Enhanced Image-aided Navigation Algorithm with Automatic Calibration and Affine Distortion Prediction

THESIS

Juan D. Jurado, Captain, USAF

AFIT/GE/ENG/12-23

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/12-23

Enhanced Image-aided Navigation Algorithm with Automatic Calibration and Affine Distortion Prediction

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Juan D. Jurado, B.S.E.E.

Captain, USAF

March 2012

AFIT/GE/ENG/12-23

Enhanced Image-aided Navigation Algorithm with Automatic Calibration and Affine Distortion Prediction

Juan D. Jurado, B.S.E.E.
Captain, USAF

Approved:

Maj K. Fisher, PhD (Chairman)	1 Mar 2012 date
Lt Col M. Stepaniak, PhD (Member)	1 Mar 2012 date
Dr. J. Raquet (Member)	1 MAR 2012 date

AFIT/GE/ENG/12-23

Abstract

The Air Force Institute of Technology's Advanced Navigation Technology center has invested significant research time and effort into alternative precision navigation methods in an effort to counteract the increasing dependency on Global Positioning System for precision navigation. Such alternative navigation methods have become particularly useful in environments where the required direct line of sight to the satellite constellation is not available or when the enemy is purposely denying access to the GPS signal. The use of visual sensors and feature tracking has since emerged as a valuable and feasible precision navigation alternative which, when coupled with inertial navigation sensors can reduce navigation estimation errors by approximately two orders of magnitude [22]. Although the basic mathematics and algorithms have been thoroughly documented, image-aided navigation is still in its early stages. This research aims to improve two key steps within the image-aided navigation process: camera calibration and landmark tracking. The camera calibration step is improved by automating the point correspondence calculation within the standard camera calibration algorithm, thereby reducing the required time for calibration while maintaining the output model accuracy. The landmark tracking step is improved by digitally simulating affine distortions on input images in order to calculate more accurate feature descriptors for improved matching through large changes in relative viewpoint. These techniques are experimentally demonstrated in an outdoor environment with a consumer-grade inertial sensor and three imaging sensors, one of which is orthogonal to the others. Using a tactical-grade inertial sensor coupled with GPS position data as the truth source, the improved image-aided navigation algorithm is shown to reduce navigation errors by 24% in position, 16% in velocity, and 35% in attitude compared to the standard two-camera image-aided navigation setup.

Acknowledgements

The list of individuals that have helped me throughout my research is long, but there are a few key people who I'd like to highlight. First, I'd like to thank my advisor who has been a role model, patient mentor and an amazing sounding board for ideas. Next, I'd like to thank the members of the ANT Center staff who provided priceless support, suggestions and expertise. Finally, but most importantly, I'd like to thank my wife, who has not only enabled me to succeed but inspired me to excel while tolerating my increased work hours away from home. Thank you all for your invaluable help, I could have not done this without you.

Juan D. Jurado

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Abbreviations

Abbreviation		Page
GPS	Global Positioning System	1
ANT	Advanced Navigation Technology	2
NED	North, East and down	8
DCM	Direction Cosine Matrix	12
IMU	Inertial Measurement Unit	15
INS	Inertial Navigation System	16
MEMS	Micro Eletro-mechanical Systems	16
RGB	Red Green Blue	16
SIFT	Scale Invariant Feature Transform	21
DOG	Difference of Gaussians	22
NNDR	Nearest Neighbor Distance Ratio	26
RANSAC	Random Sample Consensus	26
ASIFT	Affine Scale Invariant Feature Transform	27
MMSE	Minimum Mean Squared Error	36
EKF	Extended Kalman Filter	37
ADP	Affine Distortion Prediction	47
IAEKF	Image-aided Extended Kalman Filter	48
RSS	Root Sum Squared	80
PVA	Position, Velocity, and Attitude	80

Enhanced Image-aided Navigation Algorithm with Automatic Calibration and Affine Distortion Prediction

I. Introduction

The success of the United States Air Force mission is dependent on the availability of precision navigation information. This need is currently fulfilled by the combination of inertial sensors and position information from the Global Positioning System (GPS). Without GPS, inertial sensors alone can only provide a reliable solution for a short term before inertial drift-induced uncertainty exceeds acceptable limits. The absence of an alternate inertial error-constraining technology has created a dependency on GPS, which is vulnerable to disruption from urban line-of-sight occlusion or intentional enemy jamming. The Chief of Staff of the Air Force has addressed this issue by stating, "It seems critical to me that the Joint Force should reduce its dependence on GPS-aided precision navigation and timing, allowing it to ultimately become less vulnerable, yet equally precise, and more resilient" [17].

1.1 Current Technology

One of the major research thrusts emerging from the need to decrease dependency on GPS is image-aided navigation. The concept of image-aided navigation involves augmenting inertial sensors with imaging sensors in order to track visual landmarks. Landmark tracking coupled with inertial sensor measurements has been shown to produce stable navigation solutions that are nearly two orders of magnitude more accurate than inertial sensors alone [22]. This research focuses on improving two aspects within image-aided navigation: camera calibration and landmark tracking.

1.1.1 Camera Calibration. The current tool of choice for camera calibration is the Camera Calibration Toolbox from Caltech [1]. The algorithms used inside

the tool are heavily based on Dr. Zhang's camera calibration work [24] and further explained in the next chapter. Currently, a standard two-camera calibration procedure is as follows:

1. First, the cameras are rigidly mounted to a camera bar.
2. Next, a planar calibration surface such as the one illustrated in Figure 1.1 is held still while both cameras capture an image.
3. The planar surface is then moved into a different pose and the cameras capture another image.
4. This process is repeated until a sufficient number poses have been captured (usually around 10).
5. Next, the user must manually step through each image for each camera and click on the outer four corners of calibration surface as shown in Figure 1.2.
6. If the calibration surface is not fully visible within an image, such image must be discarded.
7. The calibration toolbox uses the corner mapping from each image to calculate intrinsic and extrinsic parameters for each camera.
8. Lastly, the calibration toolbox produces the relative translation and rotation between the two cameras and calibration is complete.

1.1.2 Landmark Tracking. The current image-aided navigation algorithm employed at the Advanced Navigation Technology (ANT) center uses stochastic-constrained search areas derived from inertial measurements to find and track features from one discrete time to the next. A successful feature match depends on the Euclidean distance between the 128-vector descriptor of a feature and its candidate match. Figure 1.3 illustrates successful (green) and unsuccessful matches (yellow) over consecutive time steps. Common causes for unsuccessful matches include features leaving the image plane between discrete time steps and large affine distortions arising from either vehicle movement or camera selection.

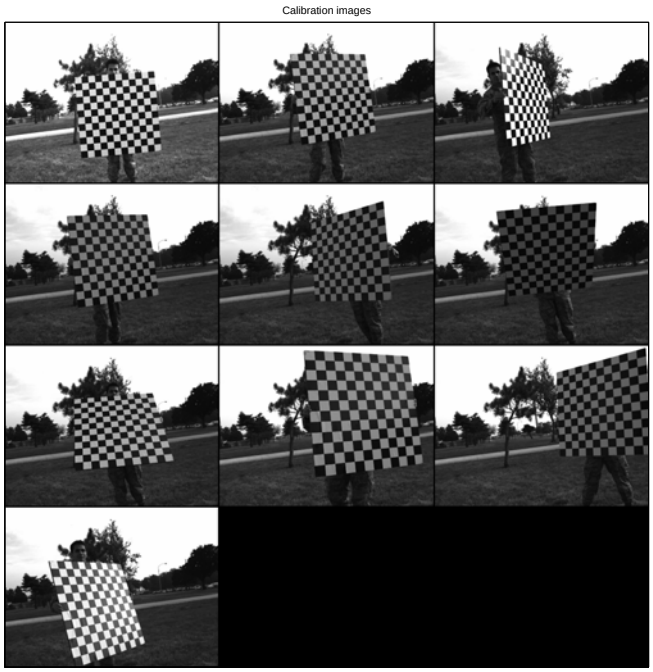


Figure 1.1: Traditional planar calibration surface. The chessboard pattern printed on the planar surface allows for practical feature mapping from simple corner detection.

1.2 Problem Formulation

The problem addressed in this research is composed of two parts. First, the current camera calibration procedure is unnecessarily manual and therefore slow and prone to user-induced error. The first goal is to automate the entire camera calibration process by using a different planar calibration surface, one with distinct features that can be automatically recognized and matched by a computer without user interaction (eliminating the need to click on the outer four corners of the chessboard). Second, the current landmark tracking algorithm does not account for affine distortions arising from drastic changes in vehicle position and camera selection, such as when a feature tracked by a forward camera moves into the field of view of a side camera. Therefore, the second goal is to develop an affine distortion prediction algorithm that preemptively transforms tracked features using inertial measurements in

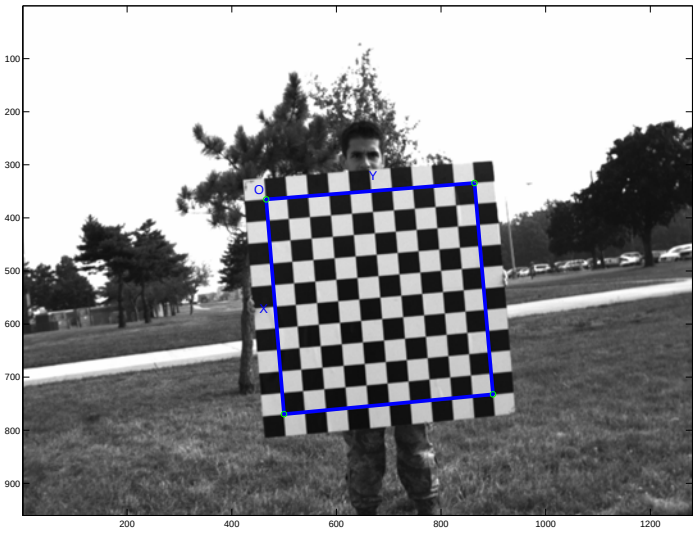


Figure 1.2: Outer corner delineation. The user must individually click on the outer four corners of each image to identify the boundaries of the calibration surface.

order to improve feature matching success rate during extended periods of navigation and over orthogonal cameras. An increased feature matching success rate will directly improve the navigation solution by reducing error.

1.3 Research Contributions

The primary contribution of this research is an enhanced image-aided navigation algorithm that includes a deeply-coupled image and inertial navigation solution with predictive affine distortion modeling. Additionally, this research delivers a fully automated camera calibration algorithm that enables the calibration of cameras with minimal overlapping between fields of view. The algorithms described herein generate a stable navigation solution accurate to within 1.5% (of distance traveled) in position, 1 meter per second in velocity and 4 degrees in attitude using a consumer-grade inertial sensor and three consumer-grade cameras.

Figure 1.3: Feature matching illustration. Successful matches from t_i to t_{i+1} are denoted by green search areas while unsuccessful matches are denoted by yellow search areas.

1.4 Outline

The remainder of this thesis is divided into four additional chapters. Chapter II discusses the background topics needed to build a foundational understanding of inertial navigation, imaging sensors, camera calibration, image-aided navigation and feature generation and matching. Chapter III outlines the methods used to develop the automatic calibration and affine distortion prediction algorithms as well as the experimental procedures used to evaluate them. The experimental results are presented and analyzed in Chapter IV, and finally, conclusions and suggestions for further development are made in Chapter V.

II. Background

This chapter outlines the major concepts needed to implement the automatic calibration and affine distortion prediction algorithms described in this thesis and to understand their importance. The notational conventions used throughout this thesis are presented in Section 2.1. A brief overview on the use of reference frames for navigation is given in Section 2.2. Section 2.3 overviews the basic concepts in inertial navigation. Notable digital signal processing techniques used in this research are discussed in Section 2.4. Camera models and camera calibration techniques are summarized in Section 2.5. Sections 2.6 and 2.7 cover the basic concepts in Kalman filtering and image-aided navigation as well as their associated limitations. The major contributions from previous research efforts are briefly discussed throughout the appropriate sections.

2.1 Mathematical Notation

The quantities expressed in equations, figures, tables and text throughout this research adhere to the following conventions:

Scalars: Scalars are represented by either upper or lowercase characters in *italics*, e.g., a or A.

Vectors: Vector quantities are represented by lowercase characters in **bold**, e.g., $\mathbf{a}$. Unless specifically stated otherwise, all vectors should be interpreted as column vectors.

Vector Components: The scalar components of a vector are represented with subscripts indicating their corresponding axes, e.g., the x-component of the vector $\mathbf{a}$ is represented as a_x.

Homogeneous Vectors: Homogeneous vectors are built by augmenting standard vectors with an additional component equal to 1 and are represented with an underscore, e.g., $\underline{\mathbf{a}}$. The use of homogeneous vectors is further discussed in Section 2.2.1.3.

Matrices: Matrices are represented by uppercase characters in **bold**, e.g., $\mathbf{A}$ or $\mathbf{\Psi}$.

Estimated Variables: Variables that represent an estimate of a particular quantity are represented with the "hat" accent, e.g., $\hat{\mathbf{a}}$.

Direction Cosine Matrices: Direction cosine matrices representing a rotation from frame a to frame b are denoted by $\mathbf{C}_a^b$.

Reference Frame: If a vector is expressed in a specific reference frame, a superscript letter is used to designate the reference frame, e.g., $\mathbf{p}^a$ is a vector in the a frame.

Relative Position or Motion: In cases where is it important to specify relative motion, combined subscript letters are used to designate the frames, e.g., $\boldsymbol{\omega}_{ab}$ represents the angular rate vector from frame a to frame b.

A Priori and A Posteriori Estimates: When describing the operation of a Kalman filter algorithm, it is necessary to distinguish between estimates computed before (*a priori*) or after (*a posteriori*) a measurement update. In such instances, a "minus" character superscript is added to the variable for *a priori* estimates while a "plus" character superscript is added to *a posteriori* estimates, e.g., $\hat{\mathbf{a}}(t^-)$ or $\hat{\mathbf{a}}(t^+)$.

2.2 Reference Frames

Navigation reference frames are fundamentally important when expressing position, velocity, and orientation of a body. For this research, the following reference frames are defined based on those presented in [3], [20] and [22]:

The true inertial frame (I-frame) - a theoretical reference frame in which Newton's laws of motion apply. The frame is defined by a non-accelerating, non-rotating orthonormal basis in $\mathbb{R}^3$. Because of the relative nature of the universe, the true inertial frame has no predefined origin or orientation.

The Earth-centered inertial frame (i-frame) - an orthonormal basis in $\mathbb{R}^3$, with its origin at the center of mass of the Earth. The x and y axes are located

on the equatorial plane with the x-axis pointing towards Aries. The z-axis points towards the North Pole. The i-frame is a non-rotating frame, but it does accelerate with respect to the true inertial frame due to the relative rotation between celestial bodies. However, for terrestrial navigation purposes, it can be considered an inertial reference frame. The i-frame is illustrated in Figure 2.1.

The Earth-centered Earth-fixed frame (e-frame) - an orthonormal basis in $\mathbb{R}^3$, with its origin also at the Earth's center of mass. The e-frame is rigidly attached to the Earth, with the x-axis on the equatorial plane pointing toward the Greenwich meridian, the z-axis aligned with the North Pole, and the y-axis on the equatorial plane pointing toward 90 degrees East longitude. Because the e-frame is a true Cartesian reference frame, some navigation computations are simplified. The e-frame is illustrated in Figure 2.1.

The vehicle-fixed navigation frame (n'-frame) - an orthonormal basis in $\mathbb{R}^3$, with its origin located at a predefined point on a vehicle (e.g., the vehicle's center of gravity or the center of a triad of inertial sensors, etc.) The vehicle-fixed navigation frame's x, y, and z axes point in the North, East and down (NED) directions, respectively. Although the concept of down is open to interpretation, for the purposes of this research, down is defined as the direction a plumb line would point due to gravity. The n'-frame rotates with respect to the e-frame due to translational motion of the vehicle and the rotation of the earth. The n'-frame is illustrated in Figure 2.1.

The Earth-fixed navigation frame (n-frame) - an orthonormal basis in $\mathbb{R}^3$, with its origin located at a predefined location on the Earth, typically on the surface. The Earth-fixed navigation frame's x, y, and z axes point in the North, East, and down directions relative to the origin, respectively. As in the previous case, down is defined as the direction of the gravity vector. In contrast to the vehicle-fixed navigation frame, the Earth-fixed navigation frame remains fixed to the surface of the Earth. While this frame is not useful for very-long

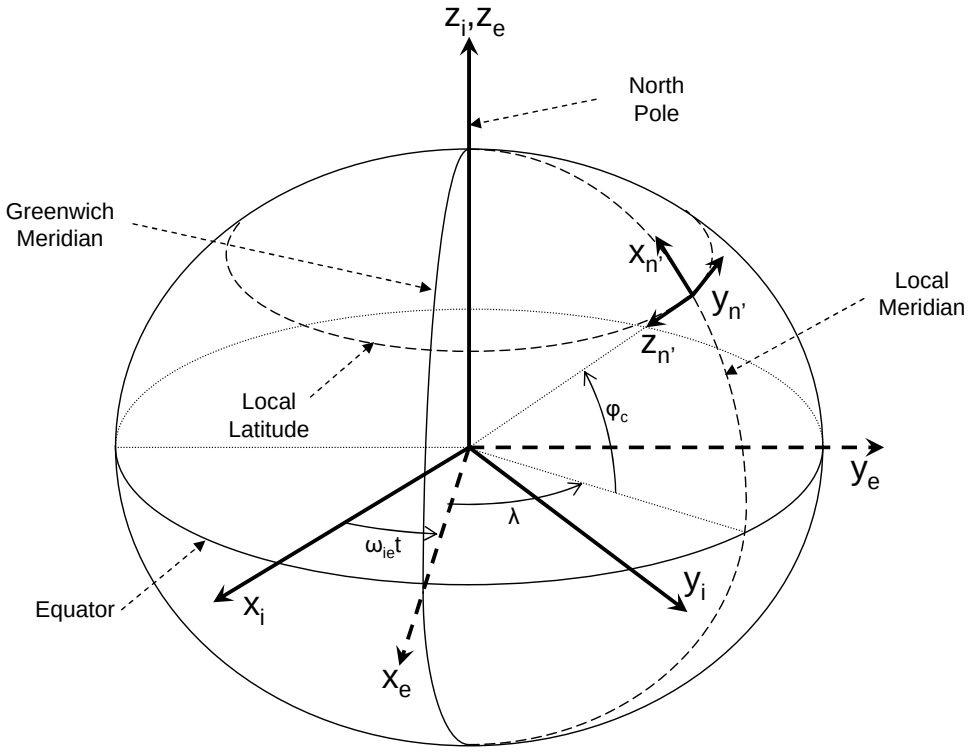


Figure 2.1: The inertial and Earth frames originate at the Earth's center of mass, while the vehicle-fixed navigation frame's origin is located at a fixed location on a vehicle [22].

distance navigation, it can simplify the navigation kinematic equations for local navigation routes. The n-frame is illustrated in Figure 2.2.

The body frame (b-frame) - an orthonormal basis in $\mathbb{R}^3$, rigidly attached to the vehicle with its origin co-located with the navigation frame. The x, y, and z axes point out the nose, right wing, and bottom of an aircraft, respectively. Strapdown inertial sensors are fixed to the b-frame, although they may not be located at the origin or aligned with the axes. The b-frame is shown in Figure 2.3.

The camera frame (c-frame) - an orthonormal basis in $\mathbb{R}^3$, rigidly attached to a camera, with its origin at the camera's optical center. The x and y axes point up and to the right, respectively, and are parallel to the image plane of the camera. The z-axis points out of the camera perpendicular to the image plane. The c-frame is shown in Figure 2.4.

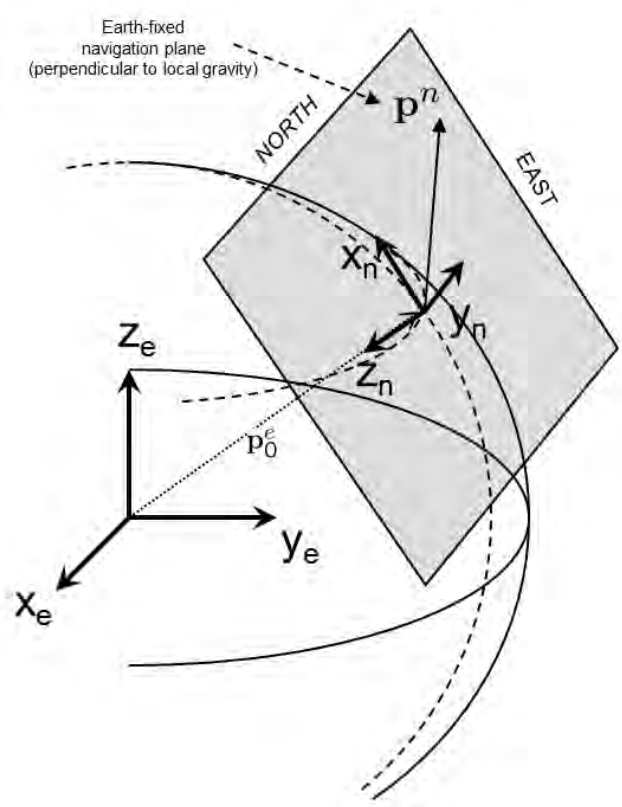


Figure 2.2: The Earth-fixed navigation frame is a Cartesian reference frame with the x and y axes perpendicular to the gravity vector, the z-axis aligned with the gravity vector, and its origin fixed to the Earth [22].

The binocular disparity frame (c_0-frame) - an orthonormal basis in $\mathbb{R}^3$, which is rigidly attached to the lever arm located between cameras in a binocular configuration, with its origin at a specified point on the lever arm. The x, y, and z axes point up, right, and forward, respectively, and parallel to the corresponding c_a-frame axes. The c_0-frame is shown in Figure 2.5.

Image frame (pix-frame) - an orthonormal basis in $\mathbb{R}^2$ with its origin beyond the upper-left pixel of a digital image. Multiple conventions exist throughout image processing literature for pixel indexing and axis orientation. In this thesis, images are indexed according to the matrix storage format used by The Mathworks, Inc's Matlab software, with the upper left pixel indexed as (1,1), the x-axis down the left side and y-axis across the top of the image.

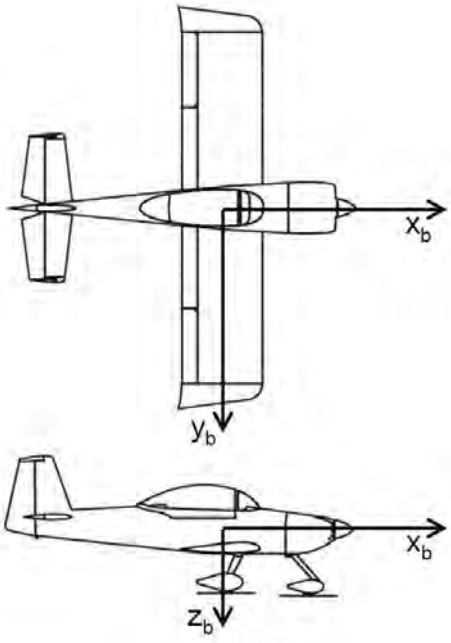


Figure 2.3: Aircraft body frame illustration. The aircraft body frame originates at the aircraft center of gravity [22].

Calibration board frame (*cal*-frame) - an orthonormal basis in $\mathbb{R}^3$ with its origin at the upper-left corner of the calibration board. The x and y axes are aligned with the calibration board while the z-axis points upward, away from the board. The *cal*-frame is shown in Figure 2.6.

2.2.1 Coordinate Transformations. It is often necessary to relate measured vector quantities from one frame to another. This is accomplished by applying a coordinate transformation matrix, which can be composed of a translation and/or a rotation component. Translations describe relative positions between the origins of two reference frames, while rotations describe the relative angle between the principal axes of two reference frames.

2.2.1.1 Translation Vectors. When two reference frames differ only in relative origin position (e.g., their principal axes are co-directional), a translation

Figure 2.4: Camera frame illustration. The camera reference frame originates at the optical center of the lens [22].

vector can be used to express vectors from a coordinate frame in terms of a second frame. The description of point P in terms of frame a can be expressed in terms of a co-directional frame b using

$$\mathbf{p}^b = \mathbf{p}^a - \mathbf{p}^a_{ab} \tag{2.1}$$

where $\mathbf{p}^b$ is the position of point P in frame b and the vector $\mathbf{p}^a_{ab}$ is the translation of the a-frame to the b-frame in a-frame coordinates. Figure 2.7 illustrates the application of a translation vector between two reference frames.

2.2.1.2 Direction Cosine Matrices. While a translation vector defines the relative origin position between two reference frames, a Direction Cosine Matrix (DCM) defines the relative rotation between the principal axes of two reference frames. DCMs apply rotations to each axis in a reference frame, using the standard Euler angles, and in the following order [16]:

1. First, a rotation angle ψ is applied about the z-axis of the originating frame.

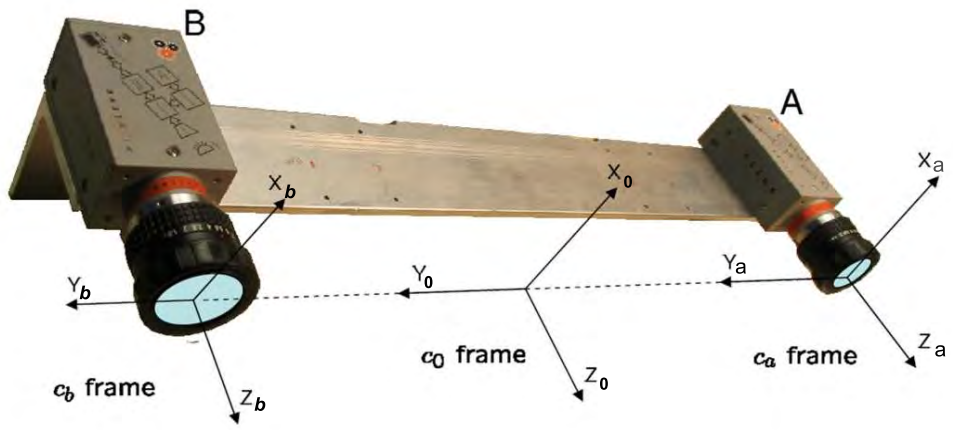


Figure 2.5: Binocular disparity frame illustration. The binocular disparity frame originates at the midpoint between the optical center of the two camera frames, c_a and c_b [22].

2. Next, a rotation angle θ is applied about the y-axis of the newly formed intermediate reference frame.

3. Finally, a rotation angle ϕ is applied about the x-axis of the second intermediate reference frame formed in step 2.

   Once the desired Euler angles are defined, a DCM can be built using

$$\mathbf{C}_a^b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\phi & sin\,\phi \\ 0 & -sin\,\phi & cos\,\phi \end{bmatrix} \begin{bmatrix} cos\,\theta & 0 & -sin\,\theta \\ 0 & 1 & 0 \\ sin\,\theta & 0 & cos\,\theta \end{bmatrix} \begin{bmatrix} cos\,\psi & sin\,\psi & 0 \\ -sin\,\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where $\mathbf{C}_a^b$ is the DCM that rotates the principal axes of frame a to become co-linear with the principal axes in frame b. A vector, originally written in terms of frame a, $\mathbf{y}^a$ can be represented in terms of frame b using

$$\mathbf{y}^b = \mathbf{C}_a^b \mathbf{y}^a \tag{2.3}$$

Additionally, due to their purely rotational function, DCMs have the following properties:

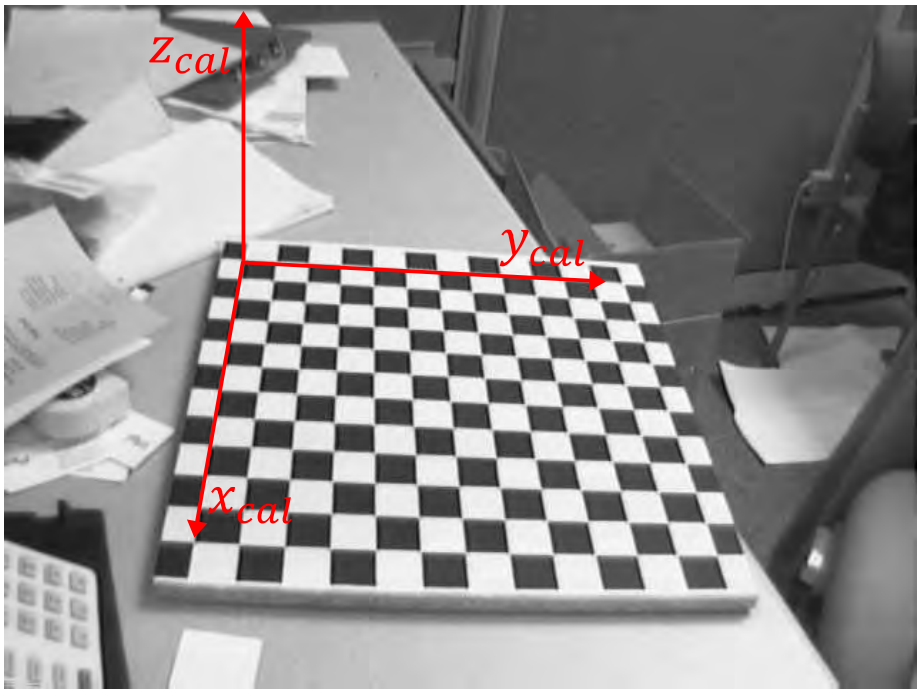


Figure 2.6: Calibration board frame illustration. The calibration board frame originates at the upper-left corner of the calibration board [1].

- The determinant of any DCM is always equal to 1, ensuring vector magnitudes are always preserved during rotations.
- A DCM is guaranteed to have an inverse since it has a nonzero determinant.
- A DCM that aligns frame b to frame a is simply the inverse of the DCM that aligns frame a to frame b.
- The inverse of a DCM equals its transpose.

2.2.1.3 Transformation Matrices. Transformation matrices provide a practical method to apply both a translation and a rotation to a vector or point using one mathematical operation. In order to use transformation matrices, a homogeneous vector is constructed by augmenting the original vector with an additional element set equal to one, i.e.,

$$\underline{\mathbf{p}} = [p_x \quad p_y \quad p_z \quad | \quad 1]^T \tag{2.4}$$

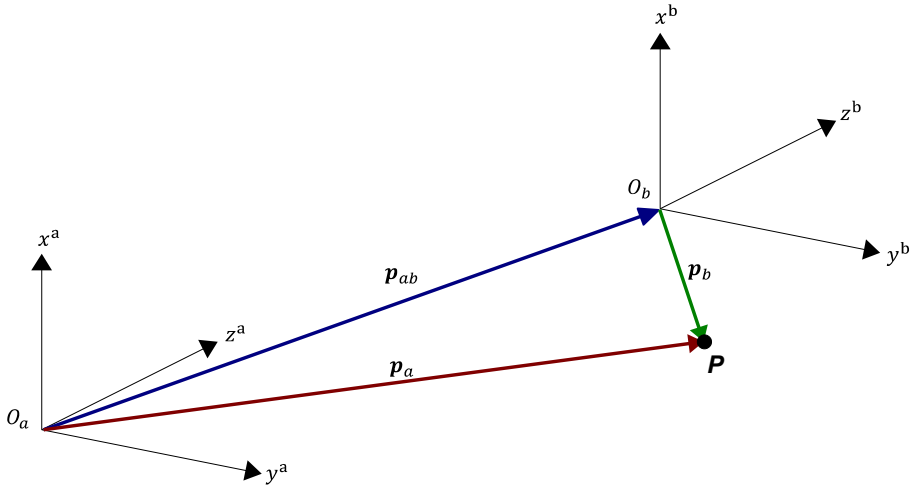


Figure 2.7: Translation vectors. The vector $\mathbf{p}_{ab}$ translates the origin of frame a to the origin of frame b and can be used to describe the position of point P relative to frame b given its coordinates in terms of frame a.

