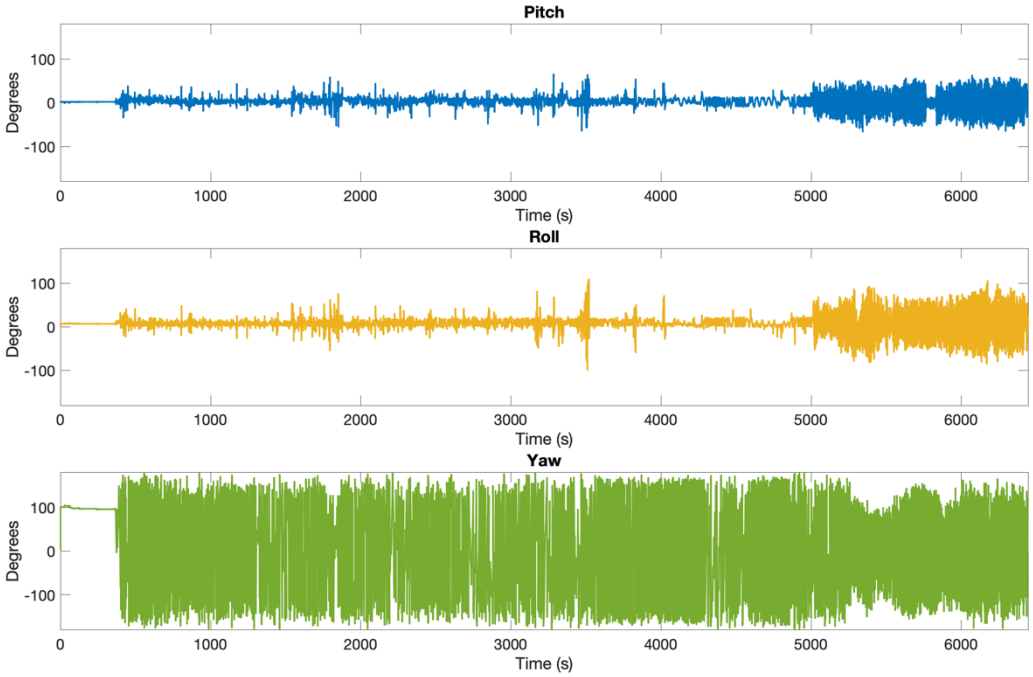

Figure A.10: Euler angle orientation captured by BNO055 IMU.

A.8. Imagery

The payload collected data from two cameras: a side-mounted and a downward-facing camera. The side-mounted camera collected data for the sole purpose of understanding the context of where the balloon is with respect to the surface. For some of the higher altitude photos, the "blackness" of space is visible. It also provided insight to the motion of the payload during ascent and descent as the camera pans, tilts, and rolls. Some of the best photos are shown in Figure A.11.

Figure A.11: Highlights from the side-mounted camera: (Top-left and Top-right) High altitude views (Bottom-left) Aerial view of Auburn GAVLAB and NCAT test track and surrounding areas. (Bottom-right) Auburn, AL

The downward-facing camera is used for the localization algorithms. During the entire flight, it very rarely captured the horizon. Capturing color imagery demonstrates how the terrain has a bluer hue at higher altitudes due to light diffracting in the atmosphere. The effect on monochrome imagery is small but could contribute to some loss in clarity. Figure A.12 highlights some of the best examples from the downward-facing camera.

Figure A.12: Highlights from the downward-facing camera: (Top-left). Macon, GA (Top-right) Lake Harding on AL-GA state line. (Bottom-left) Rural land west of Auburn, AL. (Bottom-right) Auburn University main campus

Appendix B: Details of High-Altitude Balloon SN02 Design, Launch, and Recovery

B.1. Improvements to Design

From the lessons learned from the first balloon flight, various improvements were made. A self-contained APRS radio tracking system called Stratotrack replaced the quarter wave monopole antenna system on SN01. It used a dipole antenna soldered to a small circuit board that is attached to the main parachute. This helped reduce the box size dramatically. Another major improvement is the use of a 256GB SD card, doubling the size of the too small 128GB SD card of the first flight. This helped ensure data could be collected during the entire flight.

A different set of sensors were used for the second payload as depicted in Figure B.1. The GPS module selected for this flight, u-blox MAX-M8Q, is rated for 50,000m or 164041ft which exceeds the target balloon altitude. The selected IMU module, BNO080, is an improved version of the BNO055 IMU from the previous flight. An Arduino Nano 33 BLE Sense is added that contains several additional sensors like temperature (HTS221), humidity (HTS221), and pressure (LPS22HB). A battery monitor, INA219, records the voltage and current of the battery.