A transformation matrix is then composed of a rotation DCM and a translation vector. For example, given a DCM $\mathbf{C}_a^b$, which aligns frame a to frame b, and a translation vector $\mathbf{p}_{ab}^a$, which collocates the origin of frame a with the origin of frame b, the a-frame vector $\mathbf{p}^a$ can be expressed in terms of frame b using

$$\underline{\mathbf{p}}^b = \mathbf{T}_a^b \underline{\mathbf{p}}^a \tag{2.5}$$

where

$$\mathbf{T}_a^b = \begin{bmatrix} \mathbf{C}_a^b & | & -\mathbf{p}_{ab}^a \\ - & & - \\ 0_{1\times 3} & | & 1 \end{bmatrix} \tag{2.6}$$

2.3 Inertial Navigation

Inertial navigation is based on the basic concept that, starting from a known location, attitude, and velocity, a vehicle's current position and attitude can be estimated by integrating measured changes in velocity and rotation. Inertial navigation measurement devices such as an Inertial Measurement Unit (IMU) consist of three

accelerometers, which measure specific force and three gyroscopes (commonly referred to as gyros), which measure rotational velocity relative to the I-frame.

When equipped with the navigation equations necessary to integrate for position, velocity and attitude, the entire measurement system is referred to as an Inertial Navigation System (INS). In general, there are two types of INSs: platform and strapdown. A platform INS is composed of measurement sensors mounted on a platform and gimbaled such that the vertical sensor of the unit is always aligned with local gravity. In contrast, a strapdown INS consists of a a simple IMU rigidly mounted to a vehicle with its motion sensors mounted orthogonally and aligned with the vehicle's b-frame. During the experiments conducted for this thesis, vehicles were equipped with Micro Eletro-mechanical Systems (MEMS) grade strapdown IMUs due to their small size, light weight and low-power consumption. The navigation equations were then implemented in software separate from the device. Further information regarding strapdown inertial navigation can be found in the works of Titterton and Weston [20].

2.4 Digital Image Processing

This section explores the basic concepts behind digital imaging and signal processing on which camera calibration, landmark tracking and image-aided navigation are built.

2.4.1 Digital Imaging. In this research, the digital imaging model consists of a light source, a subject or scene, an optical collection device and an imaging sensor. The light source illuminates the subject while the reflected light rays are collected by a lens and captured by an optical image sensor. The image sensor translates image intensities into voltages, which are then quantized into discrete Red Green Blue (RGB) values by an analog-to-digital converter as shown in Figure 2.8.

2.4.1.1 Perspective Projection Geometry. Perspective projection geometry describes the mathematical relationship between the 3-D coordinates of a

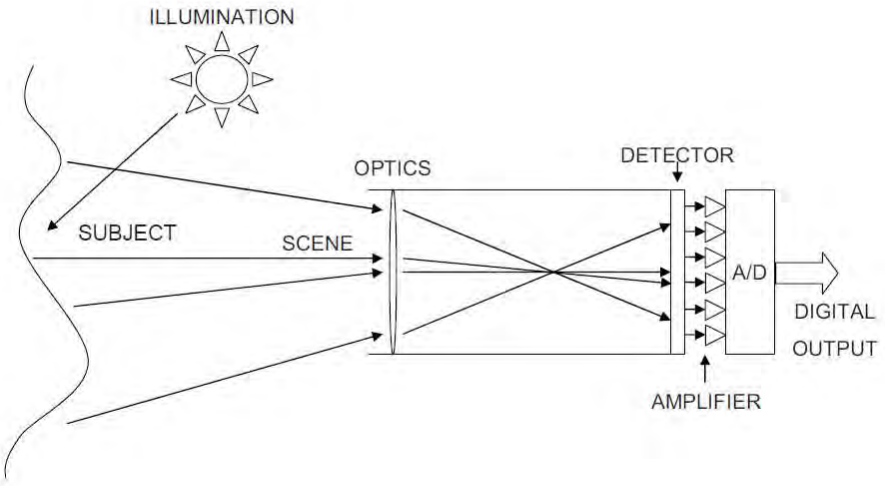


Figure 2.8: Image sensor diagram. Light captured by the lens is interpreted by the imaging sensor to produce a digital image [16].

point in the c-frame and its projection onto the 2-D pix-frame. The physical properties of a scene can then be extracted from captured images using such relationships. The process by which a 3-D point is projected onto the 2-D image plane is referred to as perspective projection. The perspective projection model is derived from the basic pinhole camera shown in Figure 2.8. In the standard pinhole camera model, light rays travel from the subject, pass through the lens and are projected onto the focal plane located at a distance f behind the lens. However, because light rays travel in straight lines through the lens and towards the origin of the imaging sensor, the received image is naturally inverted. In order to account for and undo this natural image inversion, a virtual image plane is placed at a distance f in front of the lens, on which geometric proportions are identical to those perceived by the imaging sensor. The modified pinhole camera model is illustrated in Figure 2.9. The vectors depicted in Figure 2.9 will be discussed in the next section.

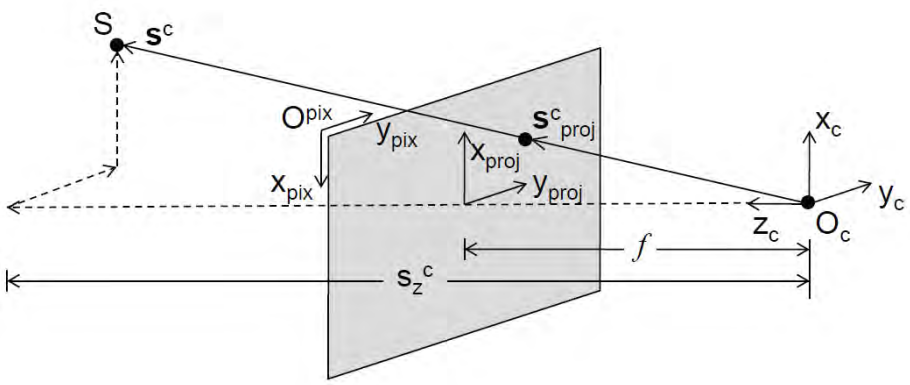


Figure 2.9: Pinhole camera model. The effects of image rotation through the lens are modeled by placing a virtual image plane in front of the sensor [16].

2.4.1.2 Intrinsic Camera Matrix. Using the properties of perspective projection geometry, the mathematical expression that transforms a 3-D point in the c-frame into a 2-D point in the *pix*-frame can be developed. The vector $\mathbf{s}^c$ originates at the camera sensor origin O_c and terminates at S. As shown in Figure 2.9, the vector $\mathbf{s}^c_{proj}$, which also originates at O_c but terminates at the image plane, is collinear with and a scalar multiple of $\mathbf{s}^c$. Since the image plane is located at $z = f$, the z-coordinate of $\mathbf{s}^c_{proj}$ is also f. Using similar triangle geometry, the scaling factor which relates the two vectors can be derived as shown in Equation (2.7)

$$\mathbf{s}^c_{proj} = \frac{f}{\mathbf{s}^c_z}\mathbf{s}^c \tag{2.7}$$

Next, the z-coordinate in $\mathbf{s}^c_{proj}$ is discarded as the point moves from the c-frame to the *pix*-frame. The z-coordinate removal can be represented mathematically using

$$\mathbf{s}^{proj} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{s}^c_{proj} \tag{2.8}$$

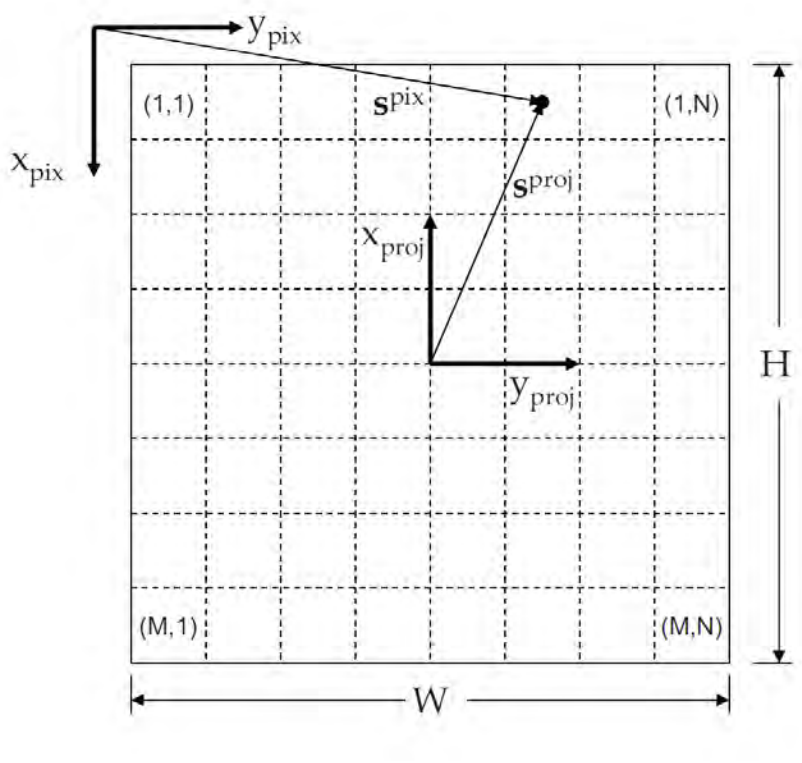


Figure 2.10: Image plane diagram. The image plane has both physical dimensions $H \times W$ and pixel dimensions $M \times N$. The x and y axes from the c-frame project onto the image plane at the coordinates $(\frac{M+1}{2}, \frac{N+1}{2})$ [16].

where $\mathbf{s}^{proj}$ is the 2-D projection of $\mathbf{s}^{c}_{proj}$ onto the pix-frame. Finally, the translation and scaling between the c-frame and the pix-frame is shown in Figure 2.10. The relative rotation between the two frames involves simply inverting the x-axis and is done using

$$\mathbf{p}^{pix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{p}^{proj} \tag{2.9}$$

where the vectors $\mathbf{p}^{pix}$ and $\mathbf{p}^{proj}$ describe the position of a point p in the pix-frame and projected c-frame respectively. The projected c-frame is depicted in Figure 2.10 by the x_{proj} and y_{proj} axes and is obtained by projecting the 3-D c-frame onto the 2-D image plane.

The scaling factors are derived from the ratios of physical-to-pixel dimensions in the image plane. The x-axis is scaled from a physical length H to a pixel length M which leads to a scaling factor of M/H in the x-direction. Similarly, the y-axis is scaled using the factor N/W. Finally, the origin of the c-frame is offset by $\frac{M+1}{2}$ pixels in the x-direction and $\frac{N+1}{2}$ pixels in the y-direction. Combining all transformations between the c-frame and pix-frame up to this point yields

$$\mathbf{s}^{pix} = \begin{bmatrix} -\frac{M}{H} & 0 & 0 \\ 0 & \frac{N}{W} & 0 \end{bmatrix} \mathbf{s}^{c}_{proj} + \begin{bmatrix} \frac{M+1}{2} \\ \frac{N+1}{2} \end{bmatrix} \tag{2.10}$$

where the vector $\mathbf{s}^{pix}$ is the 2-D projection of a point in the c-frame, described by $\mathbf{s}^{c}_{proj}$, onto the pix-frame. Finally, homogeneous vector representations can be used to describe the combined rotation, translation, and scaling through the transformation matrix

$$\mathbf{T}^{pix}_{c} = \begin{bmatrix} -f\frac{M}{H} & 0 & \frac{M+1}{2} \\ 0 & f\frac{N}{W} & \frac{N+1}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

where

$$\underline{\mathbf{s}}^{pix} = \frac{1}{\mathbf{s}^{c}_{z}} \mathbf{T}^{pix}_{c} \mathbf{s}^{c} \tag{2.12}$$

The camera model can also be used to convert coordinates from the pix-frame back to the c-frame. A set of coordinates expressed in terms of the pix-frame can be converted back to c-frame using

$$\mathbf{T}^{c}_{pix} = \mathbf{T}^{pix\,-1}_{c} = \begin{bmatrix} -\frac{H}{fM} & 0 & \frac{H(M+1)}{2fM} \\ 0 & \frac{W}{fN} & -\frac{W(N+1)}{2fN} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

where

$$\underline{\mathbf{s}}^{c} = \mathbf{T}^{c}_{pix}\mathbf{s}^{pix} \tag{2.14}$$

Note that since there is a loss of dimension when transforming from the c-frame to the pix-frame, it is impossible to fully invert the transform. Rather, Equation (2.14) yields a homogeneous 3-D vector that is co-directional with the true vector but not necessarily equal. The stereo-vision techniques designed to recover the lost dimension are discussed in Section 2.5.3.

2.4.2 Scale Invariant Feature Transform. Feature identification and matching are integral processes within the proposed automatic calibration and affine distortion prediction algorithms. Although many feature detection and matching techniques are found throughout computer vision literature, only a few generate features that can be recognized through changes in camera position and orientation with respect to the scene, a property which is of fundamental importance. That is, the feature description for each feature should be invariant to changes in scale, rotation or translation within the image-space. Because of these requirements, the Scale Invariant Feature Transform (SIFT) algorithm developed by Lowe [12] was chosen as the feature detection algorithm used in this research. The SIFT algorithm is composed of four main stages: scale-space extrema detection, keypoint localization, orientation assignment and keypoint description. This section explores how these four stages produce a feature detection algorithm that is invariant to scale, rotation, and translation.

2.4.2.1 Scale-Space Extrema Detection. In the first stage of the transform, stable features are identified within the image across all possible scales using a continuous scale function known as scale-space. The scale-space function $L(x, y, \sigma)$ is built using the Gaussian distribution as a base function kernel and is given by

$$L(x, y, \sigma) = G(x, y, \sigma) \circledast I(x, y) \tag{2.15}$$

where $I(x, y)$ is an input image, $\circledast$ is the convolution operator and $G(x, y, \sigma)$ is given by

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.16}$$

A Difference of Gaussians (DOG) function $D(x, y, \sigma)$ is constructed in order to provide scale invariant detection and is given by

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{2.17}$$

$$= L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.18}$$

which yields a computationally efficient approximation to the scale-normalized Laplacian of Gaussian function [11], thereby providing scale invariance. Figure 2.11 illustrates how the DOG functions are constructed for an input image. The initial image is incrementally convolved with 2-D Gaussian distributions to produce blurred images, which are separated by a constant k in scale-space (shown in the left column). Next, adjacent blurred images in scale-space are subtracted per Equation (2.18) to produce the DOG images (shown on the right column). Groups of blurred images are referred to as octaves and are separated by factors of 2 in scale-space. After an entire octave is completed, the next octave is formed by downsampling the $k = 2$ blurred image by a factor of 2 and repeating the process.

After forming all possible DOG functions, the local extrema within all the resulting images needs to be found. To do so, each sample point is compared to its eight nearest neighbors in the current image as well as the nine nearest neighbors in the scale image above and below as shown in Figure 2.12. Finally, a sample point is considered a possible feature only if it is larger or smaller than all of of its neighbors.

2.4.2.2 Keypoint Localization. Once a candidate feature has been found using the scale-space extrema process, a detailed fit to the nearby data in the image must be performed to determine its location, scale, and ratio of principal curvatures. The data fit allows the rejection of candidate features that have low

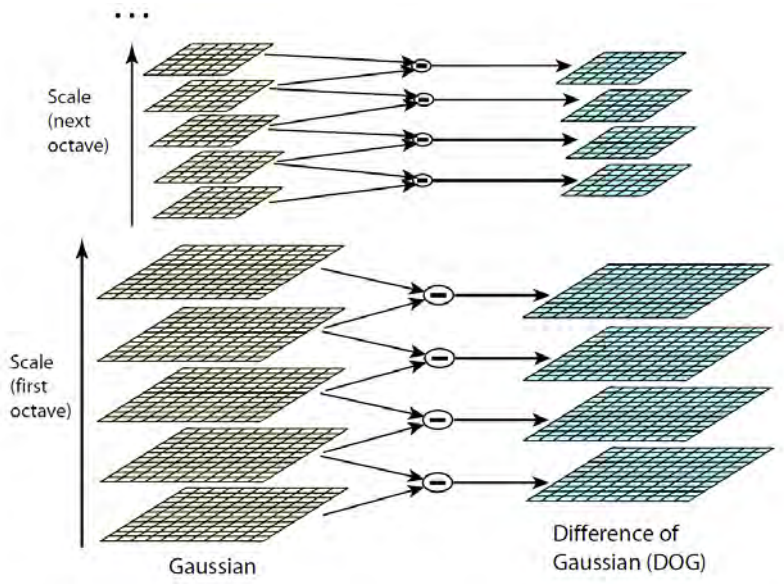


Figure 2.11: Difference of Gaussians illustration. Neighboring scale-space images are subtracted to produce the Difference of Gaussians function [12].

contrast, which makes them susceptible to noise, or are poorly localized along an edge. The data fit is found through a Taylor series expansion (up to the quadratic terms) of the DOG function $D(x, y, \sigma)$ using

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{2.19}$$

where D is evaluated at the candidate feature location and $\mathbf{x} = (x, y, \sigma)^T$ is the offset from the candidate point. The location of the extremum $\hat{\mathbf{x}}$ within the image is then estimated using Equation (2.19) by taking its derivative with respect to $\mathbf{x}$ and setting it equal to zero. If the resulting offset between the estimate of the extremum's location $\hat{\mathbf{x}}$ and the current candidate is larger than 0.5 in any dimension, a different candidate is chosen and the process is repeated. Finally, since even poorly defined candidates will have a high DOG response along the edges of the image, the ratio of their principal axes is analyzed in order to eliminate any candidates that have a high ratio of largest-to-smallest eigenvalues, indicating they are on an edge.

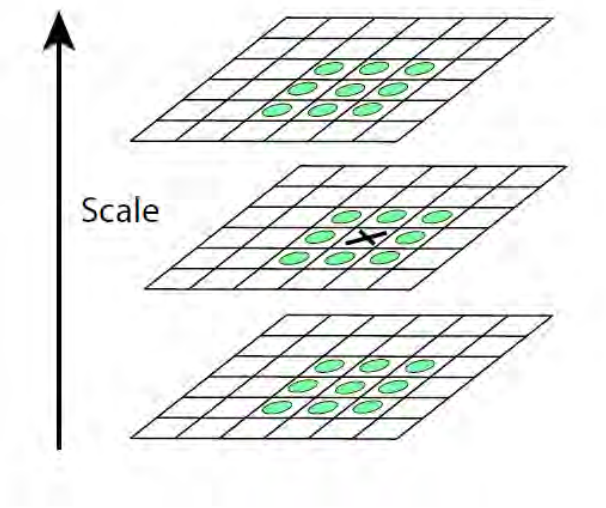


Figure 2.12: Local extrema illustration. A candidate feature must be smaller than or greater than all of its 26 neighbors in scale-space [12].

2.4.2.3 Orientation Assignment. Once all the qualifying features have been selected from the elimination processes described above, a relative orientation is assigned to each feature based on local image properties. Computing the orientation of each feature allows the generation of an unique descriptor, which identifies the feature in its own orientation frame. Therefore, features can later be matched using their descriptors regardless of the absolute orientation of their source images, making the SIFT algorithm rotation invariant. A feature's orientation is computed from the blurred image L which corresponds to the feature's scale σ. The local gradient magnitude and orientation are computed around the feature location in L using

$$m(x,y) = \sqrt{A^2 + B^2} \tag{2.20}$$

$$\theta(x,y) = \tan^{-1}(B/A) \tag{2.21}$$

where

$$A = L(x+1, y) - L(x-1, y) \tag{2.22}$$

$$B = L(x, y+1) - L(x, y-1) \tag{2.23}$$

Finally, a histogram of the orientations is built from sample points around the feature location. Local peaks are then identified within the histogram. Multiple keypoints, consisting of location, scale, and orientation, can be generated from a single feature, consisting only of location and scale, by assigning different orientations. Usually, the orientations in the top 20% of the histogram are used in generating multiple keypoints from a particular feature.

2.4.2.4 Keypoint Description. Up to this point, keypoints consisting of location, scale, and orientation have been generated from an input image. These three parameters are then used to generate local 3-D coordinate frames for each keypoint, providing a repeatable method of keypoint description, which is invariant to such parameters. The final step in the SIFT algorithm generates a more unique keypoint descriptor that provides partial invariance to factors such as illumination and small changes in 3-D viewpoint. The SIFT keypoint descriptor process is based on a model of biological vision where neurons respond to gradients at particular orientations and spatial frequencies. Using this model, a keypoint descriptor is created by computing the gradient magnitude and orientation in a sampled region around the keypoint's location. Next, the orientations are weighed by a 2-D Gaussian function, centered at the keypoint's location, and accumulated into 8-bin histograms, which summarize the contents over 4×4 subregions as shown in Figure 2.13. Consequently, the keypoint descriptor is a $8 \times 4 \times 4 = 128$ point normalized vector containing the values of the gradient orientation histogram bins in the 4×4 subregions.

2.4.3 Feature Matching Techniques. Szeliski [19] introduces a few feature matching methods for algorithms that produce feature or keypoint descriptors such

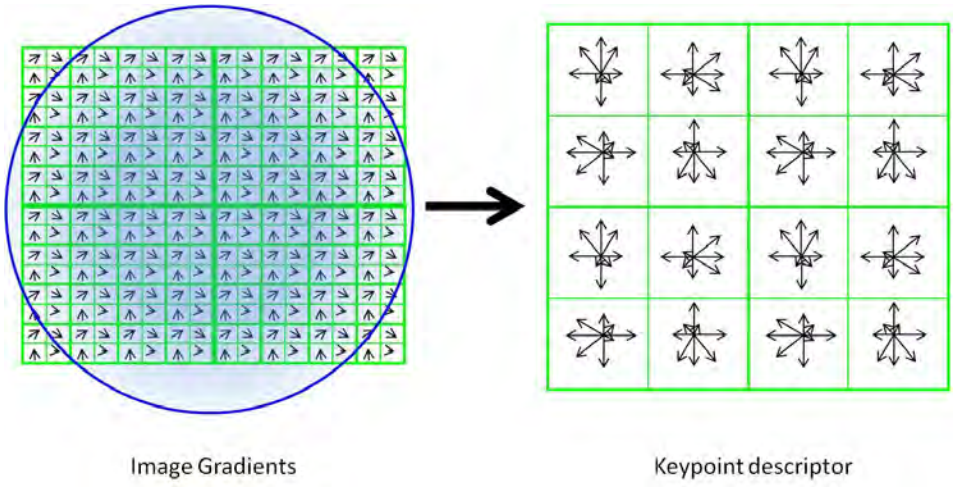


Figure 2.13: Keypoint descriptor illustration. A feature's unique descriptor is composed from a histogram of the image gradients around its location [12].

as SIFT. In this thesis, the particular feature matching algorithm used for both automatic calibration and image-aided navigation is the Nearest Neighbor Distance Ratio (NNDR). NNDR was chosen over other common techniques such as Random Sample Consensus (RANSAC) [4] due to the deep-coupling of inertial and imaging sensors provided by the image-aided navigation algorithm discussed in Section 2.7. Feature matching is accomplished by comparing feature descriptors, which are normalized vectors in 128-space as discussed in Section 2.4.2.4. Using NNDR, potential matches are found by computing the Euclidean distance in 128-space from a reference feature to all possible candidate features. Then, the following ratio is computed using the two closest candidate features

$$r = \frac{\mathbf{d}_1}{\mathbf{d}_2} = \frac{||\mathbf{d}_R - \mathbf{d}_A||}{||\mathbf{d}_R - \mathbf{d}_B||} \tag{2.24}$$

where $\mathbf{d}_R$ is the reference feature descriptor and $\mathbf{d}_A$ and $\mathbf{d}_B$ are the two closest candidate feature descriptors. The NNDR computed in Equation (2.24) is a measure of the strength of a particular match. If the ratio is much less than 1, then $\mathbf{d}_A$ is much closer to $\mathbf{d}_R$ than $\mathbf{d}_B$ and therefore the match is strong. In contrast, if the ratio is

close to 1, then both $\mathbf{d}_A$ and $\mathbf{d}_B$ are very close to $\mathbf{d}_R$ and therefore the match may be too weak to declare it with confidence. Setting a maximum NNDR ratio that must be satisfied helps prevent false matches.

2.4.4 Affine Keypoint Distortions. The keypoint descriptor generation and matching algorithms discussed up to this point provide full scale and rotation invariance, as well as partial illumination invariance. However, another major necessity in feature tracking is affine distortion invariance. As the viewpoint of a physical object changes, either through vehicle motion or camera angle, the image gradients from which keypoint descriptors are derived change along with the image of the object. Fortunately, the change can be adequately modeled using affine transformations on the initial image. Using this concept as a foundation, Morel and Yu developed the Affine Scale Invariant Feature Transform (ASIFT) algorithm [15], which they claim provides full affine invariance. The ASIFT algorithm recursively uses the basic SIFT algorithm described in the previous section as a core function. Much like the scale-space development used by Lowe to provide scale invariance, Morel and Yu simulate a series of affine transformations on a given image and calculate standard SIFT keypoint descriptors for each of the simulated images. When compared to SIFT, the ASIFT algorithm outputs nearly nine times as many keypoints per input image because of the affine distortion simulations. The ASIFT algorithm creates multiple versions of the standard SIFT descriptors for each image, which increases the probability of obtaining successful matches. The keypoint matching improvement given by ASIFT is illustrated in Figures 2.14 and 2.15. The predictive affine distortion modeling algorithm introduced in Chapter III is partly based on the ASIFT concept. However, as later explained in Section 2.6, it only needs to generate a single affine-transformed image, since distortion parameters are estimated in the Extended Kalman Filter.

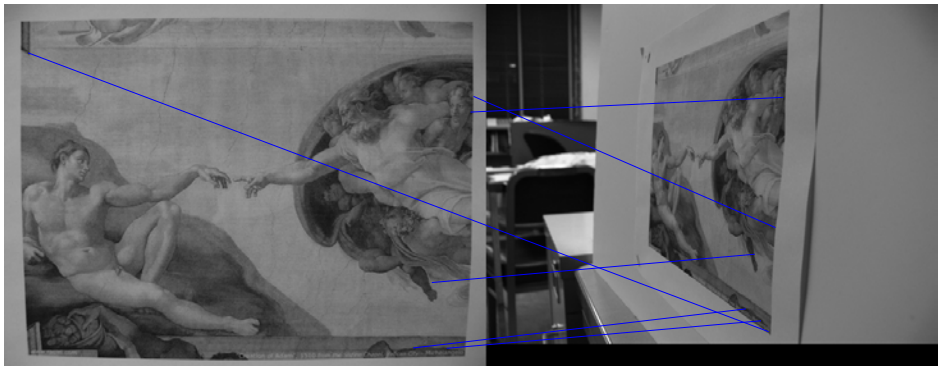


Figure 2.14: SIFT matches during high affine distortion. Since the SIFT algorithm does not account for affine distortions, 0 successful matches and 6 false matches are obtained with an NNDR of 0.55.

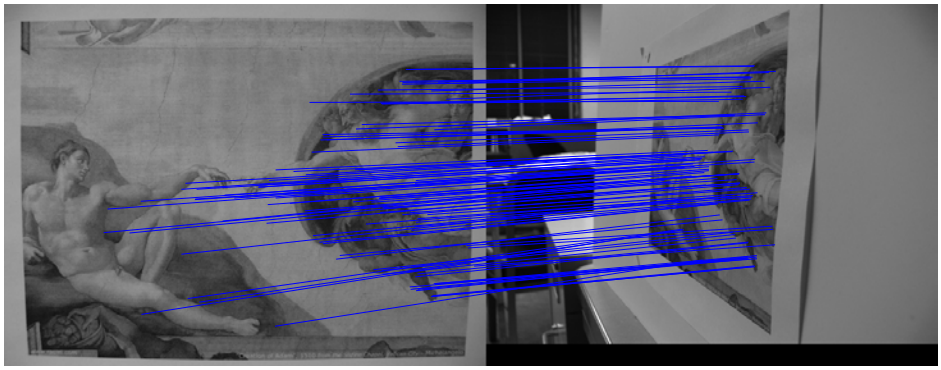


Figure 2.15: ASIFT matches during high affine distortion. Because the ASIFT algorithm provides full affine invariance, 100 successful matches and 0 false matches are obtained with an NNDR of 0.55.

2.5 Camera Calibration

Real world imaging sensors present nonlinear lens distortions that must be modeled and accounted for in computer vision algorithms such as image-aided navigation. Additionally, the relative rotation and translation between a camera and the scene it captures (referred to as extrinsic parameters) must be determined for proper image analysis. This section explores the camera calibration techniques that are currently used to compensate for the distortion effects and calculate the extrinsic configuration parameters in a computer vision system.

2.5.1 Distortion Models. The first step in determining the nonlinear lens distortion of an imaging sensor is to model the distortion. Brown [2] groups the distortion parameters into radial, tangential, and skew components.

2.5.1.1 Radial Distortion. Radial distortion, the most noticeable of the three, causes straight lines to appear curved in the images produced by the sensor. The main cause of radial distortion is non-uniform magnification inside the sensor's optics. Radial distortion is modeled as a function of r, the Euclidean distance between a point given by the vector $\mathbf{s}^{proj}$ in the c-frame and the frame's origin, which is given by

$$d^{rad} = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \tag{2.25}$$

$$r = \sqrt{(s_x^{proj})^2 + (s_y^{proj})^2} \tag{2.26}$$

where d^{rad} is the radial distortion factor for a particular length r, and k_1, k_2, and k_3 are constant distortion coefficients calculated through camera calibration.

2.5.1.2 Tangential Distortion. Tangential distortion is less visible on images produced by the sensor and causes the principal point $\mathbf{c}$ to shift away from the true (geometric) center of the image plane. The causes of tangential distortion include differing curvatures in the front and back of the sensor's lens and misalignment between the sensor's lens and collection array. Brown models tangential distortion as a 2-D function of r, s_x^{proj} and s_y^{proj} defined by

$$\mathbf{d}^{tan} = \begin{pmatrix} 2p_1(s_x^{proj})(s_y^{proj}) + p_2[r^2 + 2(s_x^{proj})^2] \\ p_1[r^2 + 2(s_y^{proj})^2 + 2p_2(s_x^{proj})(s_y^{proj})] \end{pmatrix} \tag{2.27}$$

2.5.1.3 Skew Factor. The distortion caused by a skew factor α_c is the most difficult to visualize and refers to the orthogonality between the x and y axes in the pix-frame. Although the skew factor for most real world imaging systems is

nearly zero, it is not negligible when extracting real world metrics from images of a scene. The skew factor determines how far the *pix*-frame axes are from perfectly orthogonal. When included in a camera model, the skew factor multiplies the upper middle coefficient in the camera transformation matrix $\mathbf{T}_c^{pix}$ such that

$$\underline{\mathbf{s}}^{pix} = \frac{1}{s_z^c} \begin{bmatrix} -f\frac{M}{H} & -\alpha_c f\frac{M}{H} & \frac{M+1}{2} \\ 0 & f\frac{N}{W} & \frac{N+1}{2} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{s}^c \tag{2.28}$$

2.5.1.4 Camera Distortion Model. Combining all three intrinsic distortion parameters generates a complete mathematical representation of the projection of a point given by the vector $\mathbf{s}^c$ in the *c*-frame onto the distorted *pix*-frame, which yields the true pixel coordinates $\mathbf{s}^{pix}$. The complete camera distortion model is given by

$$\underline{\mathbf{s}}^{pix} = \frac{1}{s_z^c} \begin{bmatrix} -f\frac{M}{H} & -\alpha_c f\frac{M}{H} & \frac{M+1}{2} \\ 0 & f\frac{N}{W} & \frac{N+1}{2} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d^{rad} & 0 & d_x^{tan} \\ 0 & d^{rad} & d_y^{tan} \\ 0 & 0 & 1 \end{bmatrix} \mathbf{s}^c \tag{2.29}$$

2.5.2 Calibration Algorithms. Having defined a camera distortion model, an adequate method for calculating its parameters can be developed. Many camera calibration techniques are found throughout computer vision literature. While most require the extensive use of complicated calibration equipment, Dr. Zhang introduces a practical algorithm that requires only a planar surface with known feature coordinates [24]. Furthermore, Zhang's algorithm contains the fundamental techniques used by Bouguet in his camera calibration toolbox for Matlab® [1].

2.5.2.1 Camera Calibration from a Planar Surface. Zhang [24] develops a flexible and robust algorithm for calculating the intrinsic and extrinsic parameters for a given imaging sensor (or set of sensors). Using the distortion model developed in the previous section, Zhang defines the relationship between a 3-D point

and its 2-D image projection in a fashion similar to Equation (2.29) such that

$$s\underline{\mathbf{m}} = \mathbf{A}[\mathbf{R} \quad \mathbf{t}]\underline{\mathbf{M}} \tag{2.30}$$

$$\mathbf{A} = \begin{bmatrix} \alpha & c & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.31}$$

where s is an arbitrary scale factor, the augmented matrix $[\mathbf{R} \quad \mathbf{t}]$ contains the extrinsic parameters, and the matrix $\mathbf{A}$ contains the intrinsic parameters. Comparing Equation (2.28) with Equation (2.31) yields

$$\mathbf{A} = \mathbf{T}_c^{pix} \begin{bmatrix} 1 & \alpha_c & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.32}$$

and

$$\alpha = -f\frac{M}{H} \tag{2.33}$$

$$c = -\alpha_c f\frac{M}{H} \tag{2.34}$$

$$u_0 = \frac{M+1}{2} \tag{2.35}$$

$$\beta = f\frac{N}{W} \tag{2.36}$$

$$v_0 = \frac{N+1}{2} \tag{2.37}$$

Additionally, since the extrinsic parameters describe the relative rotation and translation between the 3-D *c*-frame and the 2-D *pix*-frame, the augmented matrix $[\mathbf{R} \quad \mathbf{t}]$ is of the form

$$[\mathbf{R} \quad \mathbf{t}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{11} & r_{12} & r_{13} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.38}$$

Since the 3-D points to be used for calibration all lie on a planar surface, Zhang begins by defining extrinsic parameters relative to the *cal*-frame such that all 3-D points lie on $z = 0$. Since all points lie on the same plane, Zhang drops the third coordinate and reduces all 3-D point coordinates to 2-D, yielding

$$s\underline{\mathbf{m}} = \mathbf{H}\underline{\mathbf{M}} \tag{2.39}$$

$$\mathbf{H} = \mathbf{A}[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}] \tag{2.40}$$

where $\mathbf{H}$ is a 3×3 homography matrix defined up to a scale factor. After estimating $\mathbf{H}$ from a single set of model ($\mathbf{M}$) and image ($\mathbf{m}$) points, Zhang uses the fact that the column vectors $\mathbf{r}_1$ and $\mathbf{r}_2$, which compose $\mathbf{H}$, are orthonormal to define the following additional constraints on the intrinsic parameters

$$\mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 = 0 \tag{2.41}$$

$$\mathbf{h}_1^T \mathbf{A} - T \mathbf{A}^{-1} \mathbf{h}_1 = \mathbf{h}_2 \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 \tag{2.42}$$

where the notation $-T$ represents the inverse and transpose operations and the column vector $\mathbf{h}_i$ is the i^{th} column of $\mathbf{H}$. Using the two constraints given above, Zhang then solves for the extrinsic and intrinsic parameters by observing the same planar surface from at least two views, which must differ by both rotation and translation. Observing more than two views of the planar surface creates an overdetermined system

that can be solved using least-squares. Creating an overdetermined system accounts for noise in the collection process. Further information on the closed form (analytical) and iterative (maximum likelihood) solutions to the calibration system can be found in Zhang's technical report [24].

2.5.2.2 Caltech Camera Calibration Toolbox. Bouguet [1] makes extensive use of Dr. Zhang's calibration method to develop a practical camera calibration user interface for Matlab®. Bouguet uses the exact homography estimation technique described by Zhang but chooses to use the orthogonality property of vanishing points [16] to estimate the intrinsic parameters. Finally, Bouguet implements additional algorithms [7] to estimate the tangential distortion coefficients. Although the Camera Calibration Toolbox developed by Bouguet has become one of the most widely used tools in camera calibration among computer vision research facilities, its mode of operation is unnecessarily manual and prone to user error. One of the goals of this research, which will be discussed in the next chapter, is to automate Bouguet's toolbox.

Using a common reference, the extrinsic results for each camera in a system are compared in order to compute their relative rotation and translation. To do so, the cameras and the calibration board must remain stationary (relative to one another) while capturing each view from multiple cameras. Another goal of this research is to develop a robust multi-camera calibration algorithm that can be used to calibrate multi-camera systems with non-overlapping fields of view. Further development is presented in the next chapter.

2.5.3 Binocular Stereopsis. Using the epipolar geometry illustrated in Figure 2.16, the 3-D location of a point can be calculated by observing it from two different cameras with known rotation and translation (obtained during calibration).

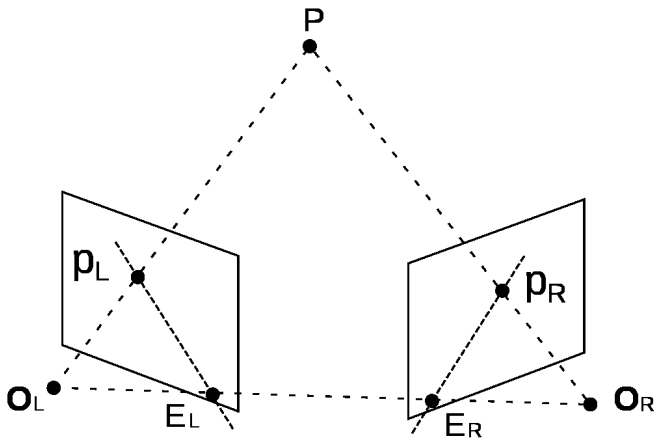


Figure 2.16: Binocular imaging geometry. The 3-D position of a point in the n-frame can be extracted from its projections onto the image planes of two cameras with known relative rotation and translation.

The epipolar geometry constraints are given by

$$\mathbf{P}^r = \mathbf{R}(\mathbf{P}^l - \mathbf{T}) \tag{2.43}$$

$$\mathbf{p}^l = \frac{f_l}{Z_l}\mathbf{P}^l \tag{2.44}$$

$$\mathbf{p}^r = \frac{f_r}{Z_r}\mathbf{P}^r \tag{2.45}$$

where $\mathbf{T}$ and $\mathbf{R}$ represent the relative rotation and translation between the left and right cameras, f_l and f_r are the respective focal lengths of each camera, the vectors $\mathbf{p}^r$ and $\mathbf{p}^l$ represent the projections of the world point $\mathbf{P}$ onto each image plane, and the vectors $\mathbf{P}^r$ and $\mathbf{P}^l$ represent the coordinates of the world point $\mathbf{P}$ on each c-frame. Starting with the left image of a point, a 1-D search along the epipolar line on the right image can be conducted in order to extract the distance to the point thus establishing its full 3-D location. Binocular imaging geometry is used extensively in determining the 3-D location of each landmark within the image-aided Extended Kalman Filter presented in Section 2.7. Additional information on epipolar geometry and binocular stereopsis can be found in Szeliski's textbook [19].

2.6 Kalman Filtering

The Kalman filter, developed by Rudolf Kalman in 1960 [10], provides a method for the optimal combination of measurements made by multiple sensors (e.g., inertial and imaging). The Kalman filter uses Bayesian statistics to combine dynamics and measurements models, which provides a solution estimate with the lowest possible uncertainty. This section outlines the basic principles behind the Kalman filter as outlined by Maybeck [13] [14].

2.6.1 Linear Kalman Filter. The physical system dynamics are modeled using the form

$$\dot{\mathbf{x}}(t) = \mathbf{F}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{G}\mathbf{w}(t) \tag{2.46}$$

where $\mathbf{x}$ is a vector containing the system states of interest, $\mathbf{u}$ is a vector containing system control inputs and $\mathbf{w}$ is a vector of white Gaussian noise sources with

$$\mathbf{E}[\mathbf{w}(t)] = \mathbf{0} \tag{2.47}$$

$$\mathbf{E}[\mathbf{w}(t)\mathbf{w}(t+\tau)] = \mathbf{Q}\delta(\tau) \tag{2.48}$$

while the matrices $\mathbf{F}$, $\mathbf{B}$ and $\mathbf{G}$ contain constant coefficients, which specify linear combinations of the vectors they multiply.

In order to implement the Kalman filter algorithm in a computer system, the continuous-time model must be discretized to account for system propagation between samples. The discrete process noise strength matrix $\mathbf{Q}_d$ and the discrete control input matrix $\mathbf{B}_d$ are obtained by changing the limits of integration to capture a single time step Δt within the general solution to the system, which is given shown by Maybeck [14] and VanLoan [21]. Additionally, the discrete state transition matrix,

which is used to propagate system states and derived from the system dynamics model, is given by

$$\mathbf{\Phi} = e^{\mathbf{F}\Delta t} \tag{2.49}$$

Linear discrete measurements from the various sensors are modeled by

$$\mathbf{z}(t_i) = \mathbf{H}\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{2.50}$$

where $\mathbf{z}$ is a vector of sensor measurements and $\mathbf{v}$ is a vector of discrete-time, white Gaussian noise sources with

$$\mathbf{E}[\mathbf{v}(t_i)] = \mathbf{0} \tag{2.51}$$

$$\mathbf{E}[\mathbf{v}(t_i)\mathbf{v}(t_j)] = \mathbf{R}\delta_{ij} \tag{2.52}$$

and

$$\mathbf{E}[\mathbf{w}(t_i)\mathbf{v}(t_j)] = \mathbf{0} \tag{2.53}$$

while the matrix $\mathbf{H}$ contains constant coefficients, which specify linear combinations of the system state vector $\mathbf{x}$. Since the system is fully linear, the Kalman filter algorithm guarantees a Minimum Mean Squared Error (MMSE) optimal solution for estimating the system states.

The quantities of interest estimated by the Kalman filter are contained within the random vector $\mathbf{x}$. The Kalman filter provides the probability density function for $\mathbf{x}$ at each discrete time step, conditioned on noise corrupted measurements provided by sensors. The Kalman filter algorithm begins with initial conditions, which include the initial state estimate vector $\hat{\mathbf{x}}$ and its uncertainty, which is contained by the covariance matrix $\mathbf{P}_{xx}$. The initial conditions are propagated from one discrete time step to the next using the discrete state transition matrix such that

$$\hat{\mathbf{x}}(t_{i+1}^-) = \mathbf{\Phi}\hat{\mathbf{x}}(t_i^+) + \mathbf{B}_d\mathbf{u}(t_i) \tag{2.54}$$

$$\mathbf{P_{xx}}(t_{i+1}^-) = \mathbf{\Phi}\mathbf{P_{xx}}(t_i^+)\mathbf{\Phi}^T + \mathbf{Q}_d \tag{2.55}$$

As linear measurements become available at discrete time intervals, the propagated state estimates and their covariance are optimally combined with the incoming measurements using the Kalman gain matrix $\mathbf{K}$, which is given by

$$\mathbf{K}(t_i) = \mathbf{P}_{xx}(t_i^-)\mathbf{H}^T[\mathbf{H}\mathbf{P}_{xx}(t_i^-)\mathbf{H}^T + \mathbf{R}]^{-1} \tag{2.56}$$

The state estimates and covariances are updated with the Kalman gain matrix from Equation (2.56) using

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)[\mathbf{z}(t_i) - \mathbf{H}\hat{\mathbf{x}}(t_i^-)] \tag{2.57}$$

$$\mathbf{P}_{xx}(t_i^+) = \mathbf{P}_{xx}(t_i^-) - \mathbf{K}(t_i)\mathbf{H}\mathbf{P}_{xx}(t_i^-) \tag{2.58}$$

As shown in Equations (2.57) and (2.58), the Kalman gain matrix serves as an optimal weighting factor that gives adequate preference to either the propagated or measured estimates, given their individual uncertainties, in order to minimize mean squared error.

2.6.2 Extended Kalman Filter. If a particular system cannot be adequately represented using linear dynamics or measurement models, the linear Kalman filter algorithm does not guarantee optimal solutions. However, in certain cases, linear approximations to nonlinear systems can still yield accurate estimates. In such cases, the Extended Kalman Filter (EKF) is used. The basic system dynamics equation for a nonlinear system is given by

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t] + \mathbf{G}(t)\mathbf{w}(t) \tag{2.59}$$

where $\mathbf{f}$ is a vector containing functions which represent the system. In turn, the nonlinear measurement equation is given by

$$\mathbf{z}(t_i) = \mathbf{h}[\mathbf{x}(t_i), t_i] + \mathbf{v}(t_i) \tag{2.60}$$

where $\mathbf{h}$ is a vector of functions which model the sensor. The main goal is to linearize nonlinear models about their nominal estimates in order to use the conventional linear Kalman update equations. To do so, the states are redefined using the perturbation model given by

$$\delta\mathbf{x}(t) \triangleq \mathbf{x}(t) - \hat{\mathbf{x}}(t) \tag{2.61}$$

where $\delta\mathbf{x}(t)$ represents the difference between the true state vector and its estimate. In order to propagate the system from initial conditions or a previous measurement to the time of the next measurement, the EKF integrates the nonlinear dynamics function over the discrete time difference using

$$\hat{\mathbf{x}}(t_{i+1}^-) = \int_{t_i}^{t_{i+1}} \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t]dt + \hat{\mathbf{x}}(t_i^+) \tag{2.62}$$

while the state covariance matrix is propagated using Equation (2.49), Equation (2.55) and a linearized dynamics model matrix $\mathbf{F}$ given by

$$\mathbf{F}(t_i) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}(t_i^+)} \tag{2.63}$$

In order to update the propagated state estimates using an incoming (possibly nonlinear) measurement, the measurement must first be predicted by evaluating the measurement model function with the most recent estimate using

$$\mathbf{z}_{pred}(t_i) = \mathbf{h}[\hat{\mathbf{x}}(t_i^-), t_i] \tag{2.64}$$

$$\delta\mathbf{z}(t_i) = \mathbf{z}(t_i) - \mathbf{z}_{pred}(t_i) \tag{2.65}$$

where $\delta\mathbf{z}$ is called the measurement perturbation and represents the difference between the actual and predicted measurements.

In order to combine the propagated and measured state estimates, the nonlinear measurement function $\mathbf{h}$ is linearized to obtain $\mathbf{H}(t_i)$ in a similar fashion to $\mathbf{F}(t_i)$ using

$$\mathbf{H}(t_i) = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}(t_i^-)} \tag{2.66}$$

The linearized matrix $\mathbf{H}$ is then used in Equation (2.56) to obtain a Kalman gain matrix, and the measurement update equation reduces to

$$\delta\hat{\mathbf{x}}(t_{i+1}^+) = \mathbf{K}(t_{i+1})\delta\mathbf{z}(t_{i+1}) \tag{2.67}$$

due to the use of perturbation state estimates and measurements. The perturbation state $\delta\hat{\mathbf{x}}$, which starts at zero during each filter recursion, is updated using Equation (2.67) and added to the nominal trajectory to produce a nominal estimate. Prior to the next recursion, the perturbation state is reset to zero. Since the EKF does not guarantee optimal solutions, it must often be tuned prior to use by adding process noise and selecting specific initial conditions. Tuning increases filter stability and usually increases solution uncertainty.

2.7 Image-aided Navigation

Veth [22] describes a novel approach to image-aided navigation in which features are tracked within an image over time in order to correct inertial navigation errors. At the same time, inertial sensor measurements are used to correct errors within the visual feature tracking algorithm. Using a so-called stochastic feature tracker, Veth presents a deeply-coupled inertial and visual navigation algorithm that uses rigorous stochastic developments in order to integrate the two systems.

Table 2.1: Image-aided navigation parameters. This table summarizes the major parameters in Veth's image-aided navigation algorithm [22].

Parameter	Description
$\mathbf{p}^n$	Vehicle position in navigation frame
$\mathbf{v}^n$	Vehicle velocity in navigation frame
$\mathbf{C}_b^n$	Vehicle body to navigation frame DCM
$\mathbf{a}^b$	Accelerometer bias vector
$\mathbf{b}^b$	Gyroscope bias vector
$\mathbf{t}_m^n$	Location of landmark m in the navigation frame
$\mathbf{d}^b$	Camera-to-body lever arm in body frame
$\mathbf{C}_c^b$	Camera-to-body orientation DCM in body frame
$\mathbf{y}$	Vector of landmark locations in navigation frame

2.7.1 Algorithm Description. The major system parameters involved in the algorithm are listed in Table 2.1. Veth uses an EKF to periodically update the error estimates in the system parameters, which can be grouped into navigation parameters (position, velocity, attitude), inertial sensor biases and a vector ($\mathbf{y}$) describing the location of the landmarks of interest. Throughout the EKF, navigation parameters are calculated using bias-corrected inertial measurements (vehicle velocity and angular increment) and used to propagate the error estimates through the strapdown mechanization equations described by Titterton and Weston [20]. The navigation parameters are modeled as stochastic random processes while the inertial sensor biases are modeled as first-order Gauss-Markov processes, which are based on manufacturer specifications. Finally, the landmarks used in visual feature tracking are modeled as stationary objects with respect to the Earth, with a small amount of process noise added for filter stability. Figure 2.17 illustrates Veth's overall algorithm.

2.7.2 Landmark Track Maintenance. Due to computer limitations and online implementation practicalities, Veth constrains the number of features that are tracked at any particular time through the use of a track maintenance algorithm. In general, a maximum and minimum number of tracks is set, and features are constantly "pruned" in order to provide the EKF the best information possible. In his

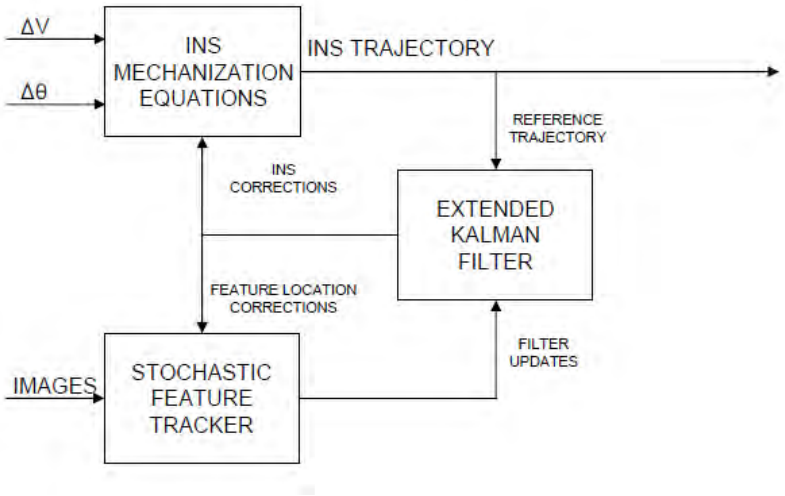


Figure 2.17: The 3-D locations of stationary features are tracked and used to provide measurement updates in order to estimate and correct inertial navigation errors. In turn, the inertial navigation system is used to support feature tracking [22].

algorithm, Veth uses SIFT to generate features of interest because they are easy to identify, locally distinct from other features and well separated in image-space. The main purpose of the track maintenance algorithm is to remove "stale" tracks when no correspondence is found for extended periods of time.

2.7.3 Measurement Model. One of the most crucial developments in Veth's algorithm [22] is the use of rigorous stochastic projections to propagate features between images. Veth describes a tracking loop algorithm responsible for incorporating new landmark tracks, predicting and matching feature locations between images through the use of stochastic projections and providing actual filter measurements to the EKF. In turn, the EKF assists the tracking loop by providing mean-square error state estimates from which stochastic projections are derived.

The tracking loop incorporates new tracks (as necessary) by estimating the 3-D location of the feature using a combination of binocular stereopsis and the current

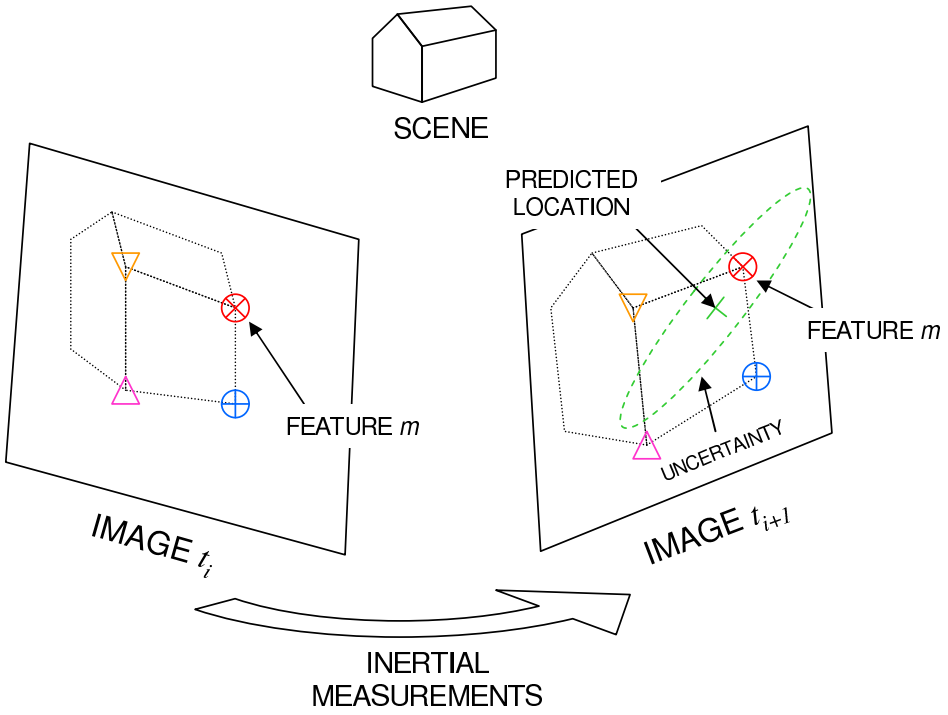


Figure 2.18: Features of interest are propagated into future images using inertial measurements and stochastic projections [22].

navigational state vector. The vector containing feature locations, along with its associated covariance matrix, is augmented into the EKF using the system's stochastic properties [23]. As the EKF propagates the state estimate, the location of the features in a future image are predicted along with an uncertainty ellipse, which is derived from the propagated covariance matrix. Finally, the landmark correspondence search can be restricted based upon the *pix*-frame projection of each landmark's location uncertainty in order to speed up the tracking process. Once (and if) a match is found, the pixel location of the matched features is used to update the navigation state. Figure 2.18 illustrates the stochastic projection process.

2.8 Summary

This chapter has laid the basic foundation upon which this thesis is based. Topics including nomenclature, reference frames, inertial navigation, digital image processing, camera calibration, Kalman filtering and image-aided navigation have been

explored. The next chapter of this research describes how these concepts were used to develop an improved image-aided navigation algorithm that includes automatic camera calibration and accounts for affine distortions such as those presented by cameras with non-overlapping fields of view.

III. Methodology

This chapter presents the methods used to generate a solution to the problems described in Chapter I, as well as the experiments used to evaluate the solution. This chapter is divided into two sections: algorithm development and experimental methods. The algorithm development for automatic calibration is presented in Section 3.1.1. The affine distortion prediction process is presented in Section 3.1.2. Finally, the experimental methods used to validate the major research contributions of this thesis are presented in Section 3.2.

3.1 Algorithm Development

3.1.1 Automatic Calibration . This section describes the procedures used to generate an automated calibration algorithm based on the manual calibration processes described in Section 2.5. The automatic calibration algorithm developed in this section removes the need for user interaction and complements the existing Matlab® toolbox.

As previously stated, the current calibration process relies heavily on user involvement. Although current computer vision algorithms are very efficient at recognizing corners within an image [6], there are currently no mature methods for separating a standard calibration board, such as the one in Figure 1.2, from the image background. Because of this, the current calibration process starts off by a manual delineation of the calibration board perimeter for every calibration image. Additionally, the user is asked for the size of each square and the number of squares along each axis. As the user defines the board perimeter within each image, the toolbox builds a set of coordinate correspondences between points in the *pix*-frame ($\mathbf{m}$) and points in the *cal*-frame ($\mathbf{M}$). Once enough correspondences are established, the toolbox can run through a gradient descent algorithm to find the best solution for

$$\underline{\mathbf{m}} = \mathbf{H}\underline{\mathbf{M}} \tag{3.1}$$

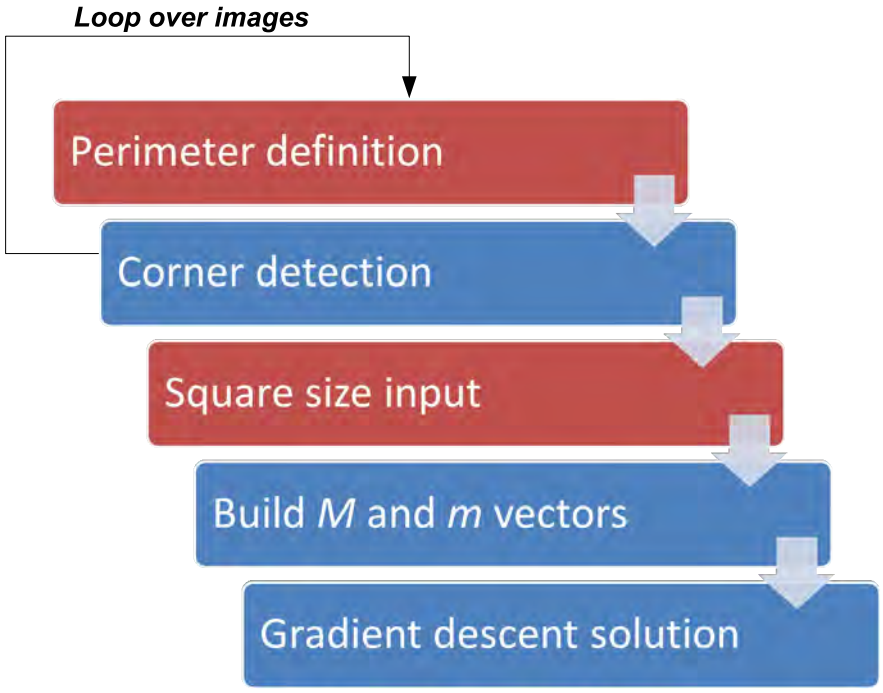


Figure 3.1: Standard (manual) calibration process. During standard calibration, the user is heavily involved in the board delineation and square size input steps.

where the homography matrix **H** contains the camera model and transformation matrix that takes points from the *cal*-frame into the *pix*-frame.

As illustrated by Figures 3.1 and 3.2, the manual steps from the standard algorithm (color-coded red) are replaced with proven automated computer functions (color-coded blue) in order to eliminate the need for user interaction. First, the standard calibration board is replaced by an arbitrary image for which distinct SIFT features can be found and tracked. SIFT matches can then be automatically found between the physical board and a digital copy with no need for user interaction. A sample feature match set from an experimental calibration is shown in Figure 3.3.

Having automatically found the *pix*-frame points **m** on the right image, their corresponding *cal*-frame coordinates can be computed through a transformation ma-

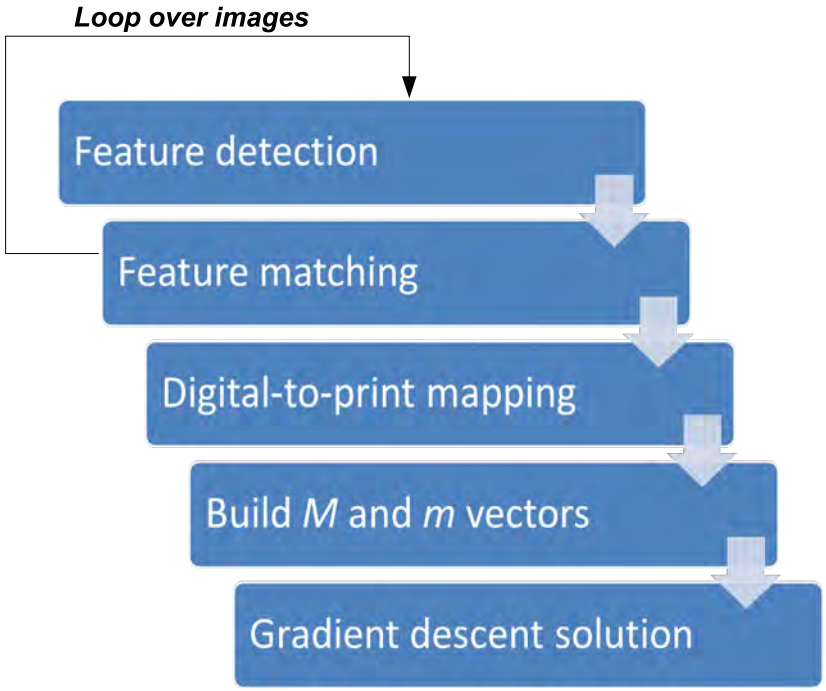


Figure 3.2: Automatic calibration process. During automatic calibration, user involvement is drastically reduced by changing the calibration board and replaced by automatic feature recognition.

trix $\mathbf{A}_{pix}^{print}$ such that

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.2}$$

$$\mathbf{M} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{M}} \tag{3.3}$$

$$\mathbf{M} = \mathbf{A}_{pix}^{print} \hat{\mathbf{M}} \tag{3.4}$$

where the set of points $\hat{\mathbf{M}}$ have coordinates which are measured in pixels and the set of points $\mathbf{M}$ are measured in meters, as required by the calibration toolbox. It is important to note that since all *cal*-frame points are coplanar, their third coordinate

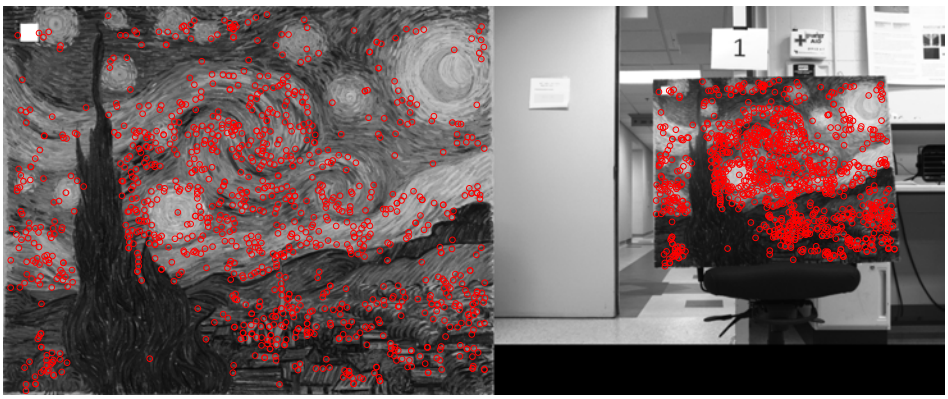


Figure 3.3: Automatic calibration matches. Board delineation and corner detection are replaced by automatic SIFT feature matching between an image printout and its digital copy. The left image shows digital *cal*-frame coordinates $\hat{\mathbf{M}}$, while the right image shows the corresponding *pix*-frame points $\mathbf{m}$.

can be assumed to be zero ($z = 0$), which reduces the transformation matrix from three to two dimensions. The coefficients that populate the transformation matrix in Equation (3.2) are computed by comparing the pixel and meter coordinates of at least three known points in a least-squares algorithm. The corners of the white square shown in Figure 3.4 illustrate four known points selected in both the digital board and the printed board.

Having defined the $\mathbf{M}$ and $\mathbf{m}$ vectors without user interaction, the standard calibration toolbox can be used to execute the gradient descent computations and complete the calibration process. The results from an automatic calibration routine are compared to the standard calibration process in terms of total time and reprojection error in Chapter IV.