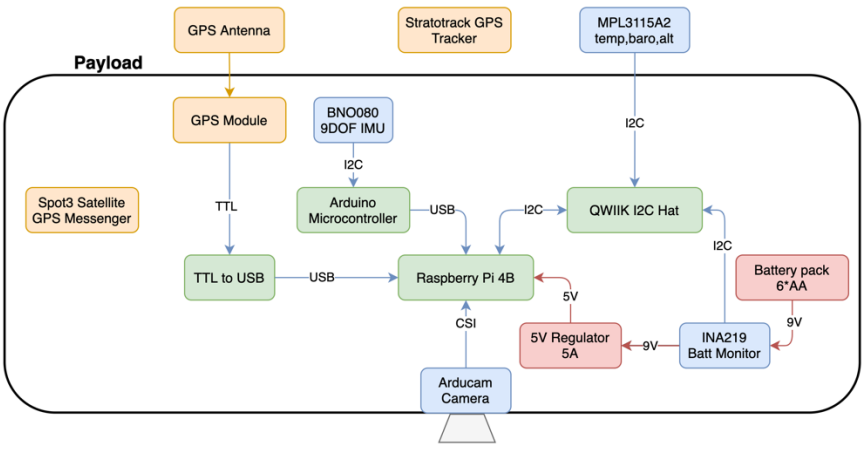

Figure B.1: System Diagram for SN02

Figure B.2: (Left) Exterior of payload shortly before flight. (Right) Interior of payload.

B.2. Flight and Collected Data

The payload was launched in a small town just west of Selma, AL, called Uniontown, AL. It flew south-east for 2 hours until it landed between Montgomery, AL and Troy, AL as shown in Figure B.2. It reached a height of 21,286m or 69,836ft according to the GPS module in Figure B.3. The barometric altitude diverged from GPS by about 1000m in the stratosphere once calibrated but maintained fairly close during the lower altitude portions of the flight. Figure B.4 shows the pressure changes. The internal and external pressure of the payload box were recorded to see if any effect of sensor placement could be observed. The pressure difference is noticeable but negligible in terms of estimating altitude. Figure B.5 shows the orientation Euler angles and Figure B.6 shows the recorded temperature during the flight. Both of these plots show similar results from the first flight.