3.1.2 Affine Distortion Prediction . This section describes the methods and algorithms used to develop a predictive algorithm that accounts for affine distortions in SIFT feature descriptors, which arise from changes in 3-D viewpoint. The predictive algorithm, referred to from here on as Affine Distortion Prediction (ADP), is

Figure 3.4: Pixel-to-board frame mapping. In order to convert digital *cal*-frame coordinates (pixels) to actual printed *cal*-frame coordinates (meters) for all automatic SIFT matches found, a transformation matrix is constructed by comparing the digital and printed *cal*-frame coordinates of four known points (corners of the white square).

then deeply coupled into the existing Image-aided Extended Kalman Filter (IAEKF), originally developed by Veth [22].

3.1.2.1 Effects of Affine Distortion on Feature Matching. In order to fully understand the need for, and advantages provided by ADP, the effects of affine distortions in feature tracking effectiveness must be examined. As previously stated, one of the key requirements in image-aided navigation is the ability to detect and track features over many frames of an image sequence. Although SIFT descriptors allow feature matching through changes in scale (zoom), rotations about the c-frame's z-axis (2-D rotations), and illumination, feature tracking effectiveness diminishes with rotations about the c-frame's x and y axes (3-D rotations) [18]. Figure 3.5 illustrates a 3-D rotation between two images of the same scene. Using the conventional SIFT descriptors found for both images and a NNDR of 0.45, there was only one positive match found. Next, Figure 3.6 illustrates a drastic increase in SIFT matches between the two images when an affine transformation is artificially applied to the original flat image. Finally, Figure 3.7 illustrates the locations of the matches found in Figure 3.6 backprojected onto the original flat image to produce the final match set.

3.1.2.2 Simulating Affine Distortions. Having established the potential improvement offered by simulating affine distortions, the mathematics involved can be developed. The process begins with a flat, grayscale digital image $\mathbf{I}$ with height M and width N. The grayscale value stored in the (i, j) element of the matrix $\mathbf{I}$ can be thought of as a 2-D point with coordinates $(j+1, i+1)$ in the *pix*-frame. An affine distortion to $\mathbf{I}$ can be simulated by applying a 3-D DCM $\mathbf{A}_I^{\hat{I}}$ about the image center to every element in $\mathbf{I}$ such that

$$\begin{bmatrix} \hat{j}+1 \\ \hat{i}+1 \\ \hat{k} \end{bmatrix} = \mathbf{A}_I^{\hat{I}^T} \left(\begin{bmatrix} j+1 \\ i+1 \\ 0 \end{bmatrix} - \begin{bmatrix} N/2 \\ M/2 \\ 0 \end{bmatrix} \right) + \begin{bmatrix} N/2 \\ M/2 \\ 0 \end{bmatrix} \tag{3.5}$$

where the new elements $(\hat{i}, \hat{j})$ compose the simulated image $\hat{\mathbf{I}}$, which is the resulting affine-distorted version of $\mathbf{I}$. Note that in order to apply a 3-D transformation matrix onto a 2-D image, a third dimension must be created for every element in $\mathbf{I}$. Since all the elements of $\mathbf{I}$ are assumed to be coplanar, a third coordinate of $z = 0$ can be set for all points in the image prior to applying $\mathbf{A}_I^{\hat{I}}$. The DCM $\mathbf{A}_I^{\hat{I}}$ can be built from desired changes in line-of-sight azimuth (ψ) and elevation (θ) using Equation (2.2). Since SIFT descriptors are already invariant to rotations about the line-of-sight vector, ϕ is always set to zero.

Simulating an affine distortion to the original image is essentially equivalent to resampling $\mathbf{I}$ along the major axes of $\mathbf{A}_I^{\hat{I}}$. Therefore, the final step in this process consists of applying a low-pass 2-D Gaussian filter on $\hat{\mathbf{I}}$ to prevent undesired aliasing. A 2-D low-pass Gaussian filter can be applied to an image $\mathbf{X}$ using

$$\mathbf{X}_{filt} = \mathbf{X} \circledast h(n_1, n_2) \tag{3.6}$$

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n1} \sum_{n_2} h_g} \tag{3.7}$$

$$h_g(n_1, n_2) = e^{-(n_1^2 + n_2^2)/(2\sigma^2)} \tag{3.8}$$

where n_1 and n_2 represent the dimensions of the Gaussian filter and σ represents its standard deviation. Figure 3.8 illustrates the simulated affine distortion output image for four different combinations of change in line-of-sight azimuth and elevation.

3.1.2.3 Transforming SIFT Descriptors. With a method for simulating affine distortions in place, a process for transforming the SIFT descriptor of a particular feature can be developed. Transforming the SIFT descriptor of a specific feature differs from the process shown in Section 3.1.2.1 and Figure 3.6. Specifically, one is no longer simulating an affine distortion on the input image and checking for all possible matches, but rather trying to obtain a new (affine-distorted) descriptor for a single feature. The reason for this difference is that in order to integrate ADP into the IAEKF, SIFT feature descriptors need to be "propagated" to account for changes in relative viewpoint between the vehicle and the features it tracks.

The entire SIFT descriptor transformation process is illustrated in Figure 3.9. Again, the process begins with a flat input image $\mathbf{I}$, a feature of interest f_m, which is described by *pix*-frame coordinates (x_m, y_m), SIFT scale σ_m, SIFT rotation r_m and SIFT descriptor d_m. The goal is to use a desired change in azimuth ($\Delta\phi$) and elevation ($\Delta\theta$) in order to transform d_m and produce $\hat{d_m}$.

The first step is to simulate the affine distortion dictated by $\Delta\phi$ and $\Delta\theta$ on $\mathbf{I}$ using Equation (3.5). Since the affine distortion simulation equation applies to all points, it can be used to project the coordinates of f_m $(x_m, y_m, 0)$ along with the rest of the image. The affine distortion simulation process significantly alters the scale and rotation of the entire image, which changes both r_m and σ_m. This change makes it impossible to manually compute a new SIFT descriptor from $\hat{\mathbf{I}}$, which is the affine-distorted version of $\mathbf{I}$, at $\hat{f_m}$, which is the projected location of f_m in the new image. Instead, the SIFT algorithm described in Section 2.4.2 is applied to $\hat{\mathbf{I}}$ to produce a new set of SIFT features $\mathbf{G}$. Finally, the nearest neighbor in $\mathbf{G}$ to $\hat{f}_m$ is found using Euclidean distances and called g_n. It is then assumed that if the threshold for accepting a close neighbor match is low enough, $\hat{d}_m$ can be taken directly from the new

feature. That is, the "transformed" descriptor for f_m is actually the descriptor from the new feature g_n. If a new feature is not found within the established threshold, the SIFT descriptor is said to be non-transformable for the given $\Delta\phi$ and $\Delta\theta$ and the original descriptor is returned. Figures 3.10 through 3.12 illustrate the descriptor transformation algorithm described in this section.

3.1.2.4 Deep Coupling into IAEKF. Having developed an algorithm for transforming SIFT descriptors using desired changes in azimuth and elevation, ADP can now be integrated into the IAEKF in order to improve long-baseline feature tracking performance. The first step involves understanding the nature of changes in visual information through vehicle movement. As shown in Figure 3.13, assuming a forward-moving trajectory, visual elements which initially are characterized by high affine and projective distortions become "flatter" as the vehicle approaches. That is, in general, additional information about particular objects is gained in a scene as one gets closer to them.

A similar characteristic can be attributed to the affine distortion simulation algorithm described in 3.1.2.2. Note that affine distortion works by resampling initial visual information, which actually reduces the amount of information, to approximate the effects of high affine change. Affine distortion simulation can only distort currently available visual information, and more importantly, it cannot recover missing information. In other words, affine distortion simulation cannot be used to "flatten" images that are already distorted; it can only further distort images.

From the above observations, one can conclude that in order to use ADP effectively, a SIFT descriptor cannot be simply propagated forward because the visual information, which was hidden by the high affine distortion, cannot be recovered. Instead, newer measurements (sets of descriptors) must be propagated backwards. Therefore, newer descriptors, which are assumed to be "flatter" and contain more information, need to be transformed to match current descriptors, which are assumed to

be distorted and contain less information. Figure 3.14 illustrates the visual operator space problem described in this section.

Having established the correct application of ADP onto SIFT descriptors, the mechanics involved in the deep coupling with the IAEKF can be finally set. Figure 2.17 describes the baseline deep coupling algorithm described by Veth [22]. Integrating ADP into the existing visual aided navigation algorithm can be thought of as not only predicting where (in the *pix*-frame) features will appear from t_i to t_{i+1}, but also what (in terms of SIFT descriptor changes) those feature will look like at t_{i+1}.

The algorithm begins with the binocular initialization of K landmarks at t_i. As previously mentioned, the binocular initialization is necessary for estimating feature depth and is only needed when discovering new landmarks. After initializing a particular landmark m and estimating its 3-D location in the n-frame $\mathbf{t}_m^n$, the landmark's initial descriptor $d_m(t_i)$ and relative azimuth $\psi_m(t_i)$ and elevation $\theta_m(t_i)$ are recorded. The landmark's relative azimuth and elevation are computed using

$$\mathbf{l}_{vm}^n = \mathbf{t}_m^n - \mathbf{p}^n \tag{3.9}$$

$$\psi = tan^{-1}\left(\frac{-\mathbf{l}_{vm}^n(3)}{\sqrt{\mathbf{l}_{vm}^n(1)^2 + \mathbf{l}_{vm}(2)^2}}\right) \tag{3.10}$$

$$\theta = tan^{-1}\left(\frac{\mathbf{l}_{vm}^n(2)}{\mathbf{l}_{vm}^n(1)}\right) \tag{3.11}$$

where the n-frame vector $\mathbf{l}_{vm}^n$ is a line-of-sight vector pointing from the vehicle to the landmark m, and the vectors $\mathbf{t}_m^n$ and $\mathbf{p}^n$ represent the landmark and vehicle positions in the n-frame respectively. In short, relative azimuth is the angle between the projection of $\mathbf{l}_{vm}^n$ onto the N-E plane and the N-axis, while relative elevation is the angle between $\mathbf{l}_{vm}^n$ and the N-E plane. Additionally, the computed relative azimuth is modified from Equation (3.10) by adding or subtracting π to ensure continuity across the entire 2π circle. Figure 3.15 illustrates the azimuth and elevation angles as defined by the n-frame. Note that relative azimuth and elevation are purely functions of the

relative position between the vehicle and the features it tracks; vehicle attitude is not a factor.

Next, using the mathematics described by Veth [22], each of the K landmarks is propagated forward in time from t_i to t_{i+1}. The pix-frame location of each landmark is estimated using INS measurements. A stochastically-defined elliptical search area around each estimated landmark location is created and based on state covariances as shown in Figure 1.3. Before attempting to match candidate features from the image at t_{i+1} (referred to as measurements) to landmark descriptors from t_i, the ADP algorithm is used to propagate $d_m(t_i)$ to $d_m^-(t_{i+1})$ for all K landmarks. In order to do so, the most recent computed position solution is used to calculate $\hat{\psi}_m(t_{i+1}^-)$ and $\hat{\theta}_m(t_{i+1}^-)$, for each of the K landmarks being tracked.

Having calculated the new relative azimuth and elevation for the m^{th} landmark, $\Delta\psi_m$ and $\Delta\theta_m$ are computed using

$$\Delta\psi_m = \hat{\psi}_m(t_{i+1}^-) - \hat{\psi}_m(t_i^+) \tag{3.12}$$

$$\Delta\theta_m = \hat{\theta}_m(t_{i+1}^-) - \hat{\theta}_m(t_i^+) \tag{3.13}$$

The SIFT descriptor transformation algorithm described in Section 3.1.2.3 is then used to back-propagate all candidate features from t_{i+1} to t_i as illustrated in Figure 3.14 while keeping track of their corresponding original SIFT descriptors. Finally, a NNDR algorithm is used to identify a match between the particular landmark and all back-propagated candidate features that fall within the search window. If a positive match is found, $d_m^-(t_{i+1})$ is then defined as the transformed SIFT descriptor that was positively matched while $d_m^+(t_{i+1})$ is defined as the corresponding original descriptor. This process is repeated for all K landmarks active at t_i. The entire ADP algorithm is illustrated in Figure 3.16.

3.1.2.5 Optimizing Simplifications. Not all magnitudes of $\Delta\psi_m$ and $\Delta\theta_m$ warrant the use of ADP. There is a range of change in either relative azimuth or elevation that can be neglected due to its minimal effect on SIFT descriptors. This range was experimentally found to be $0 - 35$ degrees in both. Additionally, it was experimentally established that distortion simulations exceeding 60 degrees in either azimuth or elevation result in SIFT descriptor transformation success rates below 5% and therefore can be neglected.

The ADP algorithm is repeated K times every discrete time step, often with very similar affine distortions. Therefore, measurement back-projections involving similar $\Delta\psi$ and $\Delta\theta$ combinations for a particular time step can be grouped into a single back-projection mode in order to eliminate computational redundancy. Figure 3.17 illustrates a sample back-projection set $(\Delta\psi, \Delta\theta)$ and the modes they were grouped into. Since back-projection involves transforming an image in accordance with a particular $(\Delta\psi, \Delta\theta)$, many features may be back-projected using the same transformed image.

3.1.2.6 Additional Enhancements to the IAEKF. Finally, several important enhancements were implemented into the existing IAEKF software:

Side Landmark Manager: In order to track landmarks over extended periods of time using side cameras, landmark coast time, which is the amount of time a landmark is allowed to remain active in the absence of positive matches, has to be lengthened so that landmarks remain active while they transition between front and side cameras. However, a longer landmark coast time increases the wait time required to replace a stale track. This is exacerbated under conditions in which the forward-looking scene changes quickly (e.g., during horizontal turns). Ultimately, a longer coast time increases the probability of losing all landmark tracks without replacement, which effectively eliminates the measurement updates provided by the feature tracker to the IAEKF. This conflict between landmark coast time and feature tracker effectiveness was solved by

adding a second feature tracker, which receives landmark hand-offs from the standard feature tracker as it deactivates stale features. The second feature tracker allows newly deactivated landmarks to still provide measurement updates, if a positive match is made through a side camera, while maintaining a relatively short landmark coast time.

SIFT Descriptor Update: In the original implementation of the IAEKF, the feature tracker used the SIFT descriptors from initial landmark activations to determine matches. If a positive match was found, the SIFT descriptor was not replaced by its match, which is by definition newer. This process was changed so that SIFT descriptors for matched landmarks are replaced by their latest match. This frequent SIFT descriptor update increases the probability of tracking landmarks over extended periods of time since it reduces the relative change between reference and candidate SIFT descriptors. Additionally, this frequent descriptor update can be used to account for descriptor changes arising from small incremental changes in viewpoint.

Lens Distortion Corrections: Finally, the original IAEKF implementation only accounted for lens distortion effects when correcting landmark locations on the *pix*-frame. However, lens distortion was not used to account for changes in feature appearance as a function of *pix*-frame location. It is logical to assume that the SIFT descriptor for a particular feature changes significantly depending on the feature's location within the *pix*-frame due to radial lens distortion effects. Therefore, in order to track features for extended periods of time, and more importantly, as they change locations within the *pix*-frame, all incoming measurement images were undistorted using the camera model from calibration prior to detecting SIFT features. This image undistortion step increases the probability of obtaining a positive landmark match even as the landmarks change position within the *pix*-frame.

3.2 Experimental Methods

This section describes the experimental methods used to evaluate the automatic calibration and affine distortion prediction algorithms developed in Section 3.1. The topics discussed include test equipment, algorithm variables, sensor models and data collection run descriptions.

3.2.1 Test Equipment. The main equipment used to experimentally verify both algorithms includes a set of 3 consumer-grade computer-vision cameras from Prosilica and the MEMS-grade MIDG II IMU from Microbotics. Additionally, portable power and processing (computer) hardware was necessary to mobilize the test setup and collect outdoor data. Finally, the tactical-grade SPAN INS/GPS system from Novatel was used in conjunction with the main test equipment in order to collect high-fidelity truth data for error computations. Figures 3.18 and 3.19 illustrate the two collection rigs used during this research. The automatic calibration algorithm was tested by calibrating a pair of consumer-grade cameras using the automatic algorithm and comparing the results with Bouguet's standard method. The ADP algorithm was tested by collecting outdoor inertial and image data using the two collection rigs and comparing the IAEKF output results with the high-fidelity truth source.

3.2.2 Algorithm Variables. In order to properly test the ADP algorithm, the standard variables needed to run the IAEKF were tuned. Table 3.1 summarizes all the variable values needed to repeat the experiments performed for this research. The standard variables used in Veth's code [22] as well as newly-added ADP variables are included. The algorithm variables are described below.

Coast time : Time an active track is allowed to remain stale.

Max landmarks : Maximum amount of tracks allowed at any moment.

ADP groups : Minimum, maximum and separation of ADP mode groups.

Table 3.1: Variable values for IAEKF algorithm. This table summarizes the variable values associated with the IAEKF implementation used for this research.

Variable	Value
Coast time (front)	1 [s]
Coast time (side)	3 [s]
Max landmarks (front)	30
Max landmarks (side)	30
ADP groups	± 0:5:60 [deg]
ADP threshold	40 [deg]
Minimum feature scale	3
Minimum correlation	0.95
SIFT match NDDR	0.45
Pixel error STD	2 [pix]
SIFT rotation error STD	0.5 [pix]
SIFT scale error coefficient	0.05 [pix]
Radial error coefficient	0.005 [pix]
Stochastic constraint distance	2.15 [pix]

ADP threshold : Minimum viewpoint change in an ADP group for algorithm usage.

Minimum feature scale : Minimum SIFT feature scale to accept a new track.

Minimum correlation : Minimum descriptor correlation to accept match.

SIFT match NNDR : NNDR used to accept feature matches in the IAEKF.

Pixel error STD : Pixel error standard deviation in measurement model.

SIFT rotation error coefficient : Rotation error multiplier in measurement model.

SIFT scale error coefficient : Scale error multiplier in measurement model.

Radial error coefficient : Radial distortion error multiplier in measurement model.

Stochastic constraint distance : Baseline distance for feature match search area.

3.2.3 Sensor Models and Installation. An accurate INS model was developed, based on manufacturer specifications and experimental data, in order to adequately propagate the EKF states during processing. Table 3.2 summarizes the main

Table 3.2: MIDG II MEMS-grade INS model. This table summarizes the MIDG II INS model variables used for this research.

Variable	**Value**
Sampling frequency	50 [Hz]
Accelerometer time constant	3600 [s]
Gyroscope time constant	3600 [s]
Accelerometer bias STD	0.2 [m/s^2]
Gyroscope bias STD	9×10^{-3} [rad/s]
Accelerometer scale factor	300 [ppm]
Gyroscope scale factor	150 [ppm]
Accelerometer random walk STD	9×10^{-3} [m/s/$\sqrt{\text{s}}$]
Gyroscope random walk STD	9×10^{-4} [rad/$\sqrt{\text{s}}$]

characteristics of the INS model used during this research. Table 3.3 summarizes the standard 15-state Pinson error model [20], which was used in conjunction with the INS model in the IAEKF. Finally, the automatic calibration algorithm described in this research was used to produce camera models for each camera as summarized in Table 3.4 as well as the extrinsic sensor installation measurements summarized in Tables 3.5 and 3.6. However, as it will be later pointed out in Section 5.2.1, the automatic camera calibration algorithm does not currently compute IMU-to-camera extrinsic configurations. The extrinsic configurations computed by the algorithm are all relative to Camera 1 (front left). Therefore, the IMU-to-camera lever arm was manually measured for Camera 1 and the two sensors were assumed to be perfectly aligned in terms of DCMs.

Table 3.3: Kalman filter state definitions. This table summarizes the standard 15-state Pinson error model [20].

State	**Variable**	**Meaning**
x_1	δp_N	North position error [m]
x_2	δp_E	East position error [m]
x_3	δp_D	Down position error [m]
x_4	δv_N	North velocity error [m/s]
x_5	δv_E	East velocity error [m/s]
x_6	δv_D	Down velocity error [m/s]
x_7	$\delta\alpha$	North attitude error [mrad]
x_8	$\delta\beta$	East attitude error [mrad]
x_9	$\delta\gamma$	Down attitude error [mrad]
x_{10}	$\delta f x_s$	x accelerometer bias [m/sec^2]
x_{11}	$\delta f y_s$	y accelerometer bias [m/sec^2]
x_{12}	$\delta f z_s$	z accelerometer bias [m/sec^2]
x_{13}	$\delta\omega x_s$	x gyro drift [rad/sec]
x_{14}	$\delta\omega x_s$	y gyro drift [rad/sec]
x_{15}	$\delta\omega x_s$	z gyro drift [rad/sec]

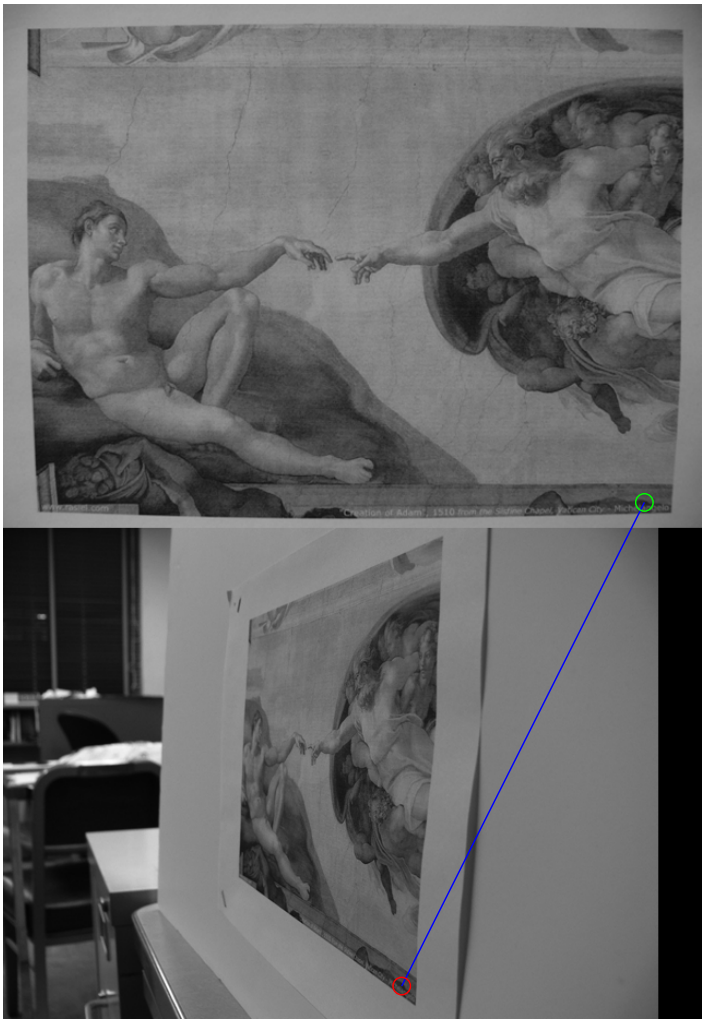


Figure 3.5: SIFT matches through large affine change. Although SIFT descriptors allow for feature matching through changes in illumination, 2-D rotation, and zoom, their matching effectiveness quickly decreases during 3-D rotations. SIFT finds one match between the two images.

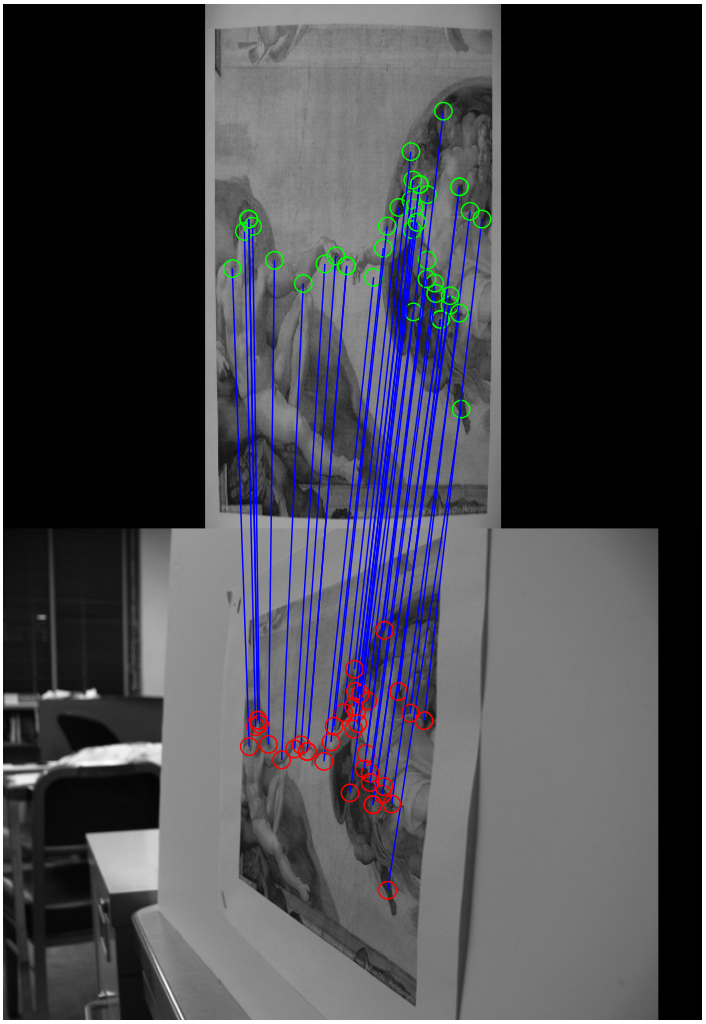


Figure 3.6: SIFT matches with simulated affine distortion. If a 3-D rotation approximating the right image is applied to the original left image, the number of SIFT matches drastically increases to 40.

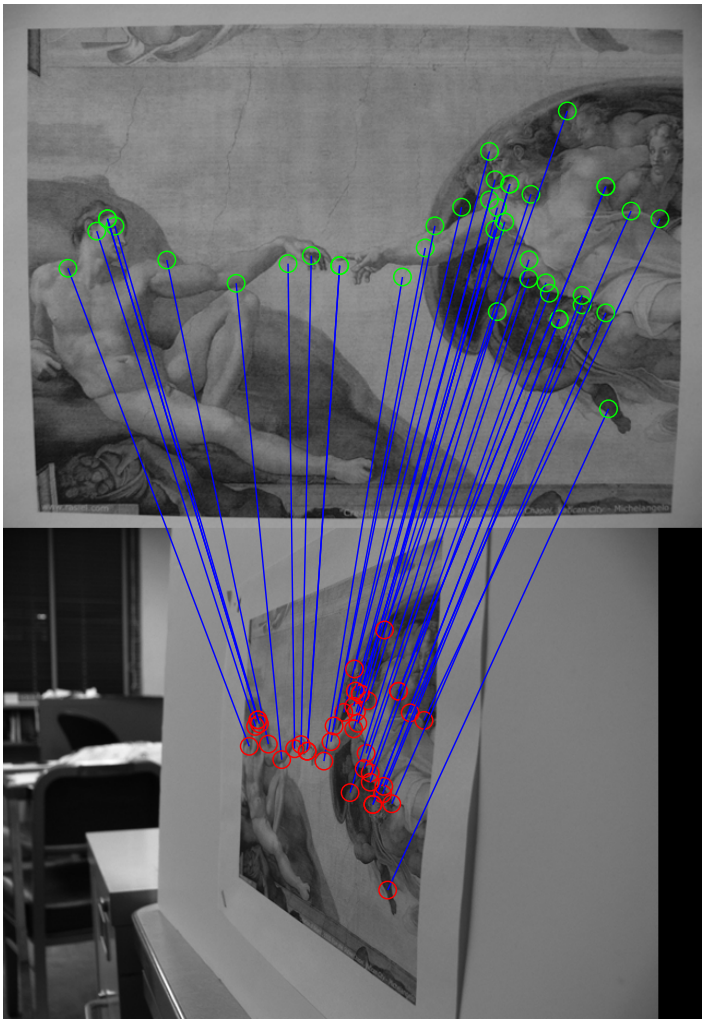


Figure 3.7: Backprojection of match locations onto original image. Finally, the match locations from the simulated image can be backprojected onto the original flat image to recover the final match locations between the two original images.

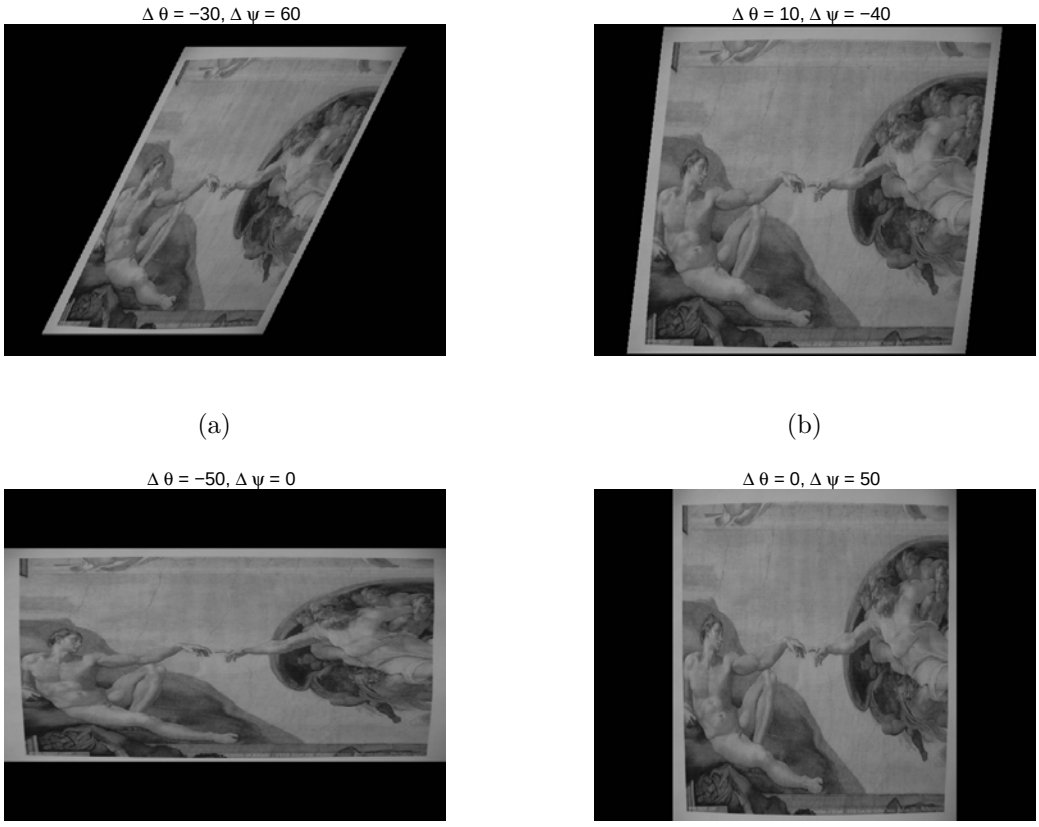


Figure 3.8: Sample affine distortion output images using various combinations of desired change in azimuth and elevation. The input image is essentially resampled along the major axes of the 3-D DCM $\mathbf{A}_I^{\hat{I}}$.

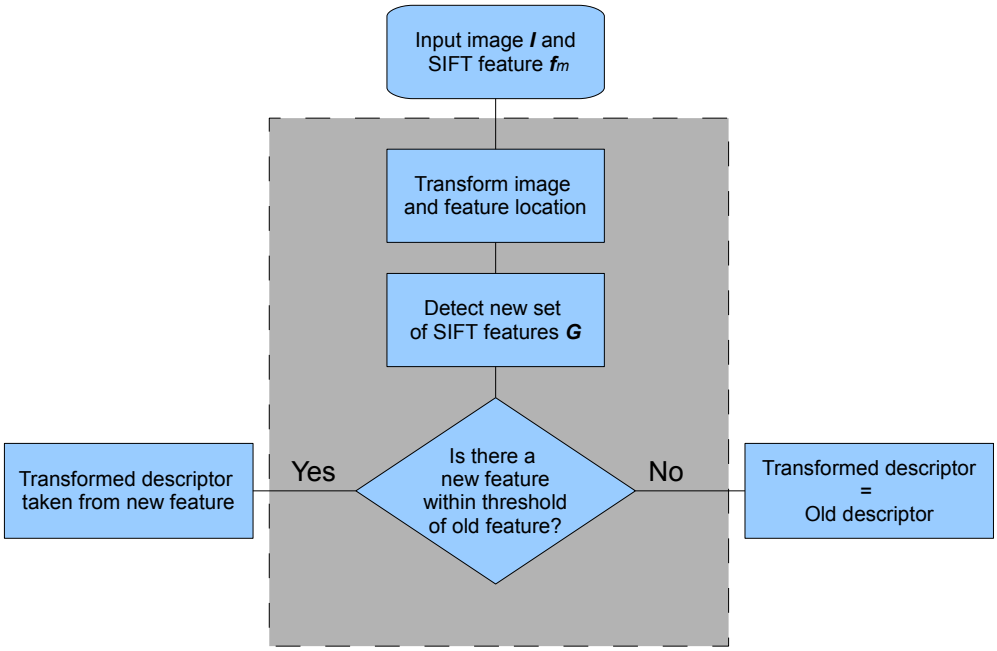


Figure 3.9: SIFT descriptor transformation algorithm.