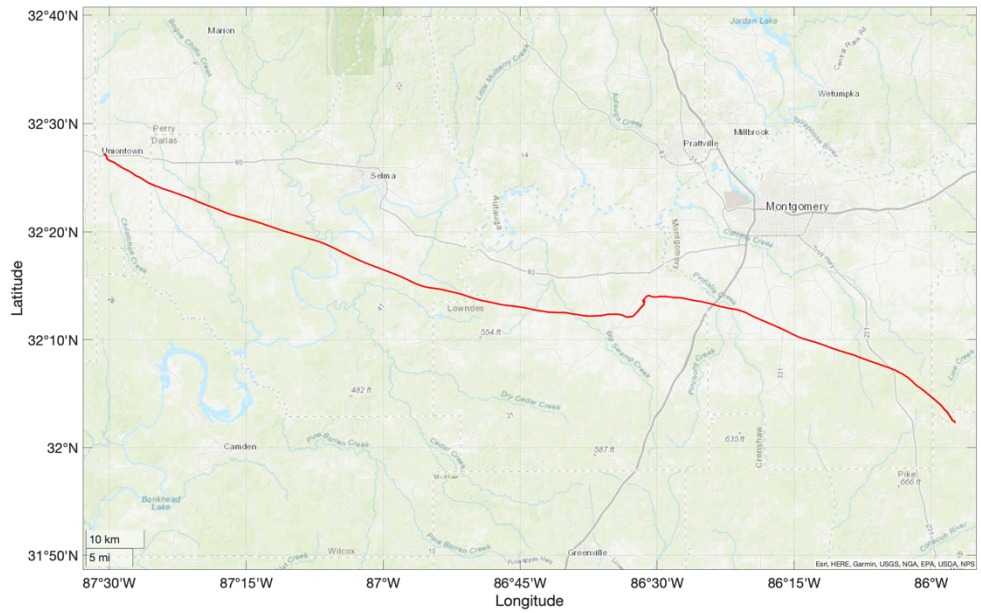

Figure B.2: Map of path of payload

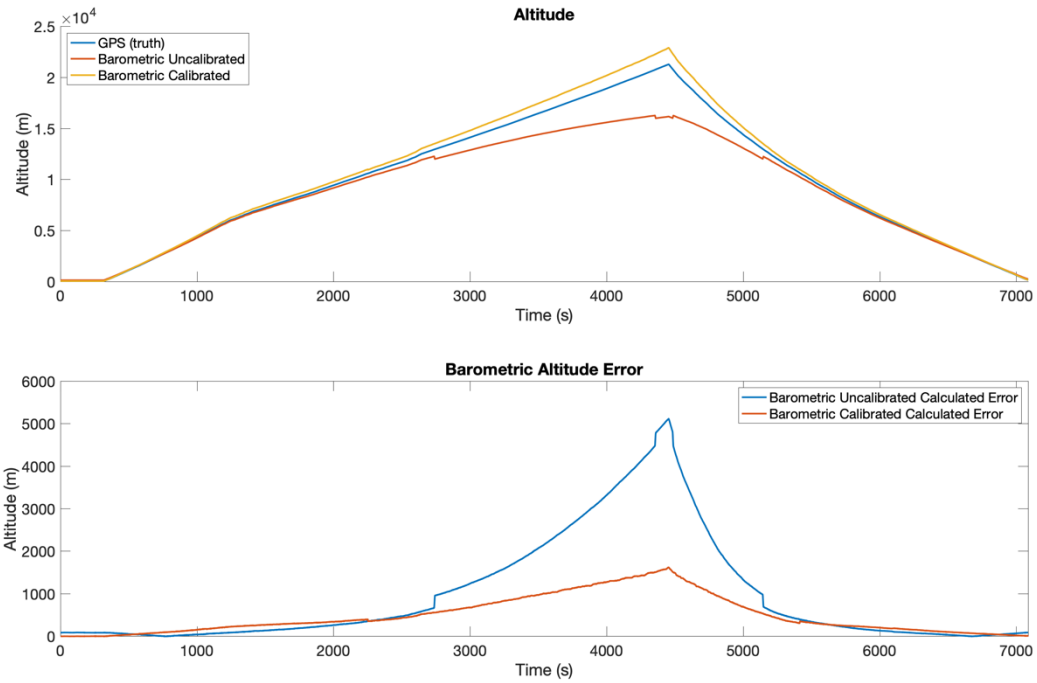

Figure B.3: Comparison of barometric altitude and GPS altitude.

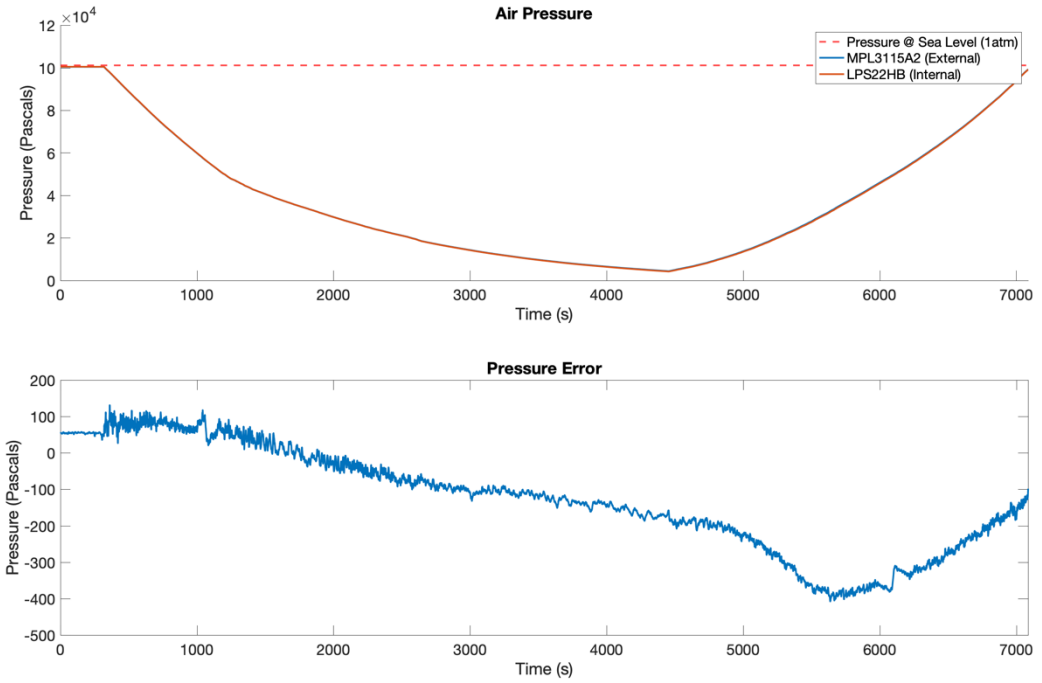

Figure B.4: (Top) Pressure of interior and exterior of payload compared to sea level. (Bottom) Difference in pressure between internal and external barometer.