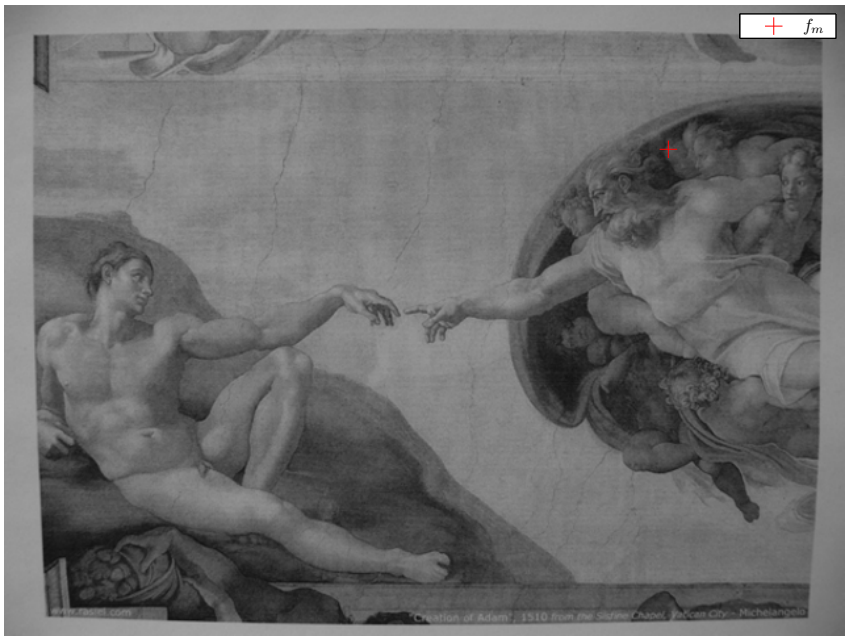


Figure 3.10: Sample flat image $\mathbf{I}$ with selected SIFT feature f_m.

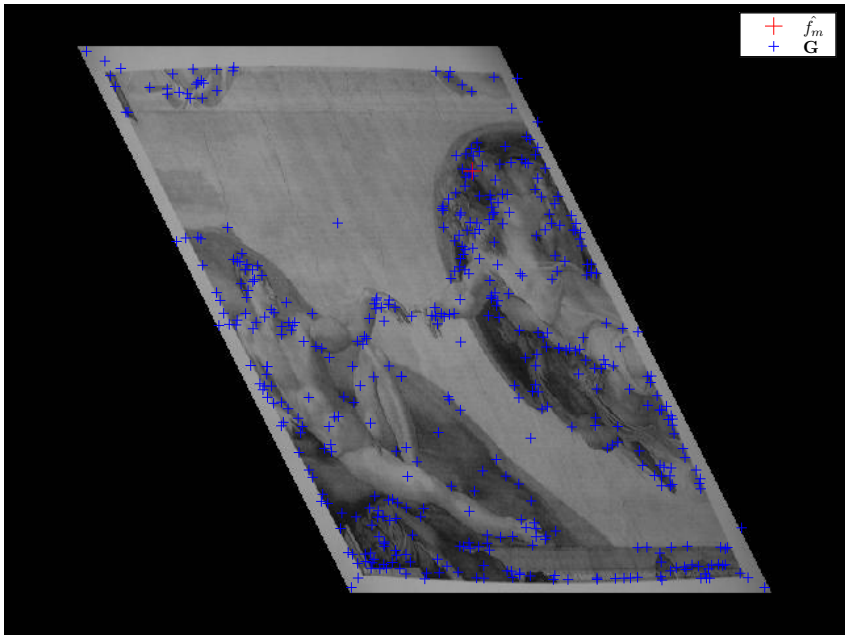


Figure 3.11: Sample affine-distorted image with new SIFT features. Since the image scale and rotation have changed, new SIFT descriptors cannot be manually computed at the exact feature location. Instead, the entire SIFT algorithm is used to re-detect a new set of features ($\mathbf{G}$) within the new image.

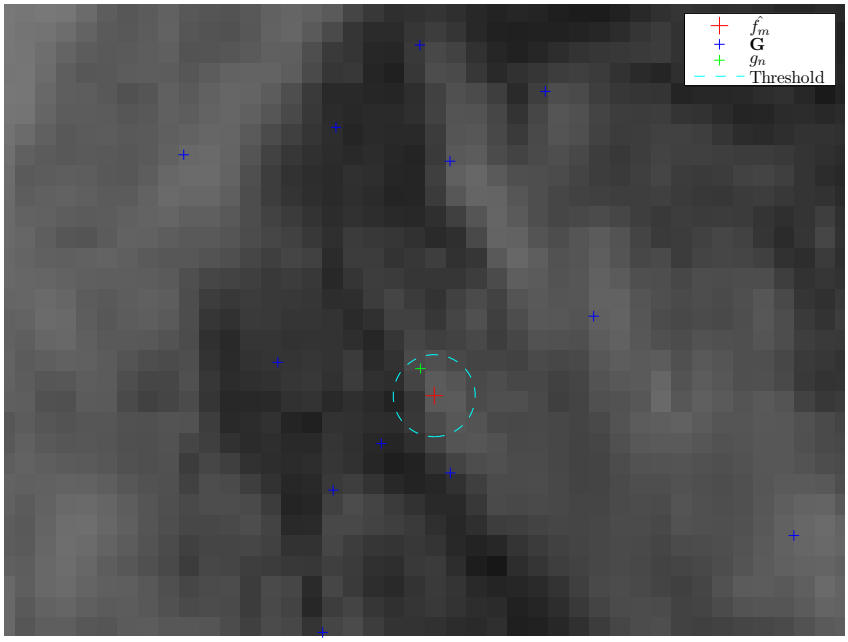


Figure 3.12: Sample nearest neighbor feature match. The SIFT descriptor for the closest new feature g_n to $\hat{f}_m$ within the predefined threshold (2 pixels in this case) is chosen as the "transformed" descriptor $\hat{d}_m$.

Figure 3.13: Nature of visual information gain over time. Initially, there is very little visual information available about the face of the object parallel to the travel path (top). As the vehicle continues on a straight and forward trajectory, the amount of visual information available increases (bottom). Affine distortion simulation is capable of taking the bottom image and approximating the top image but not vice-versa.

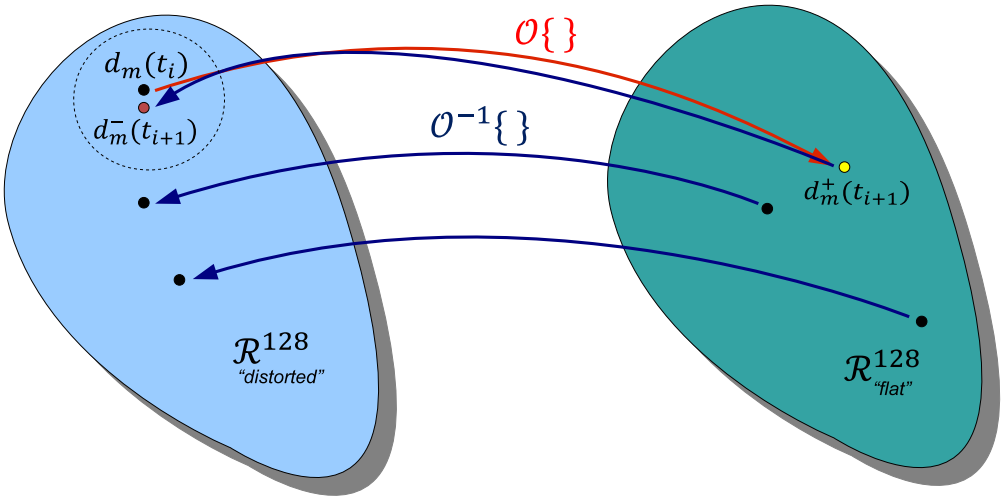


Figure 3.14: Illustration of descriptor propagation operator. Since the descriptor operator $\mathcal{O}$ cannot be applied to current descriptors, all incoming measurements are back-propagated using ADP ($\mathcal{O}^{-1}$). Descriptor matching is done on the left ("distorted" $\mathcal{R}^{128}$) and the nearest match is followed back to the right ("flat" $\mathcal{R}^{128}$).

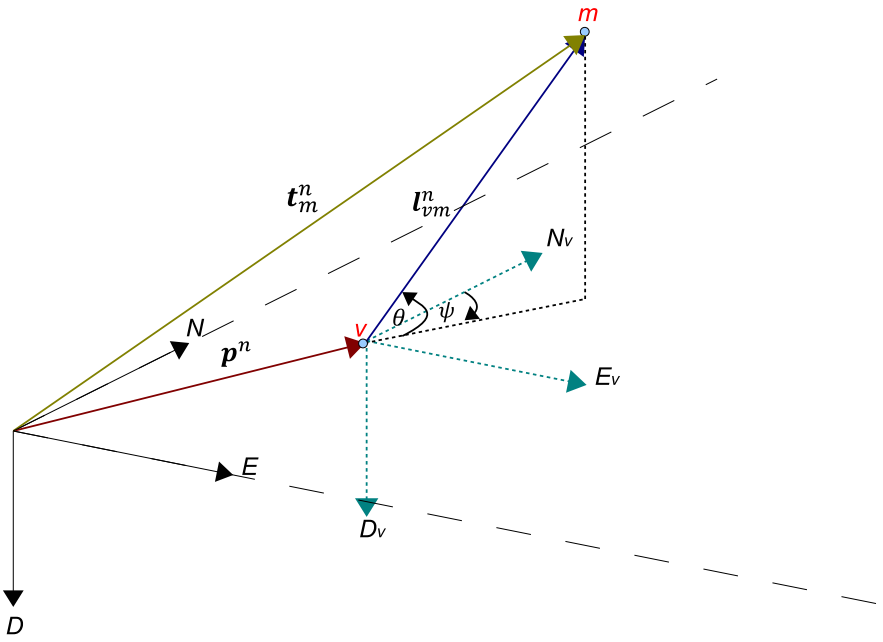


Figure 3.15: Relative azimuth and elevation in the n-frame.

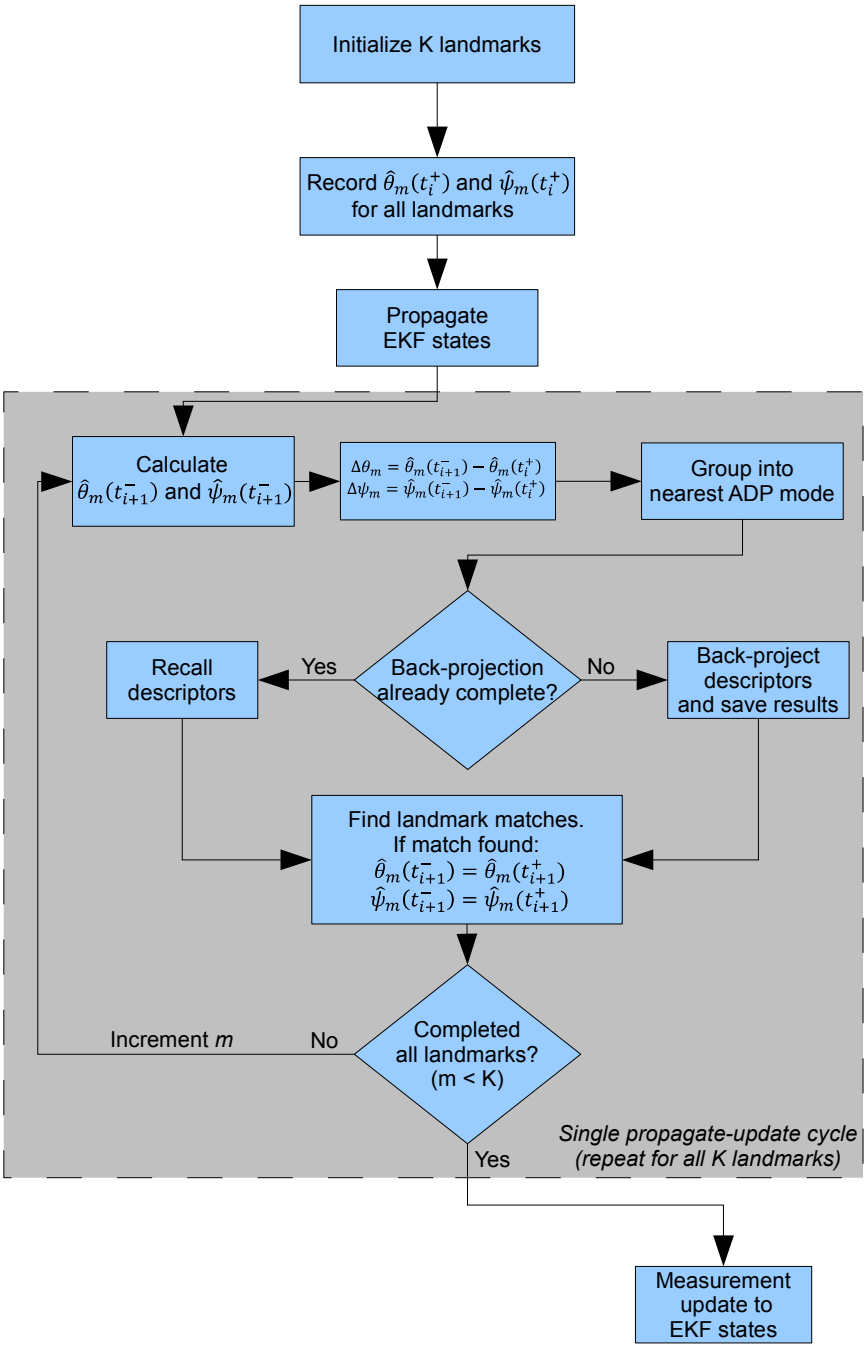


Figure 3.16: ADP algorithm block diagram.

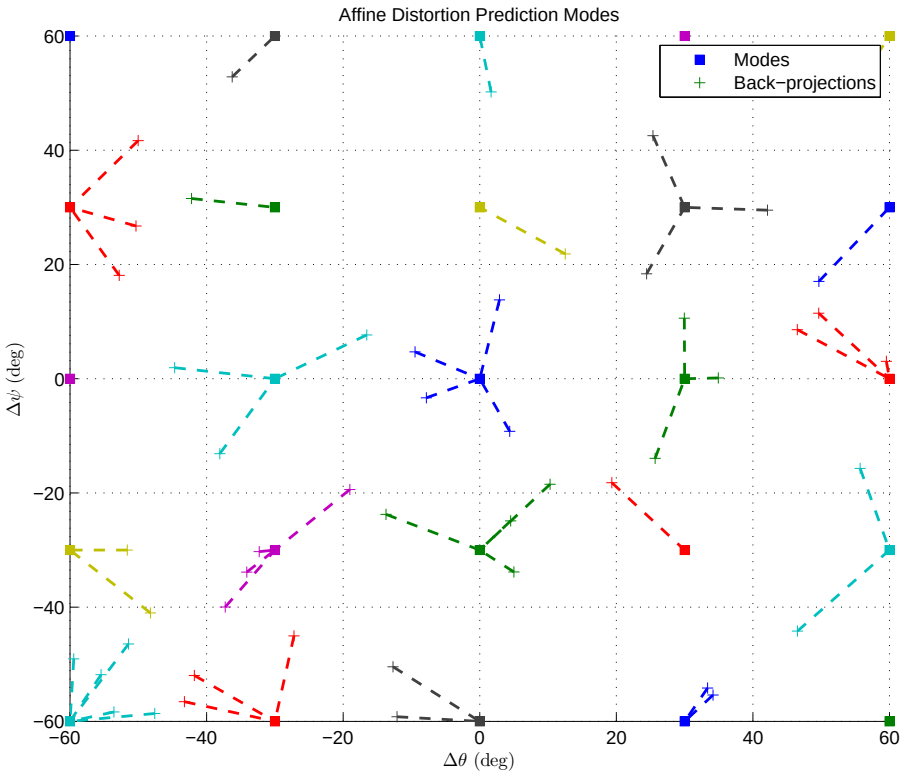


Figure 3.17: Sample ADP mode grouping.

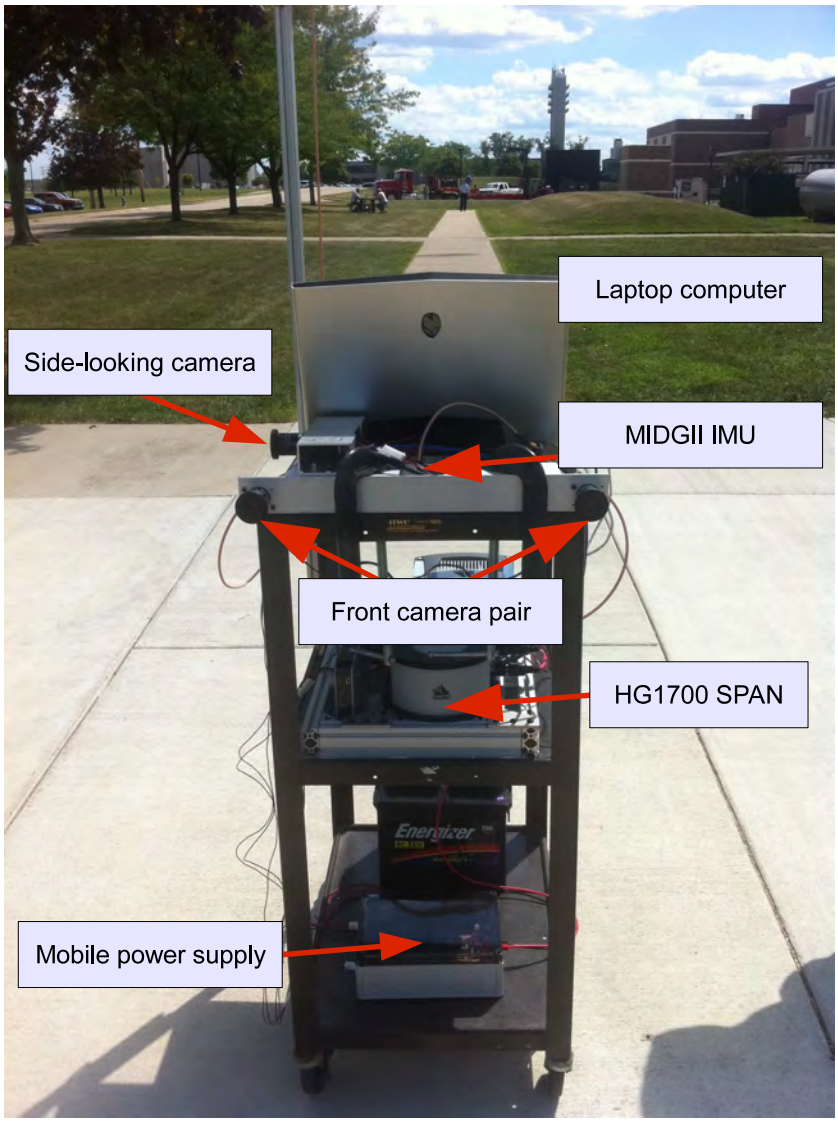


Figure 3.18: Data collection push-cart illustration. The push-cart collection rig was manually pushed around a parking lot during Run 1, primarily to collect enough data to tune the IAEKF variables.

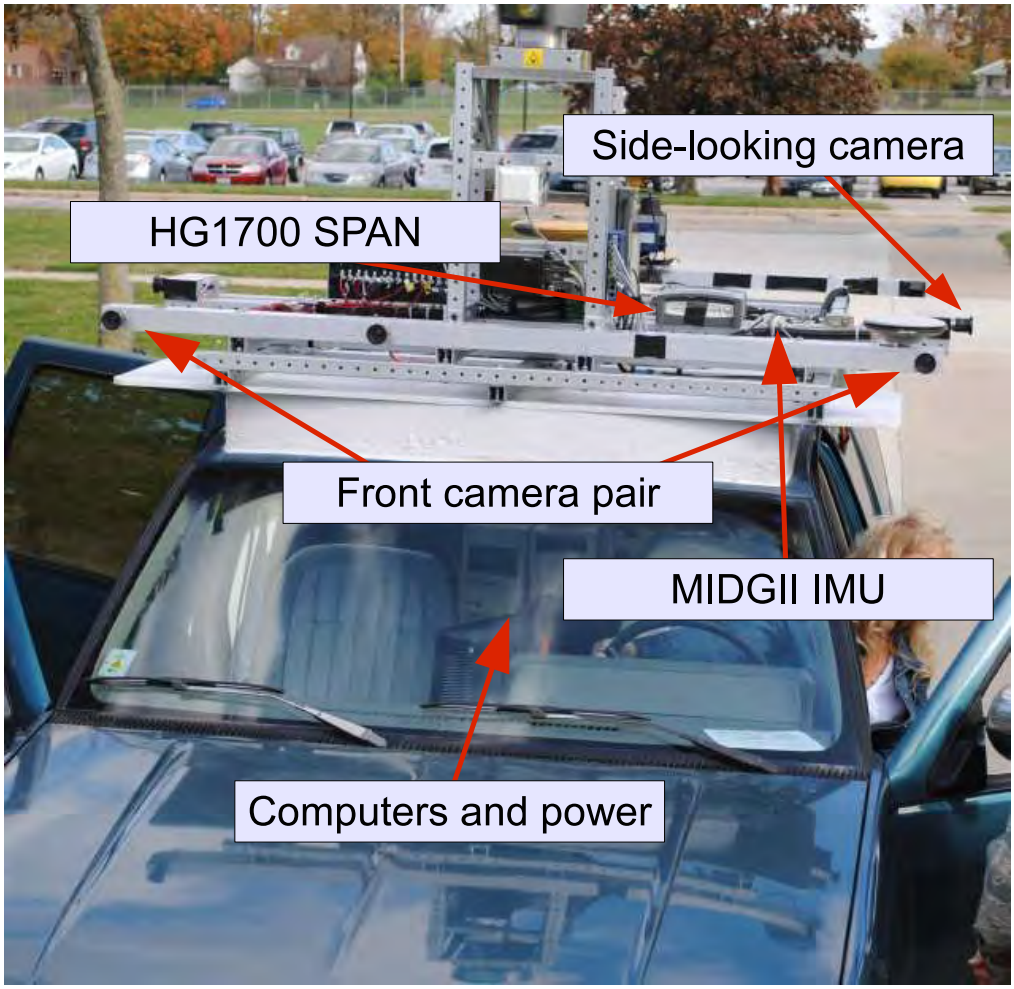


Figure 3.19: Data collection van illustration. The van-mounted collection rig was used in a downtown environment during Run 2 and was the primary collection source for this research.

Table 3.4: Camera models table. This table summarizes the models for the various imaging sensors used for this research.

	125495	**122865**	**101205**	**107971**	**102828**	**115970**
Make	Prosilica	Prosilica	Prosilica	Prosilica	Prosilica	Prosilica
Lens	Pentax	Pentax	Pentax	Pentax	Pentax	Pentax
Focal length [mm]	[1376,1374]	[1361,1359]	[1375,1373]	[928,927]	[935,933]	[1106,1105]
Principal point [pix]	[604,522]	[611,492]	[633,532]	[766,595]	[797,559]	[539,404]
$\mathbf{k}_1$	-0.119	-0.113	-0.116	-0.094	-0.118	-0.113
$\mathbf{k}_2$	-0.188	0.181	-0.163	-0.099	0.168	0.16
$\mathbf{k}_3$	-6.66×10^{-4}	-1.11×10^{-3}	-1.12×10^{-4}	-1.23×10^{-3}	9.04×10^{-3}	3.47×10^{-4}
$\mathbf{k}_4$	1.54×10^{-4}	-1.05×10^{-3}	-1.66×10^{-4}	2.50×10^{-3}	-8.66×10^{-4}	-1.98×10^{-4}
Resolution [pix]	[960,1280]	[960,1280]	[960,1280]	[1600,1200]	[1600,1200]	[1024,768]

3.2.4 Data Collection Run Descriptions. The automatic calibration algorithm was tested indoors using a single data collection and two consumer-grade cameras. The ADP algorithm was tested through two different collection rigs (Figures 3.18 and 3.19), which were used to perform two different collection runs. Run 1 was performed in a parking lot using the rig shown in Figure 3.18, while Run 2 was performed in an urban downtown using the rig shown in Figure 3.19. The purpose of Run 1 was to collect a short amount of data in order to debug and tune the ADP-enhanced IAEKF algorithm. Having properly tuned the system, Run 2 was then performed as the primary source of data. It is important to note that the system was *not* re-tuned for Run 2. The results from Run 2 are more indicative of real-world system performance, since the system would most likely not be re-tuned prior to runs. Table 3.7 summarizes the key characteristics of each run including the location, duration, and equipment used.

3.3 Summary

This chapter has presented the key contributions of this research in terms of algorithm development for both the automatic calibration and affine distortion prediction algorithms. Additionally, the test equipment and experiments used to validate the algorithms were described. The results from these experiments are analyzed and illustrated in Chapter IV.

Table 3.5: Sensor installation table for Run 1. This table summarizes the (vehicle) body-to-sensor translation vector ($\mathbf{T}_{bs}^{b}$) and sensor-to-body DCMs ($\mathbf{C}_{s}^{b}$) for each sensor used in Run 1. The body (b) frame is defined as shown in Figure 2.3.

	$\mathbf{T}_{bs}^{b}[\mathbf{m}]$	$\mathbf{C}_{s}^{b}$
IMU	$\begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix}$	$\begin{bmatrix} 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$
Cam 1	$\begin{bmatrix} 0.0191 \\ -0.2273 \\ -0.0080 \end{bmatrix}$	$\begin{bmatrix} 0.0000 & 0.0000 & 1.0000 \\ 0.0000 & 1.0000 & 0.0000 \\ -1.0000 & 0.0000 & 0.0000 \end{bmatrix}$
Cam 2	$\begin{bmatrix} 0.0204 \\ 0.2289 \\ -0.0135 \end{bmatrix}$	$\begin{bmatrix} -0.0031 & -0.0015 & 1.0000 \\ 0.0029 & 1.0000 & 0.0015 \\ -1.0000 & 0.0029 & -0.0031 \end{bmatrix}$
Cam 3	$\begin{bmatrix} -0.0864 \\ 0.1692 \\ -0.0476 \end{bmatrix}$	$\begin{bmatrix} -0.0353 & -0.9994 & -0.0049 \\ 0.0251 & -0.0058 & 0.9997 \\ -0.9991 & 0.0352 & 0.0253 \end{bmatrix}$

Table 3.6: Sensor installation table for Run 2. This table summarizes the (vehicle) body-to-sensor translation vector ($\mathbf{T}_{bs}^{b}$) and sensor-to-body DCMs ($\mathbf{C}_{s}^{b}$) for each sensor used in Run 2. The body (b) frame is defined as shown in Figure 2.3.

	$\mathbf{T}_{bs}^{b}$[m]	$\mathbf{C}_{s}^{b}$
IMU	$\begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix}$	$\begin{bmatrix} 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$
Cam 1	$\begin{bmatrix} 0.0318 \\ -0.1655 \\ -0.0088 \end{bmatrix}$	$\begin{bmatrix} 0.0000 & 0.0000 & 1.0000 \\ 0.0000 & 1.0000 & 0.0000 \\ -1.0000 & 0.0000 & 0.0000 \end{bmatrix}$
Cam 2	$\begin{bmatrix} 0.0165 \\ 1.6025 \\ -0.0040 \end{bmatrix}$	$\begin{bmatrix} 0.0081 & 0.0019 & 1.0000 \\ -0.0017 & 1.0000 & -0.0019 \\ -1.0000 & -0.0017 & 0.0081 \end{bmatrix}$
Cam 3	$\begin{bmatrix} -0.4731 \\ -0.2259 \\ -0.0500 \end{bmatrix}$	$\begin{bmatrix} -0.0075 & 0.9988 & 0.0494 \\ 0.0068 & 0.0494 & -0.9988 \\ -0.9999 & -0.0072 & -0.0071 \end{bmatrix}$

Table 3.7: Data collection run descriptions. This table characterizes the data collection runs used during this research.

	Run 1	**Run 2**
Location	Parking lot	Urban downtown
Vehicle	Push cart	Van
Duration [mm:ss]	5:40	9:00
Total distance [m]	172.04	2.257×10^3
Left Camera	125495	107971
Right Camera	122865	102828
Side Camera	101205	115970
Side Camera Direction	Right	Left
IMU	MIDG II	MIDG II
Truth Source	HG1700 SPAN	HG1700 SPAN

IV. Results and Analysis

This chapter discusses the results from the experiments described in Section 3.2, which were used to evaluate the automatic calibration and affine distortion prediction algorithms developed during this thesis. The methods used to process and analyze the collected data are presented in Section 4.1. Sections 4.2 and 4.3 present and discuss the results from the automatic calibration and affine distortion prediction algorithms respectively.

4.1 Data Processing

The calibration and navigation data collected during the experiments described in Section 3.2 were post processed using The Mathworks, Inc.'s Matlab software, version R2010B. Data processing included image formatting, truth trajectory generation, accelerometer and gyro bias estimation and error calculation.

4.1.1 Image Formatting. Prior to processing each run, the collected raw image data was timestamped within $\pm$ 1 millisecond, renamed, and organized into respective folders in order to decrease the computational load on the IAEKF. Next, every image was pre-rectified to account for lens distortions (as explained in Section 3.1.2.6). Finally, SIFT features were precomputed for every rectified image in order to reduce the run-time load on the IAEKF.

4.1.2 Error Calculation . A tactical-grade SPAN navigation platform from Novatel, consisting of a Honeywell HG1700 inertial sensor and a GPS receiver, was used to collect high-fidelity position, velocity, and attitude data during both runs. This *truth* data was preprocessed to generate true position, velocity, and attitude states in order to verify the performance of the IAEKF output. State errors will be computed by subtracting estimated states from their respective true states in Section 4.3.2.

4.1.3 Bias Estimation. During the beginning of each run, the IAEKF was allowed to estimate accelerometer and gyroscope biases by observing INS measurements and comparing them to known or reference states. For this research, the high-fidelity truth data was given to the IAEKF in the form of alignment measurements with low covariance for a short duration. In real-world applications, this process can be also accomplished by allowing the IAEKF to run while the vehicle is stationary. In order to simulate real-world applications, the collection rigs remained stationary for a brief period of time before executing the run. Run 1 (calibration run) was aligned for two minutes, while Run 2 (demonstration run) was aligned for 30 seconds.

4.2 Automatic Camera Calibration

Since there was no practical way of obtaining truth data for comparison, the automatic calibration algorithm was tested by calibrating a single binocular pair of consumer-grade cameras and comparing the calibration results to a standard calibration from Bouguet's toolbox. Instead of comparing each parameter, the calibration routines were compared in terms of average reprojection error, which represents the effective accuracy of the entire camera model produced. As shown in Table 4.1, the automatic calibration algorithm provided a 70 % increase in calibration speed (by eliminating manual calibration steps) and averaged 4 times more calibration points per image while maintaining sub-pixel errors. It is important to note that, unlike Bouguet's algorithm, the automatic calibration algorithm did not require the entire calibration board to be visible, which allowed the calibration board to be held closer to the binocular camera set. Consequently, the calibration board covered a larger part of the *pix*-frame when compared to standard calibration as shown in Figure 4.1. Although the mean reprojection error was sub-pixel, the automatic calibration algorithm exhibited a significant increase in reprojection error spread when compared to Bouget's toolbox (illustrated in Figure 4.2). This increase in spread is most likely due to the lower SIFT feature localization and matching accuracy when compared to the corner localization accuracy of the Harris corner detector. Additionally, since the

Table 4.1: Automatic calibration results. This table compares the calibration results from an automatic calibration routine and a standard calibration routine.