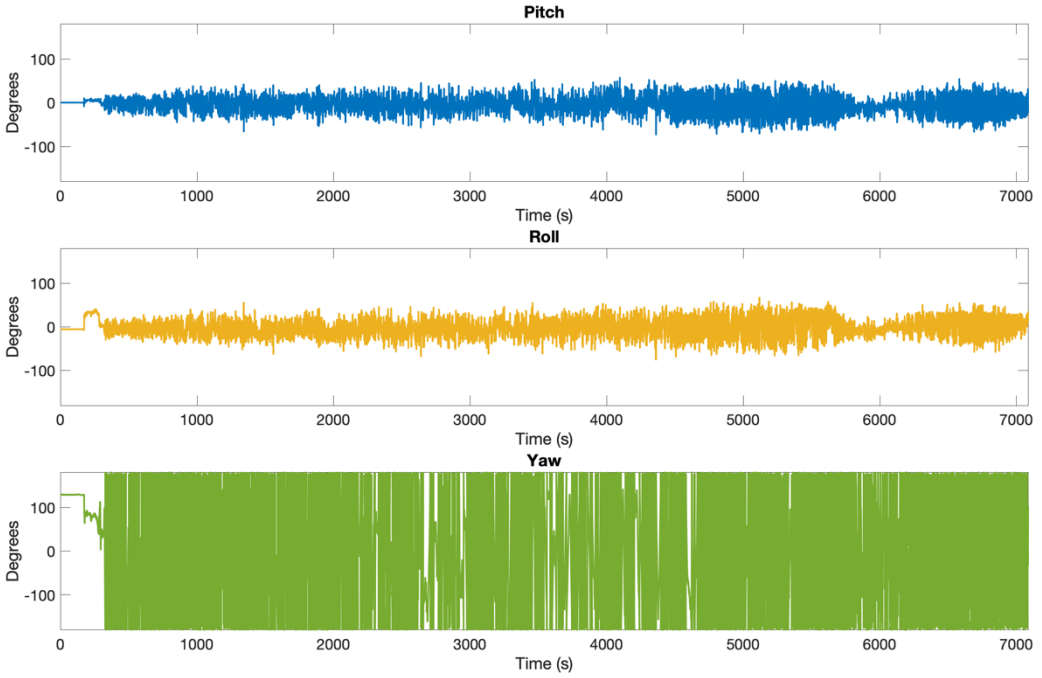

Figure B.5: Euler angle orientation captured by BNO080 IMU.

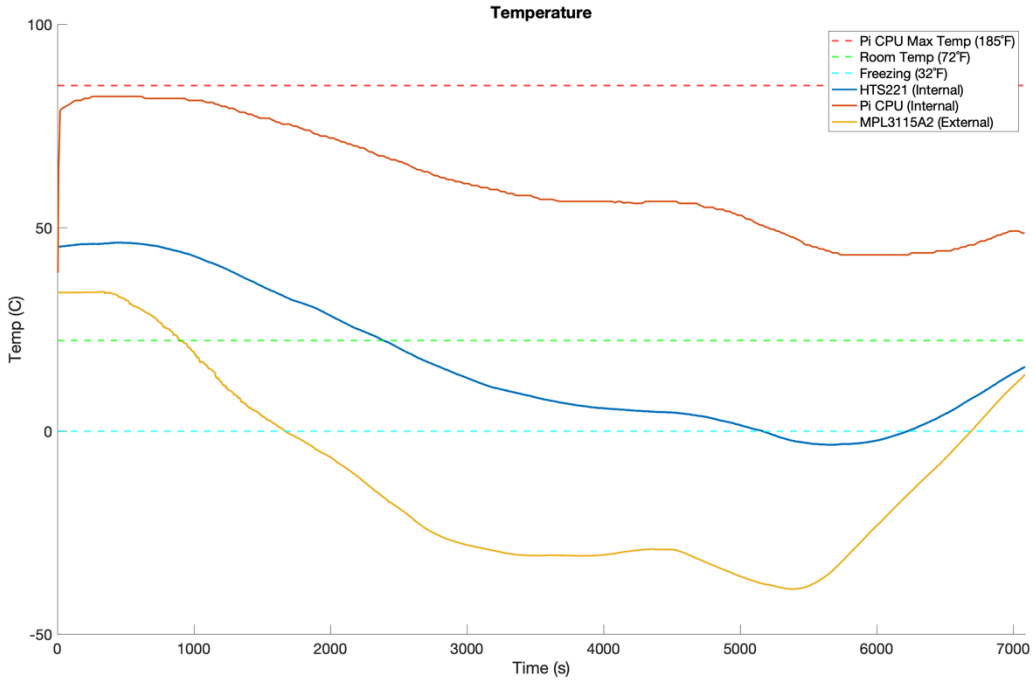

Figure B.6: Temperature of the payload during flight