	Standard	**Automatic**
Calibration time [min]	10	3
Number of images used	10	7
Approximate points per image	100	400
Average reprojection error [pix]	[0.20 0.18]	[0.35 0.27]

user did not manually delinate the calibration board, the algorithm's estimate of the board's pose ([**R t**]) could have been degraded, which resulted in degraded camera model estimates (**A**).

4.3 Affine Distortion Prediction

The ADP algorithm was tested during two collection runs, using similar collection rigs. The calibration run was collected using the push-cart rig, and its primary purpose was to collect enough data to properly tune the IAEKF. Having tuned the system variables, the demonstration run which was collected using the van-mounted rig, was used to verify the effects of ADP while varying the number of cameras used.

4.3.1 Trajectory Comparisons. As mentioned in Section 4.1.2, the high-fidelity truth data was used to generate "true" trajectories for each run. In this section, the IAEKF filtered output trajectories are compared to the truth trajectories in the horizontal (North-East plane) and vertical (Down) dimensions.

4.3.1.1 Calibration Run. As shown in Figures 4.3 and 4.4, varying the number of cameras or the usage of ADP did not significantly alter the resulting filter output trajectory for the calibration run. This was somewhat expected due to its short duration. It is important, however, to note that the processing profiles including the usage of 3 cameras (with ADP on and off) did result in a Down trajectory closer to the known terrain variations of the collection location. Since the data collection motion

profile included a double loop in the horizontal trajectory, the vertical trajectory was expected to exhibit a double "dip" which was not present in the reference trajectory, but approximated by the IAEKF output trajectories involving 3 cameras. This minor yet important discrepancy points at an underlying problem in the truth data for this run. Therefore, the truth data was discarded after $t = 250$ seconds. Again, since the run was useful in providing tuning parameters, this partial loss of truth data was not catastrophic.

4.3.1.2 Demonstration Run. The trajectory comparisons for the demonstration run are shown in Figures 4.5 and 4.6. As illustrated by the figures, each processing profile resulted in a slightly different IAEKF output trajectory. The differences in the output trajectories are most noticeable in the latter part of the run (as expected), especially in the vertical trajectory. These trajectory comparisons indicate, at an early stage, that the usage of ADP resulted in filtered trajectories closer to the truth in both the two and three camera configurations. This is especially noticeable in the vertical trajectory comparison (Figure 4.6) and could be attributed to either the increase of positive landmark matches, or the decrease of false landmark matches during the run. Unfortunately, the veracity of individual landmark matches is impractical to determine since it would require careful manual supervision of the real-time navigation computations for the run.

4.3.2 Navigation State Errors. In this section, the IAEKF filtered outputs for the nine navigation states (3 position, 3 velocity, 3 attitude) are analyzed in terms of error, which was obtained by subtracting the corresponding true states as described in Section 4.1.2.

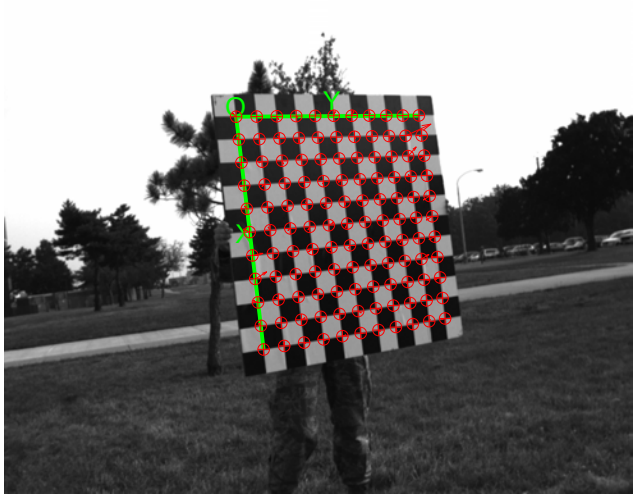
4.3.2.1 Calibration Run. Table Table 4.2 summarizes the Root Sum Squared (RSS) errors for the nine Position, Velocity, and Attitude (PVA) navigation states in the calibration run, while Figures 4.7 through 4.12 illustrate the absolute and RSS errors for each navigation state as functions of time. Since the truth data

Table 4.2: PVA RSS errors for the calibration run. This table summarizes the mean RSS position, velocity, and attitude errors for the calibration run across all four processing configurations. The 4^{th} row of the table displays position errors as a percentage of the total distance traveled (172 m).

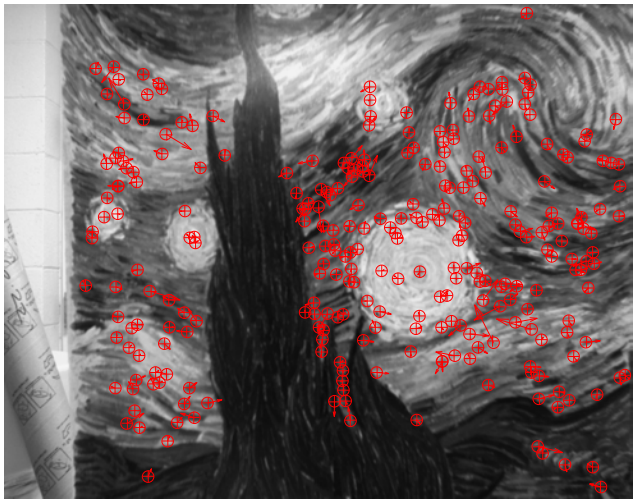
	C1	**C2**	**C3**	**C4**
Cameras	2	2	3	3
ADP	Off	On	Off	On
Position [m]	0.64	0.64	0.63	0.63
Position [%]	0.37	0.37	0.37	0.37
Velocity [m/s]	0.053	0.053	0.055	0.055
Attitude [mrad]	41.98	41.98	42.06	42.06

was determined to be invalid after $t = 250$, the error computations have been limited to exclude times greater than 250 seconds.

As illustrated by Figures 4.7 through 4.12 and foreshadowed by the trajectory comparisons, there was no significant difference in position, velocity or attitude errors for the calibration run. This is most likely due to the short duration of the run. The mean position, velocity, and attitude RSS errors increased slightly for the processing profiles involving three cameras, but the increase amounts were not statistically significant.

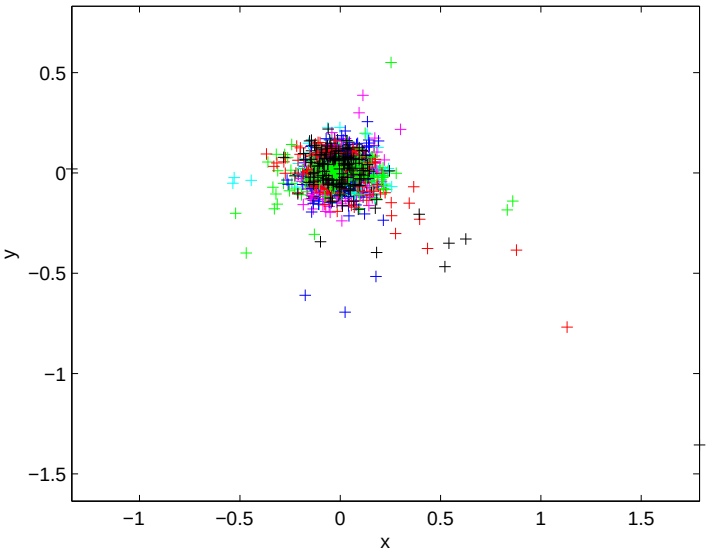


(a)

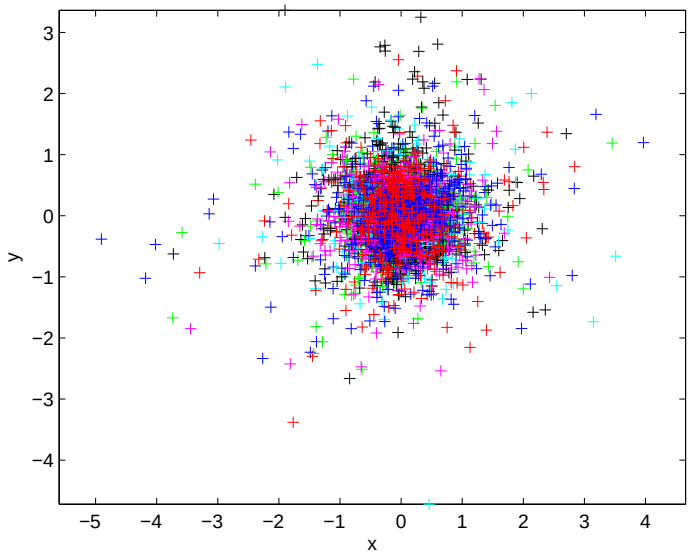


(b)

Figure 4.1: Comparison of *pix*-frame coverage for calibration. (a) Manual Calibration (b) Automatic Calibration. The increased lens and *pix*-frame coverage, enabled through automatic calibration, explains the slight increase in reprojection error when compared to Bouguet's toolbox.



(a)



(b)

Figure 4.2: Calibration reprojection error illustration. (a) Manual Calibration (b) Automatic Calibration. Although still providing sub-pixel averages, the automatic calibration routine showed a slight increase in average reprojection error and spread when compared to Bouguet's toolbox.

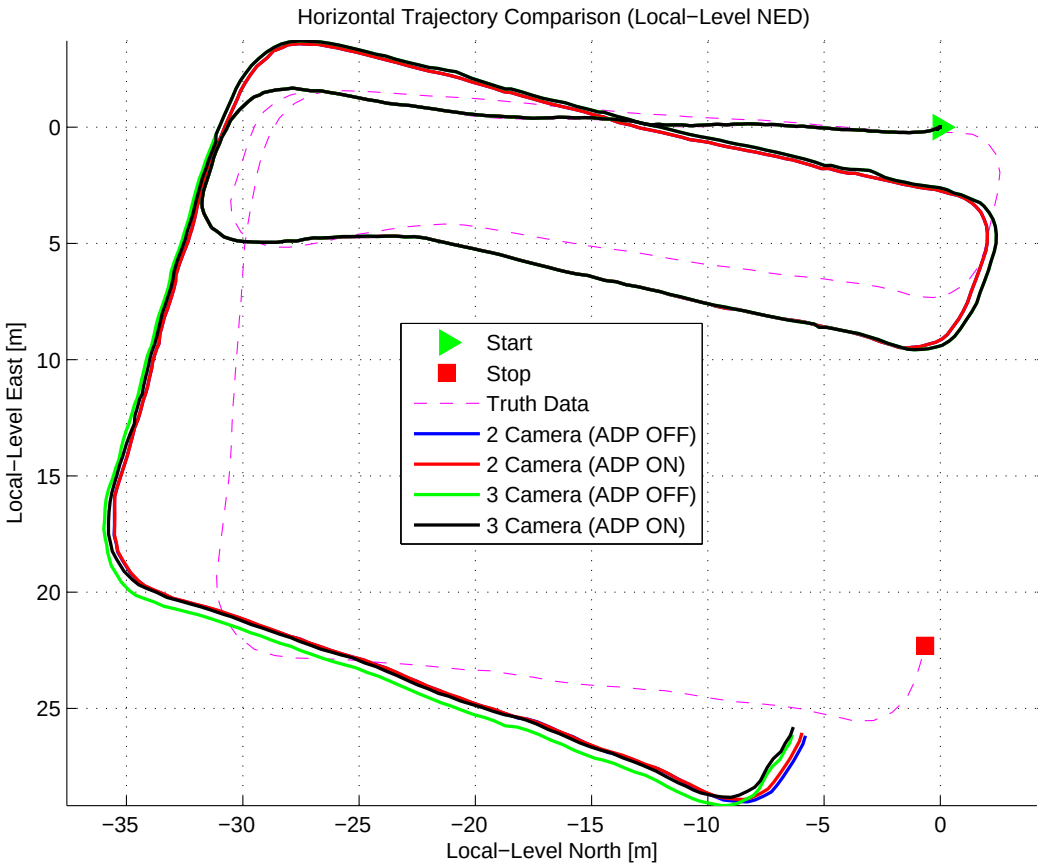


Figure 4.3: N-E trajectory comparison plots for the calibration run.

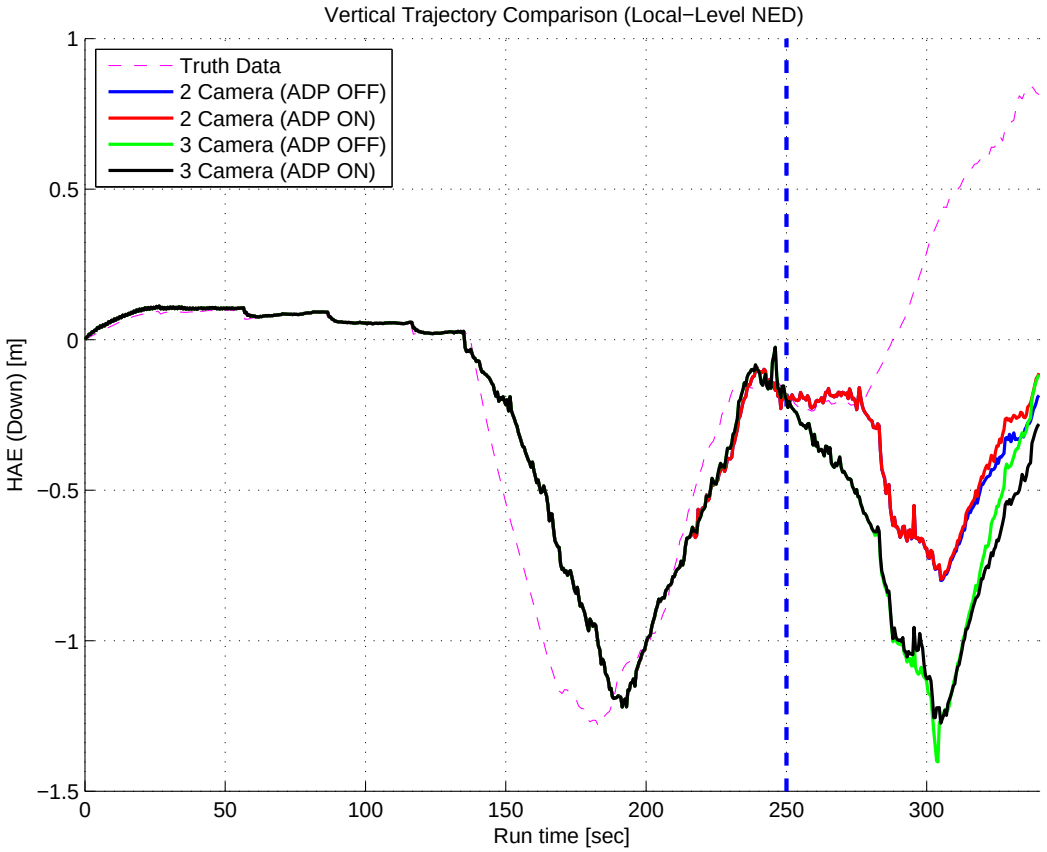


Figure 4.4: Down trajectory comparison plots for the calibration run. *Note that truth data is invalid after* $t = 250$ *s.*

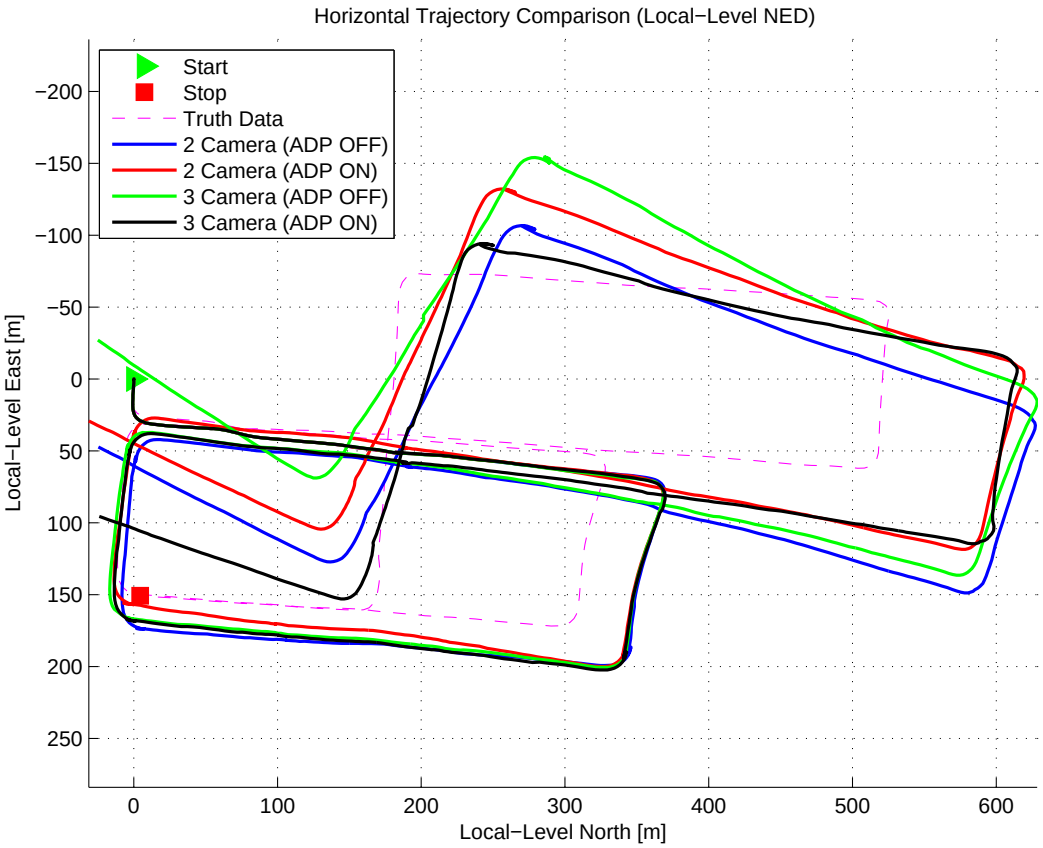


Figure 4.5: N-E trajectory comparison plots for the demonstration run.

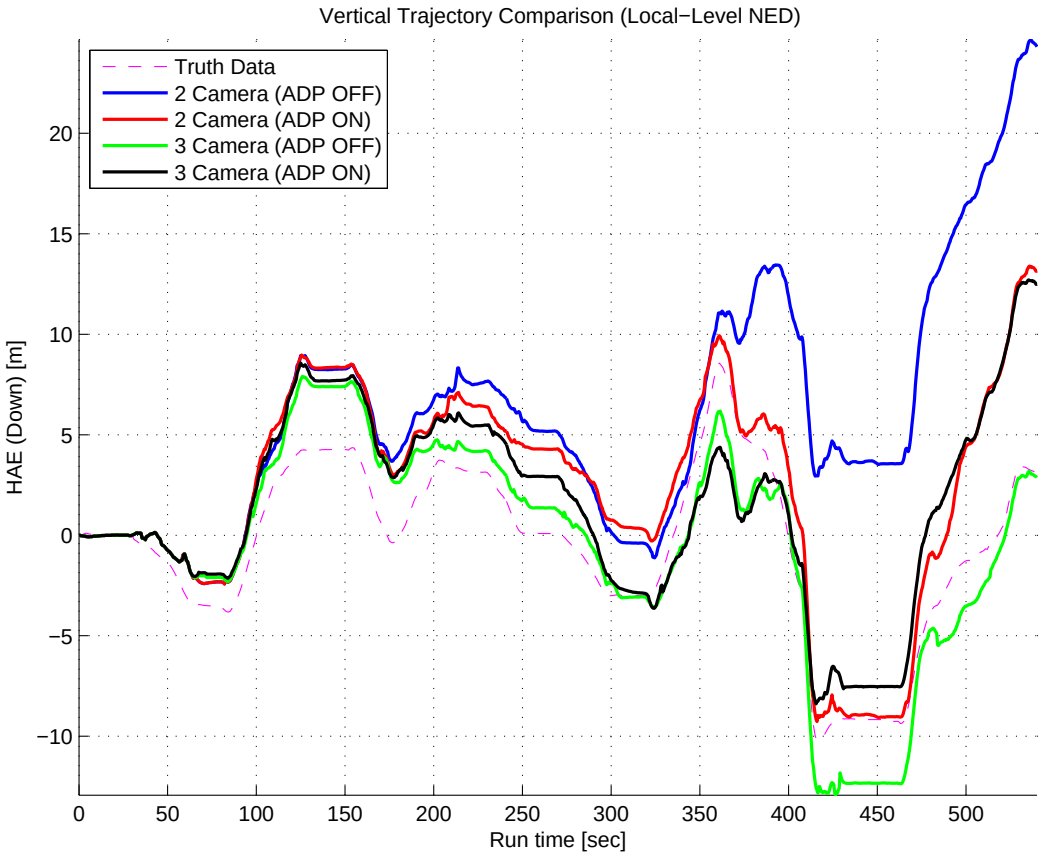


Figure 4.6: Down trajectory comparison plots for the demonstration run.

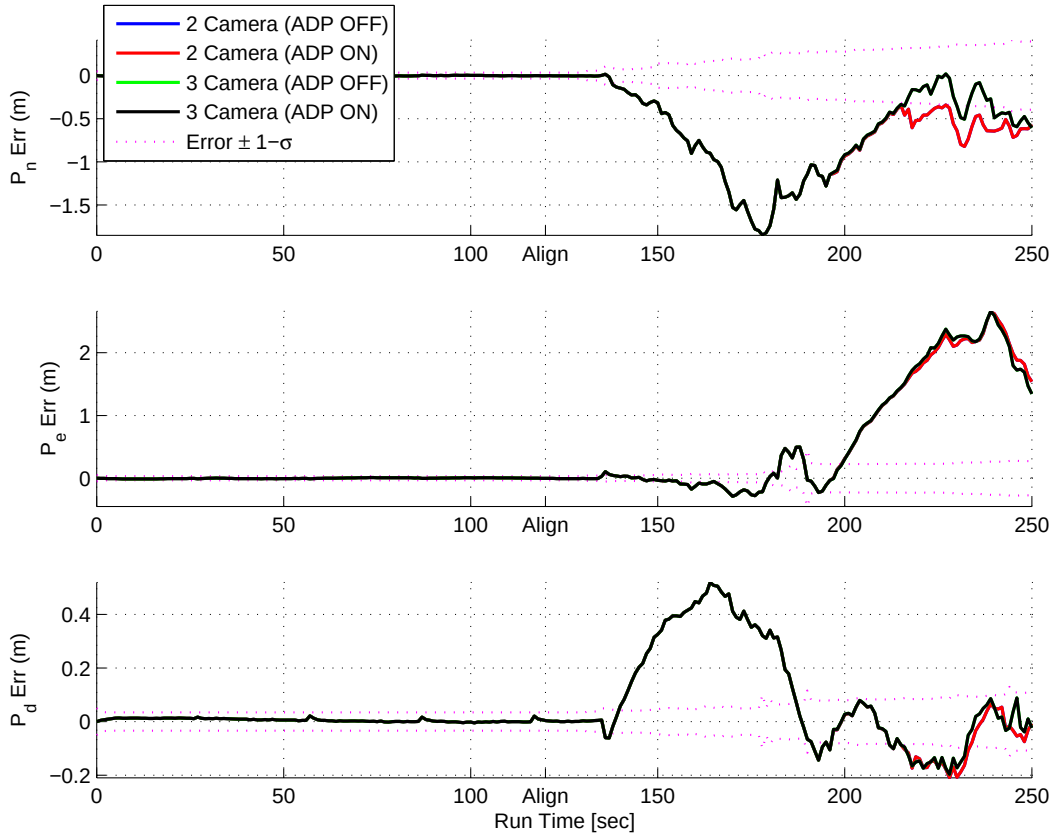


Figure 4.7: Position error plots for the calibration run. Note that ADP ON and ADP OFF profiles overlap for both 2 and 3 camera configurations.

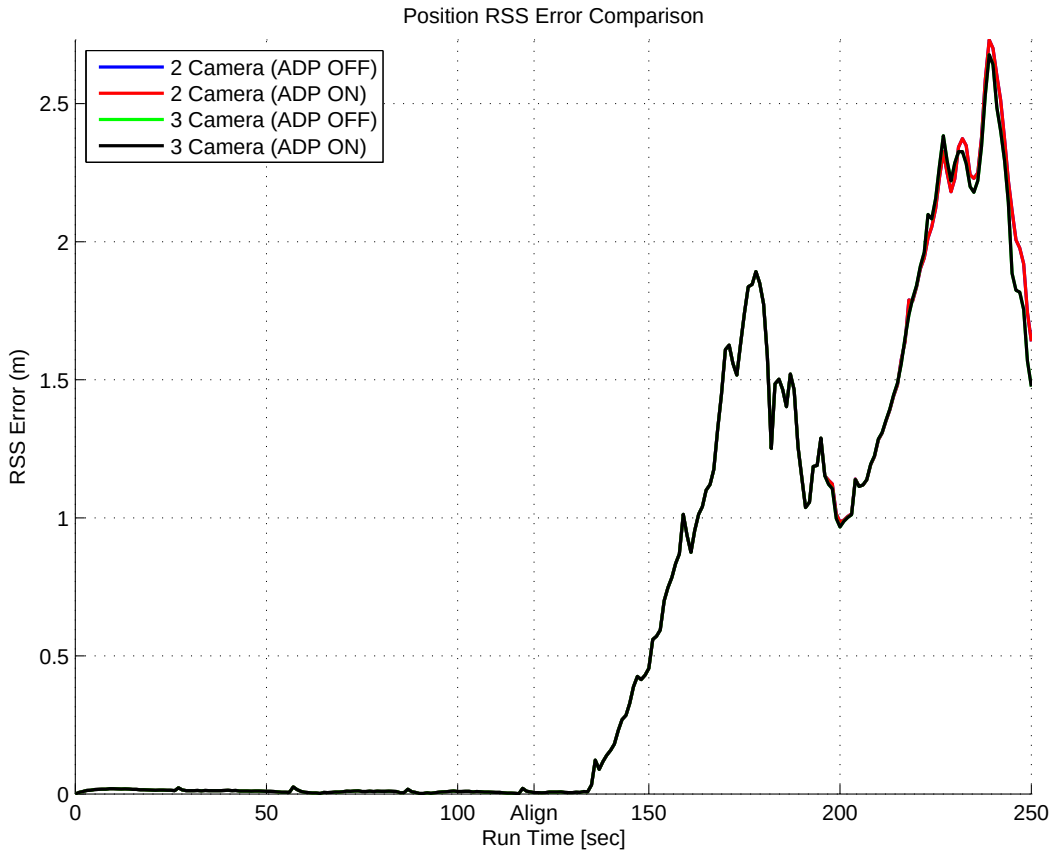


Figure 4.8: Position RSS error plots for the calibration run. Note that ADP ON and ADP OFF profiles overlap for both 2 and 3 camera configurations.

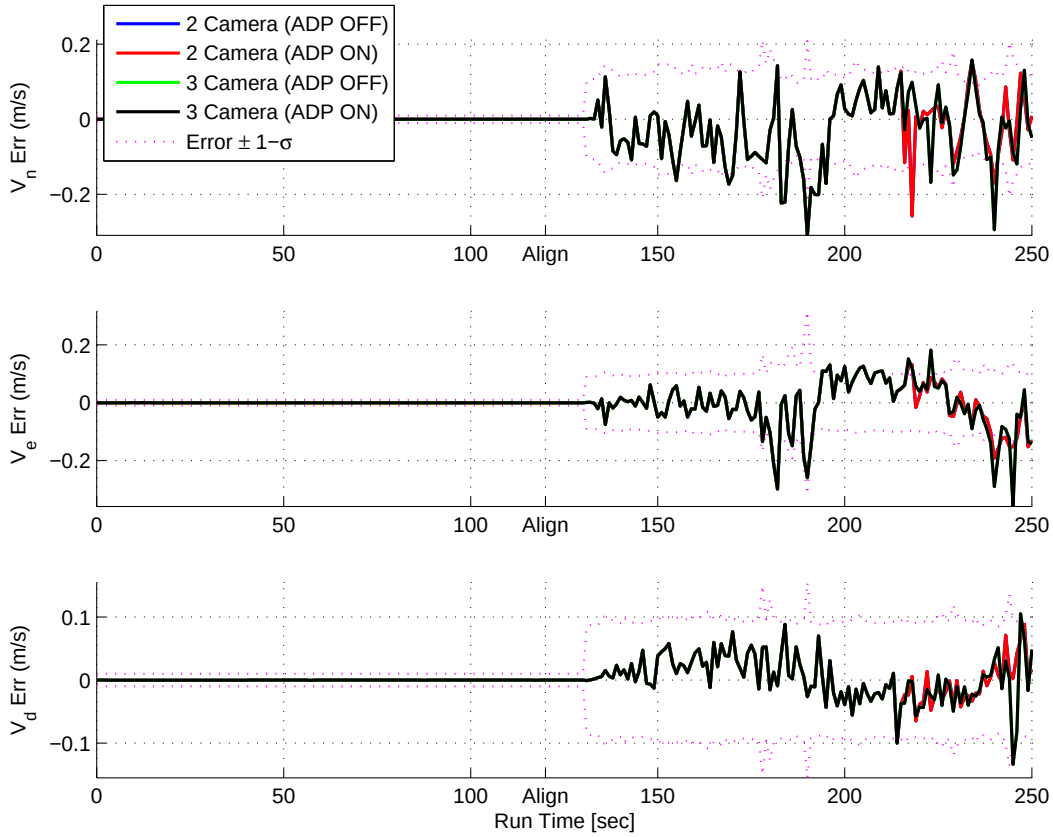


Figure 4.9: Velocity error plots for the calibration run. Note that ADP ON and ADP OFF profiles overlap for both 2 and 3 camera configurations.

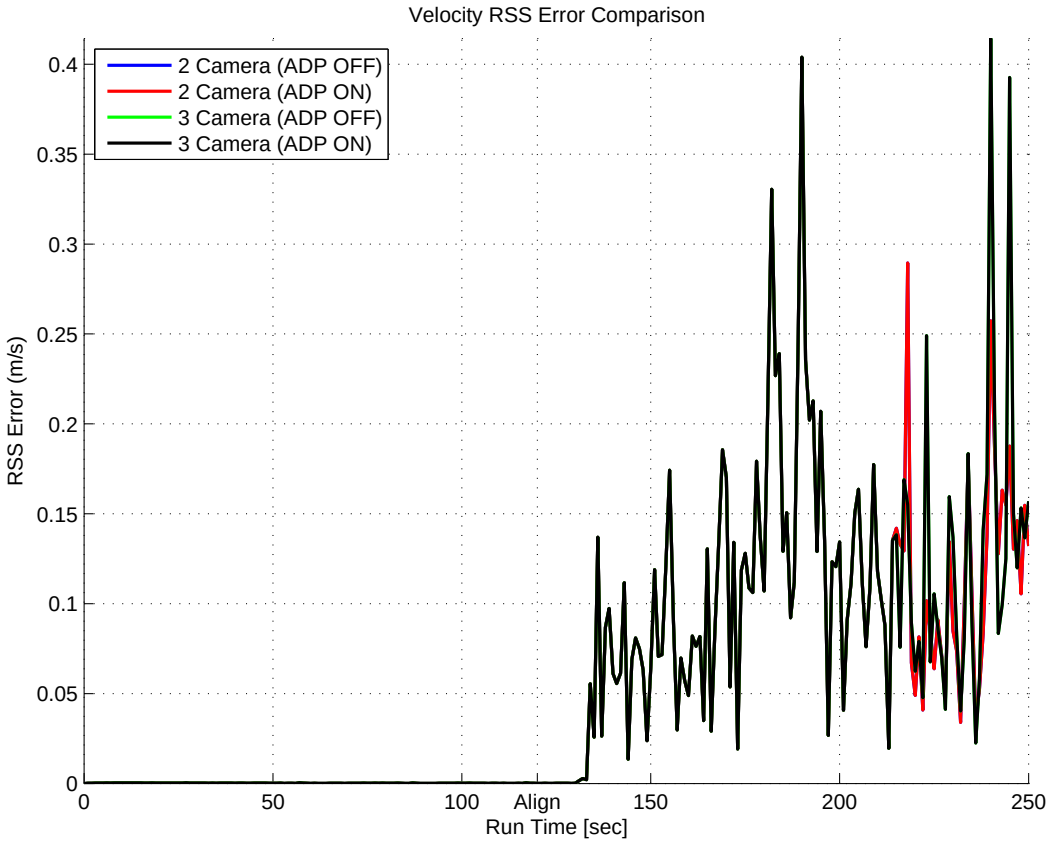


Figure 4.10: Velocity RSS error plots for the calibration run. Note that ADP ON and ADP OFF profiles overlap for both 2 and 3 camera configurations.

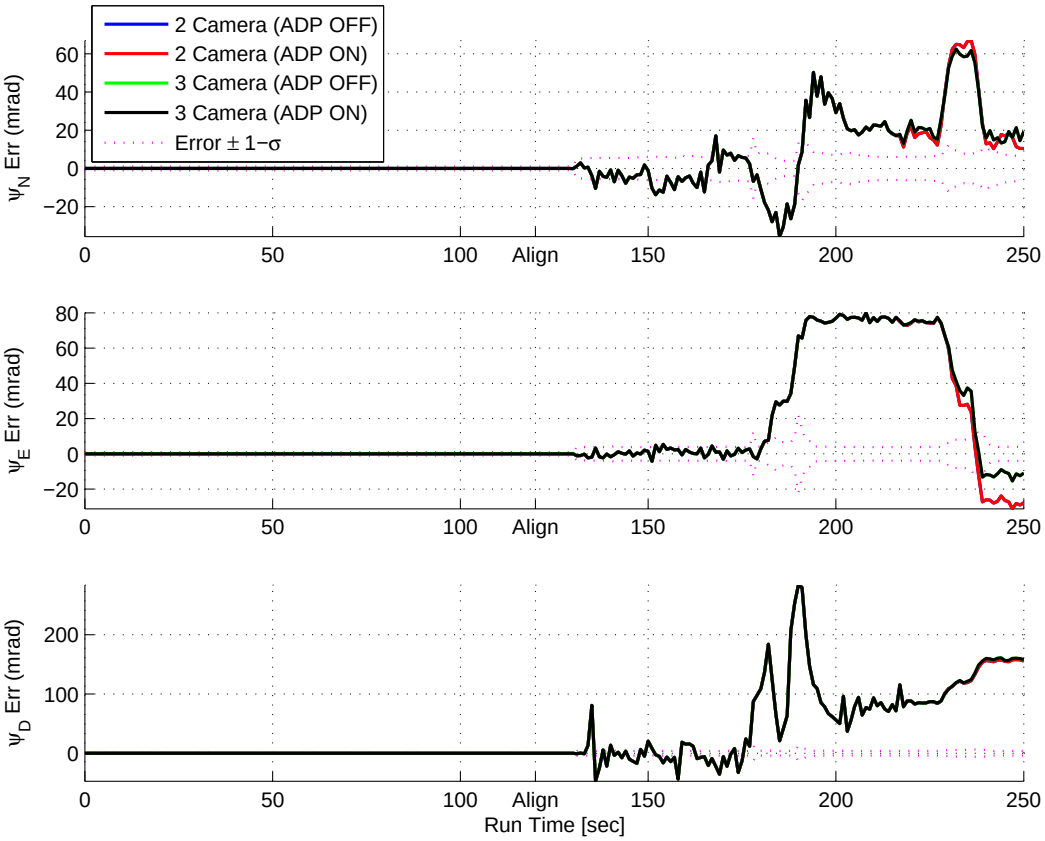


Figure 4.11: Attitude error plots for the calibration run. Note that ADP ON and ADP OFF profiles overlap for both 2 and 3 camera configurations.

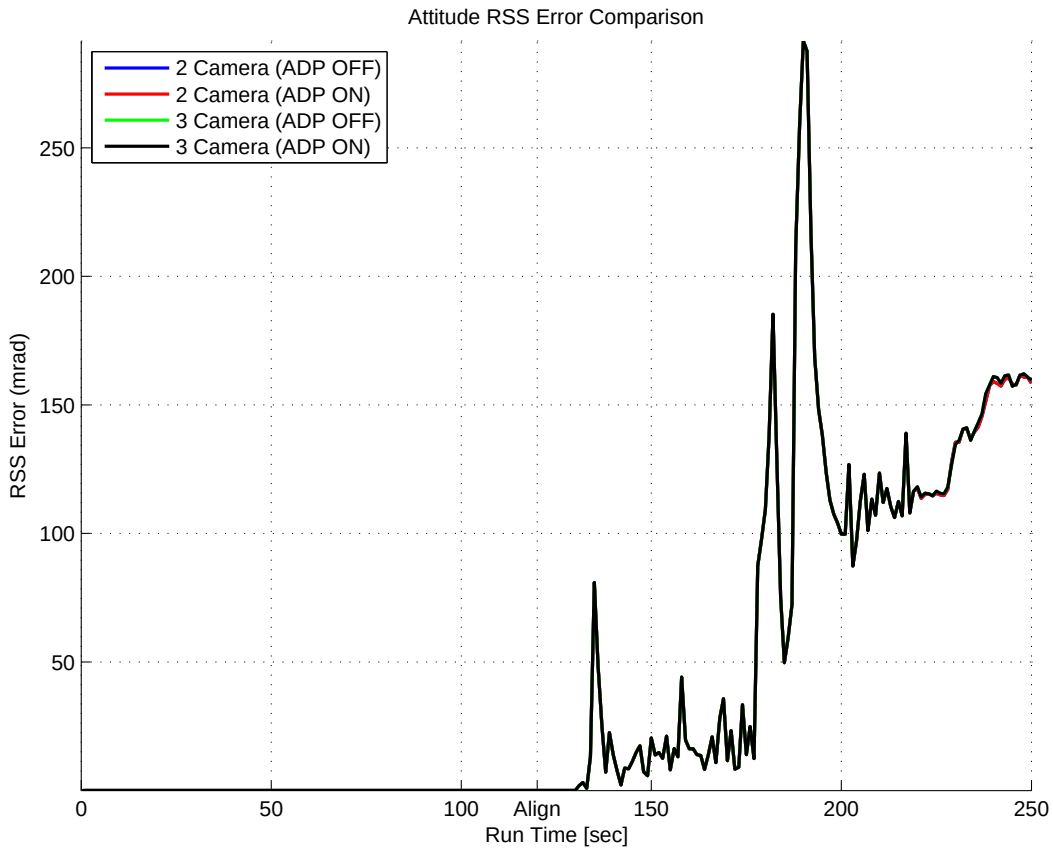


Figure 4.12: Attitude RSS error plots for the calibration run. Note that ADP ON and ADP OFF profiles overlap for both 2 and 3 camera configurations.

Table 4.3: PVA RSS errors for the demonstration run. This table summarizes the mean RSS position, velocity, and attitude errors for the demonstration run across all four processing configurations. The 4^{th} row of the table displays position errors as a percentage of the total distance traveled (2.26 km).

	C1	**C2**	**C3**	**C4**
Cameras	2	2	3	3
ADP	Off	On	Off	On
Position [m]	46.25	42.10	52.92	34.98
Position [%]	2.05	1.87	2.34	1.53
Velocity [m/s]	1.16	1.14	1.30	0.97
Attitude [mrad]	129.89	129.57	163.74	84.35

4.3.2.2 Demonstration Run. Table 4.3 summarizes the RSS errors for the nine PVA navigation states in the demonstration run, while Figures 4.13 through 4.18 illustrate the absolute and RSS errors for each navigation state over time.

For the demonstration run, the processing profile selection had a significant impact on navigation errors. In an absolute sense, the position, velocity, and attitude errors for this run were higher across all processing profiles when compared to the calibration run. However, it is important to note that the demonstration run was approximately 1.6 times longer in duration and 13.1 times larger in total distance when compared to the calibration run. Since the IAEKF algorithm is still based on dead-reckoning, its errors are neither stationary nor ergodic. Therefore, it is reasonable to expect that long runs may not have the same errors as short runs.

Analyzing the errors in the demonstration run relative to each other confirms some of the initial trends foreshadowed in the trajectory analysis, but also reveals an interesting phenomenon. Performing further analysis on the numbers provided by Table 4.3 shows that using the ADP algorithm always improved performance across all navigation states. Interestingly however, using a three-camera configuration without ADP produced the highest errors across all states in the demonstration run. In contrast, the three-camera configuration with ADP resulted in the lowest errors

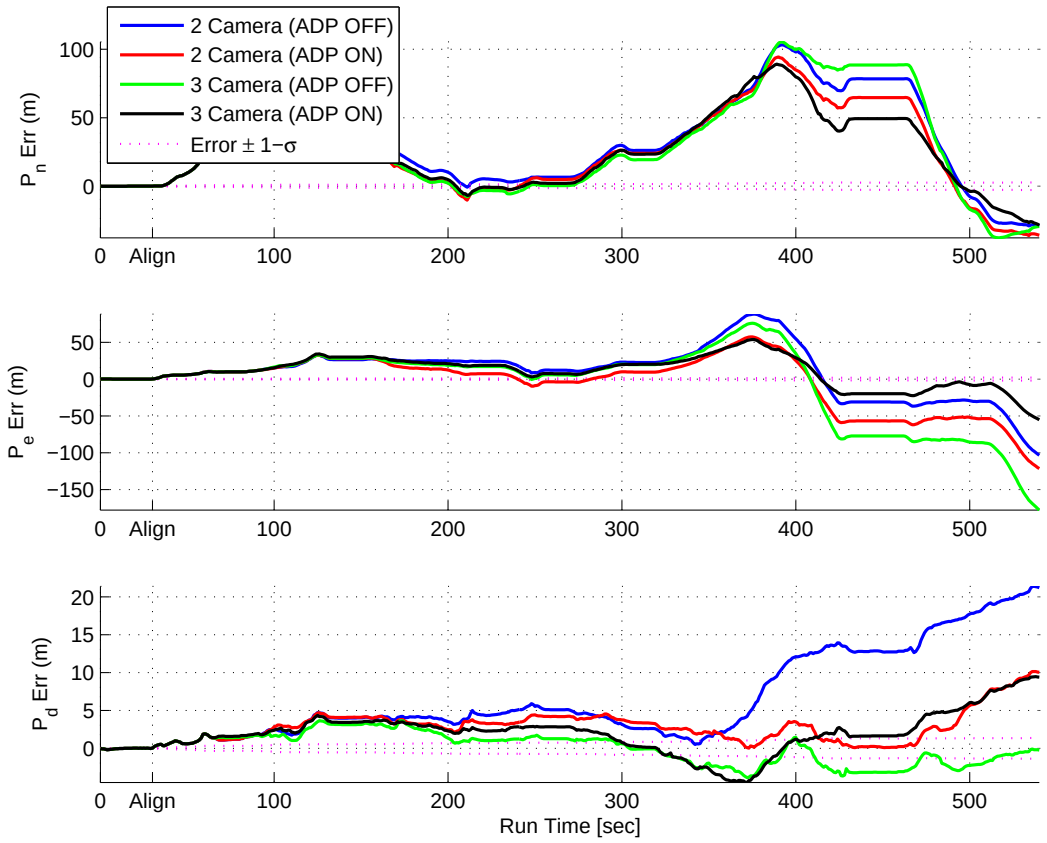


Figure 4.13: Position error plots for the demonstration run.

across all states. These results showcase of the effects of false landmark matches on navigation accuracy. This disparate behavior between the two processing profiles involving three cameras hints at a possible increase of positive matches, decrease of false matches, or a combination thereof when using ADP.

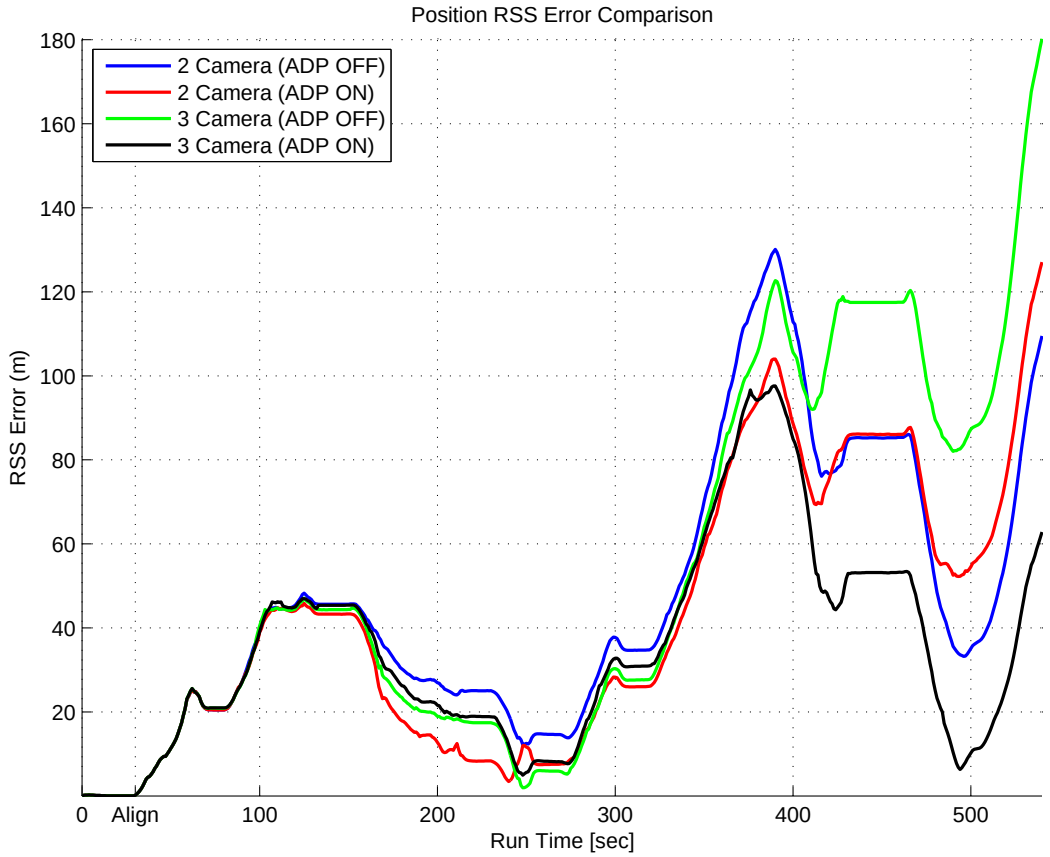


Figure 4.14: Position RSS error plots for the demonstration run.

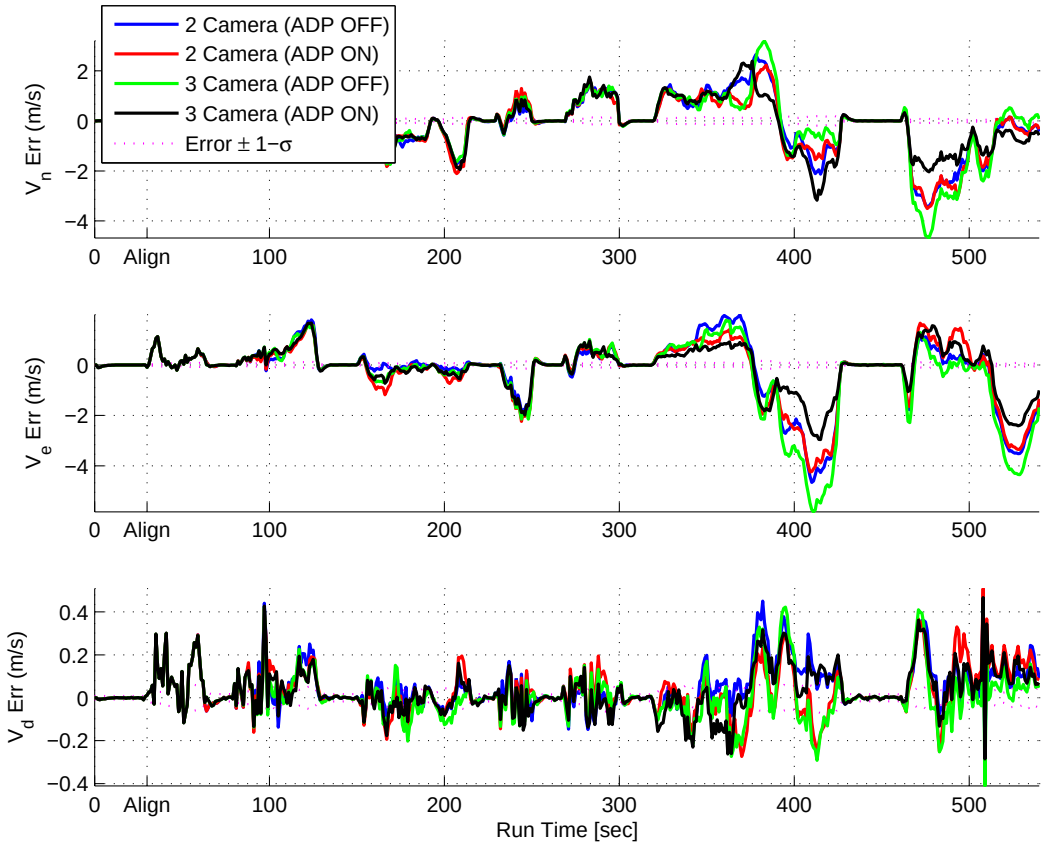


Figure 4.15: Velocity error plots for the demonstration run.

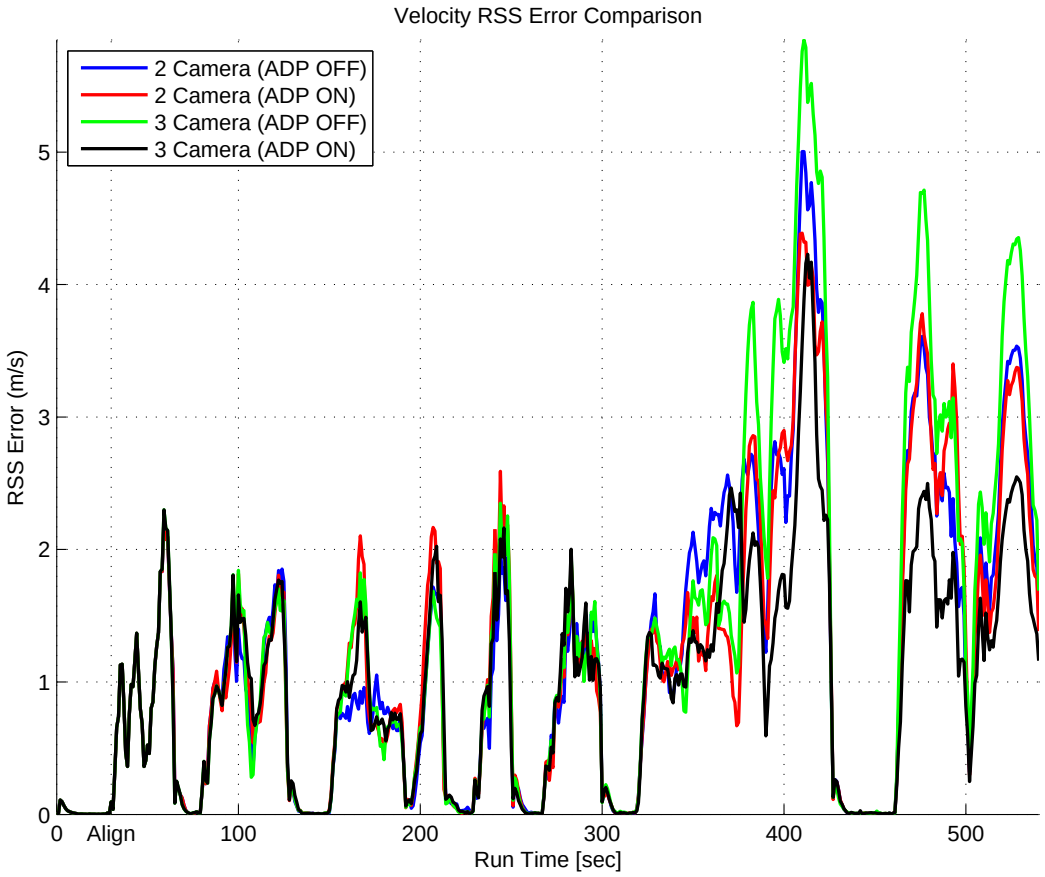


Figure 4.16: Velocity RSS error plots for the demonstration run.

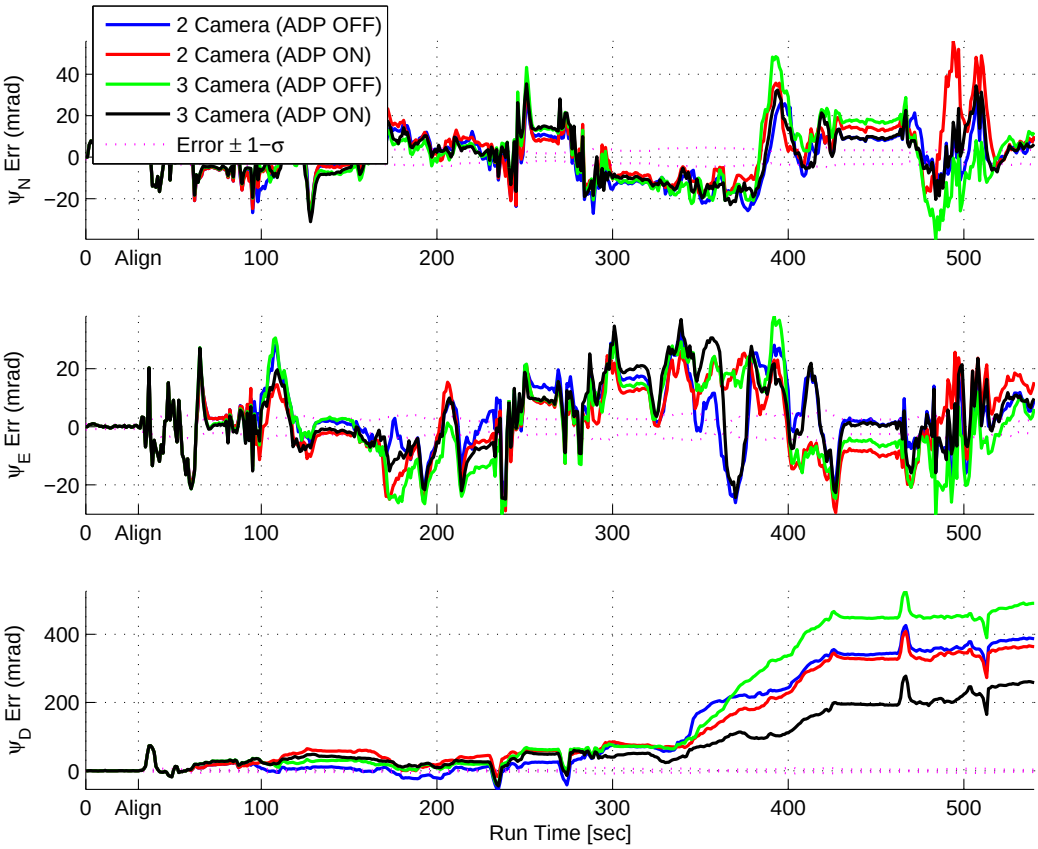


Figure 4.17: Attitude error plots for the demonstration run.

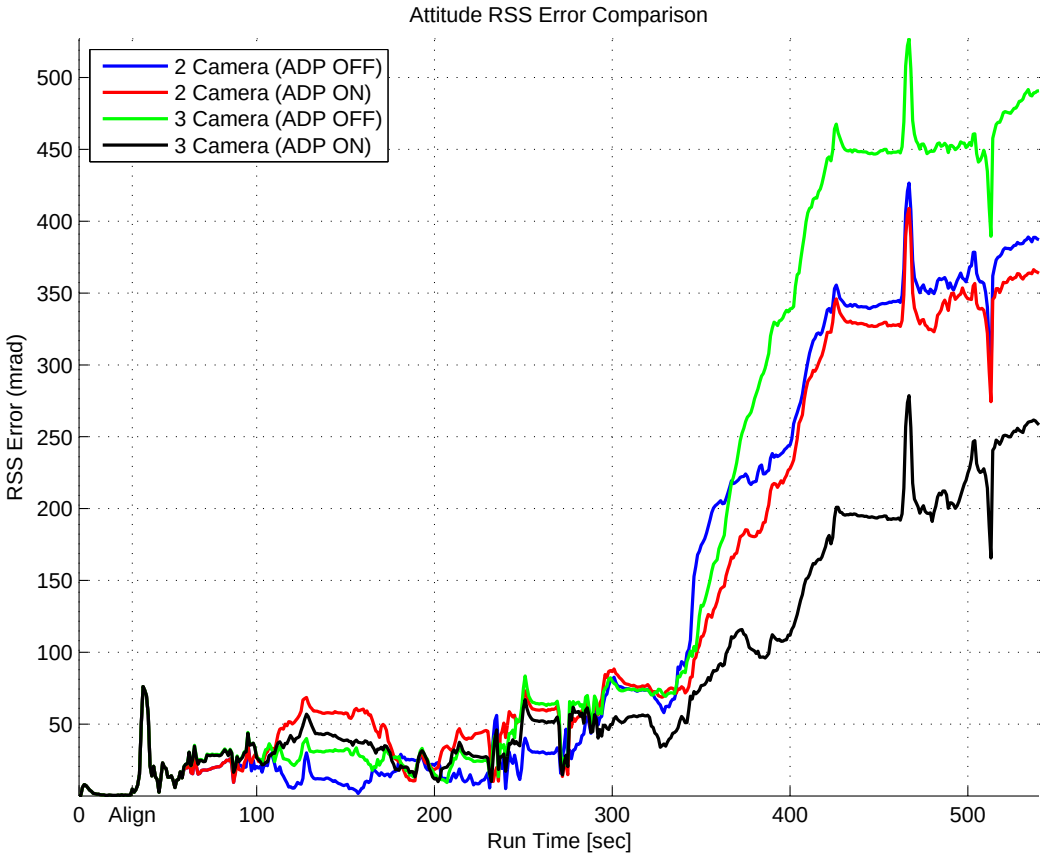


Figure 4.18: Attitude RSS error plots for the demonstration run.

4.3.3 Landmark Tracking. Since ADP is designed to directly improve long-baseline feature tracking, it is important to analyze feature matching and tracking performance for each of the runs.

4.3.3.1 Calibration Run. Tables 4.4 and 4.5 summarize the landmark match and track performance for the calibration run. As expected, the number of total landmark matches is significantly higher for Camera 1 and Camera 2 due to their relative position with respect to the vehicle. Additionally, since all landmarks are always initialized by the front camera pair, it is natural for those cameras to obtain higher match counts. Following the trend set by the trajectory and state error comparisons for the calibration run, there was no major statistical difference in the total landmark match counts across the various processing profiles. In terms of landmark track statistics, the minimum, maximum, mean, and standard deviation for landmark track duration remained statistically unchanged across all processing profiles. Figures 4.19 through 4.21 illustrate the cumulative landmark match counts over time for each of the three cameras. As illustrated by the figures, all processing profiles resulted in similar cumulative landmark match counts with respect to time. Although not statistically significant, Figure 4.21 did display a slightly increased slope on the number of landmark matches with respect to time, which could be an early indication of the effects of ADP on long-baseline landmark tracking. Additionally, it is important to note that the matches obtained with ADP off were not necessarily the same matches obtained with ADP on; it is possible, as supported by the state error results, that enabling ADP decreased false matches and increased positive matches.

4.3.3.2 Demonstration Run. Tables 4.6 and 4.7 summarize the landmark match and track performance for the demonstration run. The landmark match counts for the front camera pair were very similar across all processing profiles, much like in the calibration run. Analyzing the results from Camera 3 reveals more supporting evidence for improved side-camera match performance when using ADP. There is only a small difference between the two final landmark match counts for Cam-

Table 4.4: Landmark match counts for the calibration run 1. This table summarizes the cumulative landmark match counts for the calibration run 1 across all four processing configurations.

	C1	**C2**	**C3**	**C4**
Cameras	2	2	3	3
ADP	Off	On	Off	On
Cam 1	6.83×10^3	6.83×10^3	6.89×10^3	6.87×10^3
Cam 2	6.92×10^3	6.93×10^3	6.94×10^3	6.89×10^3
Cam 3	NA	NA	29	30

Table 4.5: Landmark track statistics the calibration run. This table summarizes the landmark tracking duration statistics for the calibration run across all four processing configurations.

	C1	**C2**	**C3**	**C4**
Cameras	2	2	3	3
ADP	Off	On	Off	On
Min [s]	0	0	0	0
Max [s]	52.00	52.00	52.00	52.00
Mean [s]	4.52	4.58	4.58	4.58
STD [s]	7.41	7.40	7.43	7.40

era 3. However, this information, coupled with the significant improvement across all navigation states when using ADP, points at bigger underlying improvement in landmark match performance. Thus, the data is not intended to imply that the Camera 3 matches with ADP off were the same ones found with ADP on. Rather, while the count is similar, the error analysis suggests that enabling ADP increased match accuracy. Figures 4.22 and 4.23 display similar landmark match performance for the front camera pair, as was the case with the calibration run. Meanwhile, Figure 4.24 shows a significant increase in cumulative landmark match counts for Camera 3 with respect to time, when using ADP. Finally, the results shown in Table 4.7 reveal no significant difference in minimum, maximum, mean or standard deviation in landmark track durations across the four processing profiles. This is yet another indicator that

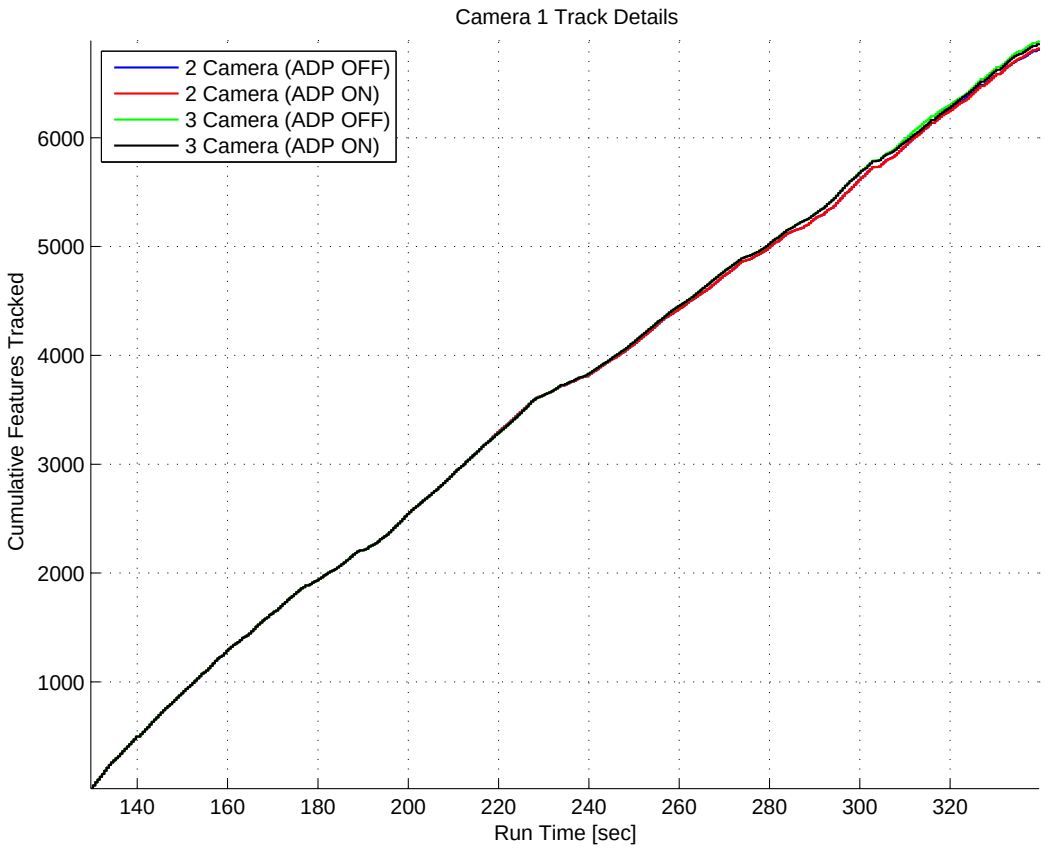


Figure 4.19: Camera 1 landmark matches for the calibration run.

using ADP may not be simply increasing the number of positive matches, but more likely performing a combination of false match reduction and positive match addition, which resulted in a small net positive change with large improvements in navigation accuracy.

4.3.4 Computational Load. Although this research was not aimed at optimizing computations for real-time implementations, it is still important to note the computational load differences between all four processing profiles. Table 4.8 summarizes the computational time required to complete each run using the various processing profiles. As expected, the processing profiles involving three cameras required longer computational times due to the recursive use of the SIFT algorithm. Additionally, the usage of ADP increased the required computational time non-linearly in every case. It is important to note that even with the fastest processing profile

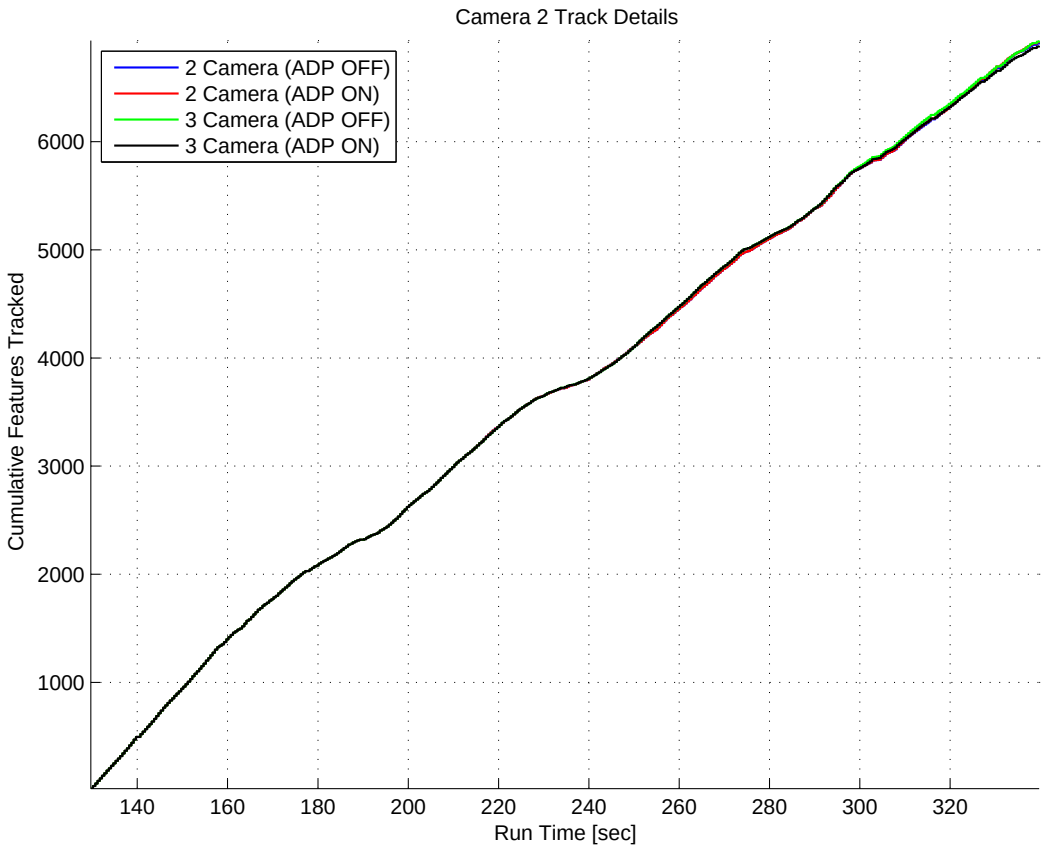


Figure 4.20: Camera 2 landmark matches for the calibration run.

(two camera, no ADP), both runs required significantly more time than the actual duration of the run. Again, this research did not aim at optimizing these algorithms for real-time implementation, but such efforts will be necessary before fully fielding a system equipped with this technology.

4.4 Summary

This section has presented the results and analysis from the experiments described in Chapter III. The automatic calibration algorithm has been shown to not only decrease the required time for calibration, but also vastly reduce the necessary user interaction when compared to the standard calibration technique. Finally, the usage of affine distortion prediction and a side-looking camera has been shown to significantly improve vehicle position, velocity, and attitude estimates by the IAEKF. By

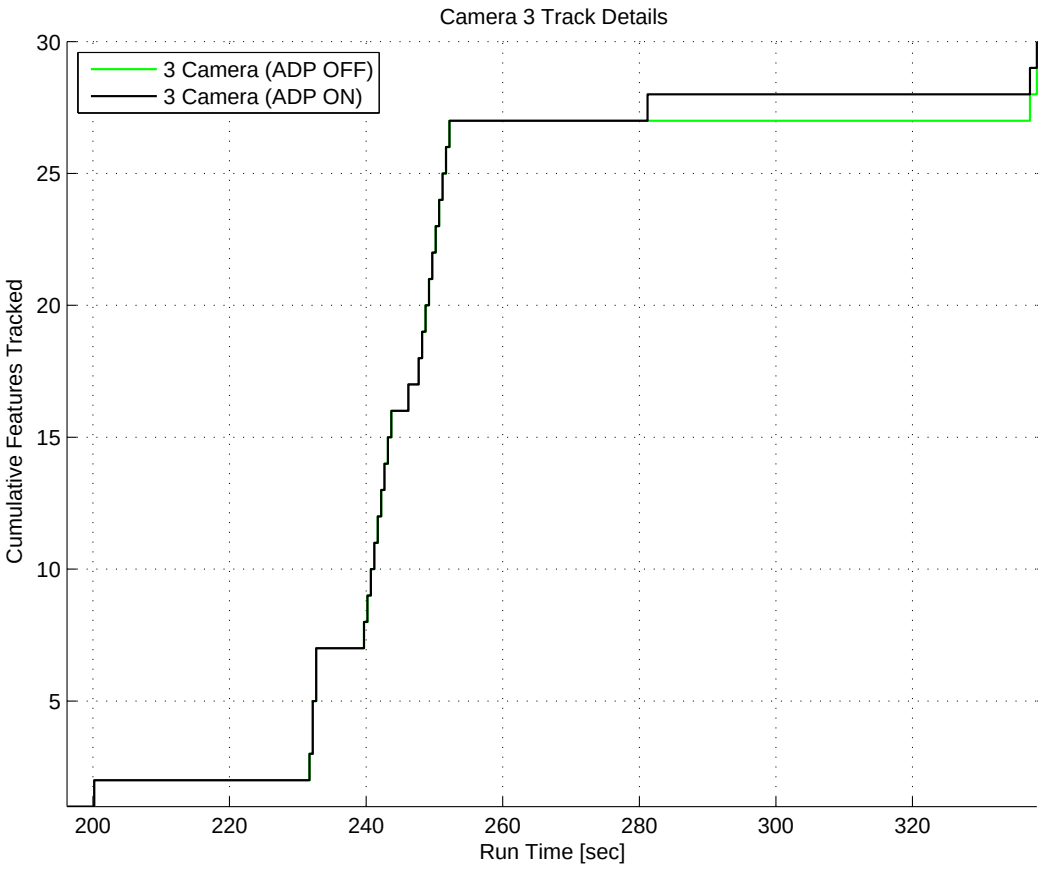


Figure 4.21: Camera 3 landmark matches for the calibration run.

understanding the nature and mathematics behind computer feature matching techniques, two major steps in the image-aided navigation cycle (calibration and feature tracking) have been further matured.

Table 4.6: Landmark match counts for the demonstration run. This table summarizes the cumulative landmark match counts for the demonstration run across all four processing configurations.

	C1	**C2**	**C3**	**C4**
Cameras	2	2	3	3
ADP	Off	On	Off	On
Cam 1	2.06×10^4	2.03×10^4	2.05×10^4	2.08×10^4
Cam 2	2.17×10^4	2.12×10^4	2.20×10^4	2.15×10^4
Cam 3	NA	NA	27	30

Table 4.7: Landmark track statistics for the demonstration run. This table summarizes the landmark tracking duration statistics for the demonstration run across all four processing configurations.

	C1	**C2**	**C3**	**C4**
Cameras	2	2	3	3
ADP	Off	On	Off	On
Min [s]	0	0	0	0
Max [s]	41.50	41.75	41.75	41.75
Mean [s]	1.39	1.39	1.45	1.42
STD [s]	4.04	3.98	4.10	4.06

Table 4.8: Processing times table. This table summarizes the processing times required to complete the two collected runs across all four processing configurations. All times are in [hh:mm:ss].

	C1	**C2**	**C3**	**C4**
Cameras	2	2	3	3
ADP	Off	On	Off	On
Calibration Run	00:06:44	00:10:10	00:18:55	01:18:16
Demonstration Run	00:48:10	06:36:59	01:24:49	14:39:26

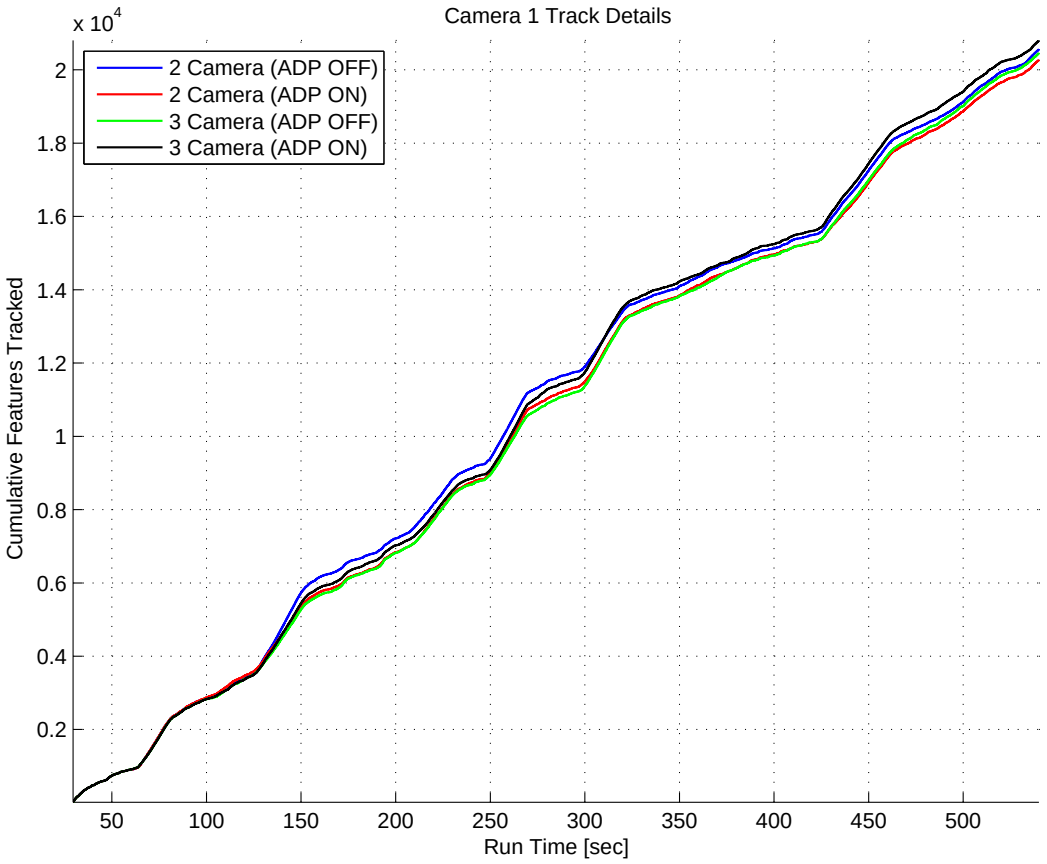


Figure 4.22: Camera 1 landmark matches for the demonstration run.

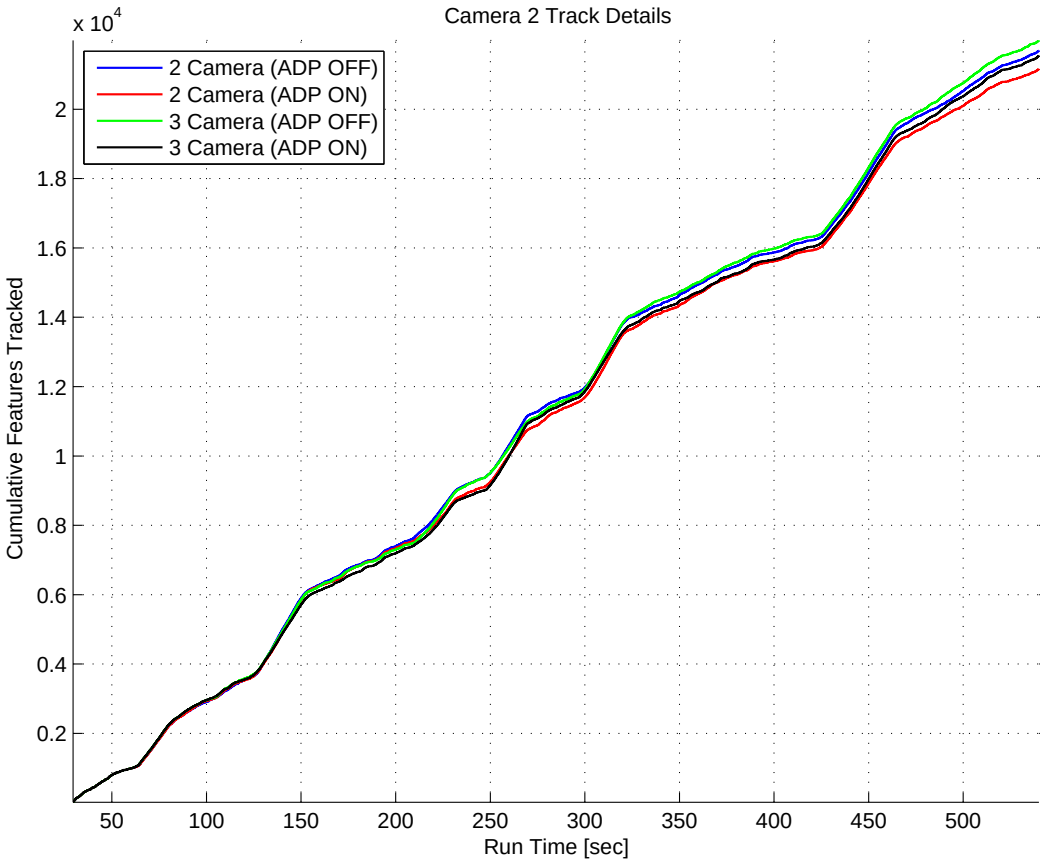


Figure 4.23: Camera 2 landmark matches for the demonstration run.

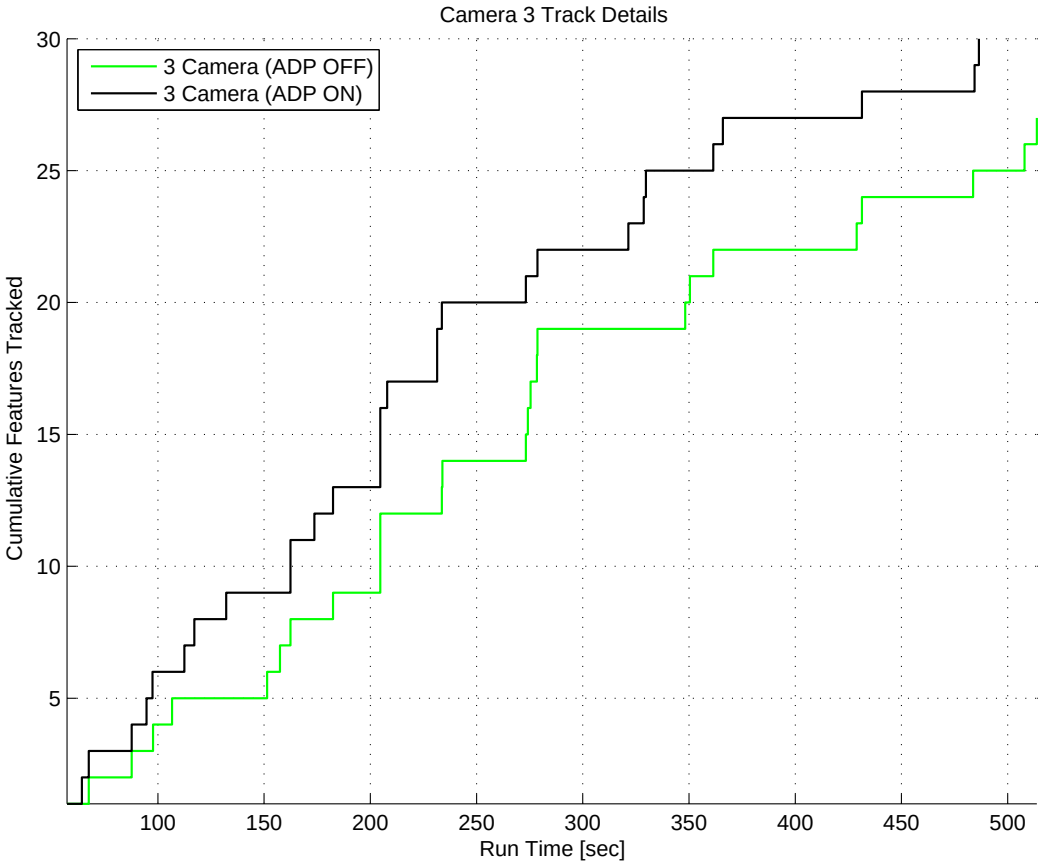


Figure 4.24: Camera 3 landmark matches for the demonstration run.

V. Conclusions and Future Work

This chapter summarizes the information presented in earlier sections of this thesis as well as suggestions for continued improvement through future work.

5.1 Conclusions

As the Air Force's dependency on precision navigation information has increased, so has the interest in overcoming the reliance on GPS. Image-aided navigation, or the coupling of inertial and imaging sensors has grown as a possible alternate precision navigation technology over the past decade and continues to show promising results. This research has focused on improving two key steps in the image-aided navigation process: camera calibration and landmark tracking. Consequently, the two main objectives of the research have been to optimize camera calibration through automation and improve landmark tracking accuracy through affine distortion prediction. Together, these two computer vision techniques allow for precision navigation through the deep, mutualistic coupling of inertial and imaging sensors.

The automatic camera calibration algorithm was developed from an existing manual camera calibration toolbox by automating specific steps which previously required a human in the loop. Specifically, the standard chess-pattern calibration board was substituted with an arbitrary image on which distinct features can be found. Feature matches between the physical and digital copies of the calibration board were automatically found, removing the need for the manual delineation of the calibration board perimeter, as was required by the original toolbox. Having automatically computed the necessary correspondences between points in the calibration board frame with points in the pixel frame, the calibration algorithm was finished as usual, which resulted in a completely automatic camera calibration toolbox. The automatic camera calibration algorithm was presented at the 7th Annual Dayton Engineering Sciences Symposium [8].

A novel affine distortion prediction algorithm was developed in order to counteract the distortions arising from changes in 3-D viewpoint, which significantly feature

matching accuracy. Affine distortions on 2-D images were simulated by assuming all points on a given image are actually three-dimensional and coplanar. Starting with the image-aided navigation extended Kalman filter built by Veth [22] as a baseline, the affine prediction algorithm was used to simulate affine distortions on existing images, based on the relative change in position between a vehicle and the landmarks it tracked. The distorted images were then used to compute new feature descriptors that better approximated those found in new images. In contrast to ASIFT [15], where all possible distortions are simulated for every input image, the affine distortion prediction algorithm only simulated the necessary image distortions due to its deep integration into the EKF. The enhanced IAEKF algorithm with affine distortion prediction was presented at the Position Location and Navigation System Conference [9].

An experiment was developed to evaluate the performance of the automatic calibration and affine distortion prediction algorithms. For the camera calibration algorithm, a binocular set of cameras were calibrated using both the standard (manual) toolbox with the chess-pattern board, and the automatic algorithm with the modified board. The two calibration techniques were compared in terms of total time required, average number of point correspondences, lens coverage, and reprojection errors in the resulting camera model.

For the affine distortion prediction algorithm, two outdoor collection runs were performed using a three-camera setup, augmented with a MEMS-grade INS sensor, and a high-fidelity truth source. The first run was collected using a push-cart around a full parking lot and featured mostly left-hand turns with a right-facing side camera. The second run was collected using a van driving around a downtown urban environment, and featured mostly right-hand turns with a left-facing side camera. The two data collections were then post-processed through the extended Kalman filter, using various processing profiles in which the number of cameras and the usage of affine distortion prediction were individually varied. The filter's results were then compared with the solutions from the high-fidelity truth source for each run in order to determine their accuracy.

The automatic calibration algorithm showed a 70% decrease in required time when compared to the manual calibration. Additionally, it increased the average number of point correspondences used for calibration from 1000 to 5600, which was directly associated with increased lens coverage. Although the average pixel reprojection error was slightly higher for automatic calibration, it was still sub-pixel for both the x and y axes. The significant increase in reprojection error spread was attributed to the decreased feature localization and matching accuracy of the SIFT algorithm when compared to the Harris corner detector.

The affine distortion prediction algorithm significantly improved the accuracy of all nine navigation states for the demonstration run. It was concluded that the calibration run was too short to show any significant differences. Further investigation revealed that using affine distortion prediction resulted in more accurate results due to a combination of reduction in false matches and increase in positive matches. It was also determined that augmenting the standard forward-looking camera pair with a side-looking camera without affine prediction actually increased navigation errors, possibly due to false landmark matches. Using a standard image-aided navigation setup consisting of two forward-looking cameras and no affine distortion prediction as a baseline, the augmented configuration with a side-looking camera and affine distortion prediction provided an average reduction in error of 24% in position, 16% in velocity and 35% in attitude.

5.2 Future Work

Both of the algorithms developed in this thesis could be improved upon in the quest for a reliable, mature alternate precision navigation technology. This section presents some of the possible areas of improvement for future research.

5.2.1 Multi-sensor Extrinsic Calibration. The automatic calibration algorithm presented in this thesis computes the camera model for each individual camera in a system, as well as the relative rotation and translation between each camera using

the first camera's frame as the reference frame. However, most image-aided navigation platforms are equipped with more than just imaging sensors. At a minimum, image-aided navigation as presented herein requires the use of an inertial sensor in addition to the imaging sensors. For this reason, it would be wise to further develop the automatic camera calibration algorithm presented in this thesis into a full-vehicle, multi-sensor calibration process, which provides camera models as well as the relative rotation and translation between every sensor. A vehicle calibration algorithm that is fully automatic would certainly propel image-aided navigation into a new level of maturity and field readiness.

5.2.2 Calibration Board Elimination. Although the automatic calibration algorithm presented in this thesis has eliminated the need for user interaction during the computation phase, a human-in-the-loop is still required to hold a calibration board in front of the cameras to collect the calibration images. The use of a calibration board makes camera calibration relatively simple, but adds a level of manual work that is not practical for field operations. If image-aided navigation and computer vision technology are to be implemented in real-world scenarios, with actual navigation platforms, the automatic camera calibration algorithm should be further developed to eliminate the need for a calibration board. The calibration board could be eliminated by using known features within the field of view of the cameras while still maintaining the automation provided by the SIFT algorithm.

5.2.3 Projective Distortion Modeling. The affine distortion prediction algorithm significantly improved feature matching accuracy by computing new feature descriptors on digitally distorted images. However, the affine camera model on which affine distortion prediction is based is not exactly representative of real-world distortions. In reality, features undergo a projective distortion which has been approximated by the affine model. Projective models include the effects of vanishing points created by the projection of light into the focal point of an imaging sensor. Although affine approximations have been shown to provide significant improvements in feature track-

ing, it would be insightful to investigate the improvements (if any) provided by fully modeling projective distortions.

5.2.4 Modular Landmark Tracking Filter. One of the key computational limitations on the image-aided navigation algorithm developed by Veth [22] is the requirement to augment the extended Kalman filter with three additional states for each landmark being tracked. For this reason, the maximum number of landmarks tracked by any camera was kept to 30 (resulting in 90 additional states). Taking into consideration that the average image contains approximately 900 salient features, only 3% of the available visual information is being used for navigation. However, setting the maximum number of features to even 200 would require the use of 600 additional states, and pose a heavy computational load. Although Veth's implementation [22] provides very deep coupling between the feature tracker and the inertial sensor, it limits (computationally) the maximum number of features that can be tracked. A potentially more computationally-friendly approach would implement feature tracking without requiring additional Kalman filter states in the navigation filter by adding a parallel filter for landmark estimation and tracking. This modular architecture may improve the overall system's ability to grow or shrink the number of landmarks as well as add or remove vision-aiding altogether without significant system re-design. Such an architecture may pave the way towards modular, all source navigation, where users may select a suite of sensors suited for their particular application or mission [5].

5.2.5 Catalog-based Landmark Navigation. Finally, the feature tracker implemented in the current image-aided algorithm provides state measurements which are relative from frame to frame. That is, the standard feature tracker tells the extended Kalman filter how the vehicle moved from one epoch to the next. Consequently, there is a significant possibility of accumulating navigation errors, especially in position and attitude, because there are no absolute measurements being provided. One method of combating such incremental error accumulation is to implement a catalog-based feature tracker in addition to the standard feature tracker. A catalog-based

tracker would provide absolute position measurements based on known landmark locations, or previously tracked landmarks that had relatively low position uncertainty.

5.3 Closing

This research has presented two signal processing algorithms designed to improve the image-aided navigation process. As this technology continues to mature, a reliable, equally-precise alternative to navigation via GPS may be achieved. The necessary processes have been developed; it is now up to the next wave of scholars to propel this theory into a practical, fielded reality.

Bibliography

1. Bouguet, Jean-Yves. "Camera Calibration Toolbox for Matlab". URL `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

2. Brown, Duane C. "Close-Range Camera Calibration". *Photogrammetric Engineering*, Volume 37:8, 855–866. 1971.

3. Committee, DMA WGS-84 Development. *Department of Defense World Geodetic System 1984 - Its Definition and Relationships with Local Geodetic Systems*. Technical Report 8350.2, Defense Mapping Agency, Washington DC, September 1987.

4. Fischler, Martin A. and Robert C. Bolles. "Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Commun. ACM*, 24:381–395, June 1981. ISSN 0001-0782.

5. Fisher, Kenneth A. and John F. Raquet. "Non-GPS Precision Position, Navigation, and Timing". *Air and Space Power Journal*, 25:24–33, 2011.

6. Harris, C. and M.J. Stephens. "A Combined Corner and Edge Detector". *Alvey Vision Conference*, 147–152. 1988.

7. Heikkila, Janne and Olli Silven. *A Four-Step Camera Calibration Procedure with Implicit Image Correction*. Technical report, University of Oulu, 1997.

8. Jurado, Juan D. and Kenneth A. Fisher. "New Techniques in Camera Calibration". *7th Annual Dayton Engineering Sciences Symposium*. October 2011.

9. Jurado, Juan D., Kenneth A. Fisher, and Michael J. Veth. "Inertial and Imaging Sensor Fusion for Image-Aided Navigation with Affine Distortion Prediction". *Position Location and Navigation System Conference*. Myrtle Beach, April 2012.

10. Kalman, R.E. "A New Approach to Linear Filtering and Prediction Problems". *Trans. ASME*, 82D:35–50, 1960.

11. Lindeberg, T. "Scale-space Theory: A Basic Tool for Analyzing Structures at Different Scales". *Journal of Applied Statistics*, 21(2):224:270, 1994.

12. Lowe, David G. "Distinctive Image Features from Scale-Invariant Keypoints". *International Journal of Computer Vision*, 2004.

13. Maybeck, Peter S. *Stochastic Models, Estimation, and Control Volume 1*. Navtech, Virginia, 1982.

14. Maybeck, Peter S. *Stochastic Models, Estimation, and Control Volume 2*. Navtech, Virginia, 1984.

15. Morel, Jean-Michel and Guoshen Yu. "ASIFT: A New Framework for Fully Affine Invariant Image Comparison". *SIAM J. Imaging Sciences*, 2(2):438–469, 2009.

16. Prahl, Dayvid M. *Coupling Vanishing Point Tracking with Inertial Navigation to Estimate Attitude in a Structured Environment*. Master's Thesis, Air Force Institute of Technology, 2011.

17. Schwartz, Gen. Norton. "Opening Keynote Address". *38th IFPA-Fletch Conference on National Security Strategy and Policy*. January 2010.

18. Shi, Jianbo and Carlo Tomasi. "Good Features to Track". *IEEE Conference on Computer Vision and Pattern Recognition*. Seattle, June 1994.

19. Szeliski, Richard. *Computer Vision: Algorithms and Applications*. Springer, 1st Edition, November 2010.

20. Titterton, D. and J. Weston. *Strapdown Intertial Navigation Technology*. Peter Peregrinus Ltd, United Kingdom, 1997.

21. Van Loan, Charles F. "Computing Integrals Involving the Matrix Exponential". *IEEE Transactions on Automatic Control*, Volume 23:3, 395–404. June 1978.

22. Veth, Michael J. *Fusion of Imaging and Inertial Sensors for Navigation*. Ph.D. Thesis, Air Force Institute of Technology, 2006.

23. Veth, Michael J., John F. Raquet, and Meir Pachter. "Stochastic Constraints for Efficient Image Correspondence Search". *IEEE Transactions on Aerospace Electronic Systems*, 42(3):973–982, July 2006.

24. Zhang, Zhengyou. "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations". *The Proceedings of the Seventh IEEE International Conference on Computer Vision*. 1999.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
22–03–2012	Master's Thesis	Sept 2010 — 22 Mar 2012

4. TITLE AND SUBTITLE

Enhanced Image-Aided Navigation Algorithm with Automatic Calibration and Affine Distortion Prediction

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Juan D. Jurado, Capt, USAF

5d. PROJECT NUMBER

12G602

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

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

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GE/ENG/12-23

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

Defense Advanced Projects Research Agency
Strategic Technology Office (DARPA/STO)
3701 N. Fairfax Drive - 3^{rd} Floor
Arlington,VA 22203
Dr. Stefanie Tompkins; (703)248–1540; Stefanie.Tompkins@darpa.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

DARPA

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution A. Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research aims at improving two key steps within the image aided navigation process: camera calibration and landmark tracking. The camera calibration step is improved by automating the point correspondence calculation within the standard camera calibration algorithm, thereby reducing the required time for calibration while maintaining the output model accuracy. The feature landmark tracking step is improved by digitally simulating affine distortions on input images in order to calculate more accurate feature descriptors for improved feature matching in high relative viewpoint change. These techniques are experimentally demonstrated in an outdoor environment with a consumer-grade inertial sensor and three imaging sensors, one of which is orthogonal to the rest. Using a tactical-grade inertial sensor coupled with GPS position data for comparison, the improved image aided navigation algorithm is shown to reduce navigation errors by 24% in position, 16% in velocity and 35% in attitude when compared to the standard image-aided navigation algorithm.

15. SUBJECT TERMS

image aided navigation, feature tracking, camera calibration, affine distortion, feature matching

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maj Kenneth A. Fisher, USAF (ENG)
U	U	U	UU	132	19b. TELEPHONE NUMBER *(include area code)* (937)255–3636, ext 4677; kenneth.fisher@afit.edu