

Air Force Institute of Technology

**AFIT Scholar**

---

Theses and Dissertations

Student Graduate Works

---

9-8-2009

## Electronic Image Stabilization for Mobile Robotic Vision Systems

Michael John Smith

Follow this and additional works at: <https://scholar.afit.edu/etd>



Part of the [Electrical and Electronics Commons](#), and the [Robotics Commons](#)

---

### Recommended Citation

Smith, Michael John, "Electronic Image Stabilization for Mobile Robotic Vision Systems" (2009). *Theses and Dissertations*. 2566.

<https://scholar.afit.edu/etd/2566>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact [richard.mansfield@afit.edu](mailto:richard.mansfield@afit.edu).



ELECTRONIC IMAGE STABILIZATION  
FOR  
MOBILE ROBOTIC VISION SYSTEMS

THESIS

Michael John Smith, Second Lieutenant, USAF

AFIT/GE/ENG/09-53

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

AFIT/GE/ENG/09-53

ELECTRONIC IMAGE STABILIZATION FOR MOBILE ROBOTIC VISION  
SYSTEMS

THESIS

Presented to the Faculty  
Department of Electrical and Computer Engineering  
Graduate School of Engineering and Management  
Air Force Institute of Technology  
Air University  
Air Education and Training Command  
in Partial Fulfillment of the Requirements for the  
Degree of Master of Science in Electrical Engineering

Michael John Smith, B.S.  
Second Lieutenant, USAF


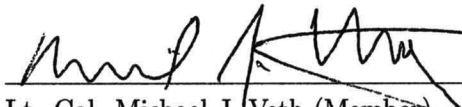
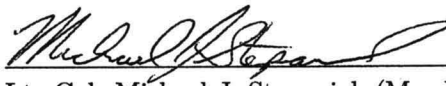
September 2009

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ELECTRONIC IMAGE STABILIZATION FOR MOBILE ROBOTIC VISION  
SYSTEMS

Michael John Smith, B.S.  
Second Lieutenant, USAF

Approved:

	<u>3 SEP 09</u>
Dr. Gilbert L. Peterson (Chairman)	Date
	<u>3 SEP 09</u>
Lt. Col. Michael J. Veth (Member)	Date
	<u>4 Sep 09</u>
Lt. Col. Michael J. Stepaniak (Member)	Date

*Abstract*

When a camera is affixed on a dynamic mobile robot, image stabilization is the first step towards more complex analysis on the video feed. This thesis presents a novel electronic image stabilization (EIS) algorithm for small inexpensive highly dynamic mobile robotic platforms with onboard camera systems. The algorithm combines optical flow motion parameter estimation with angular rate data provided by a strapdown inertial measurement unit (IMU). A discrete Kalman filter in feedforward configuration is used for optimal fusion of the two data sources. Performance evaluations are conducted by a simulated video truth model (capturing the effects of image translation, rotation, blurring, and moving objects), and live test data. Live data was collected from a camera and IMU affixed to the DAGSI Whegs<sup>TM</sup> mobile robotic platform as it navigated through a hallway. Template matching, feature detection, optical flow, and inertial measurement techniques are compared and analyzed to determine the most suitable algorithm for this specific type of image stabilization. Pyramidal Lucas-Kanade optical flow using Shi-Tomasi good features in combination with inertial measurement is the EIS algorithm found to be superior. In the presence of moving objects, fusion of inertial measurement reduces optical flow root-mean-squared (RMS) error in motion parameter estimates by 40%. No previous image stabilization algorithm to date directly fuses optical flow estimation with inertial measurement by way of Kalman filtering.

# *Table of Contents*

	Page
Abstract . . . . .	iv
List of Figures . . . . .	ix
List of Tables . . . . .	xi
1. Introduction to Electronic Image Stabilization . . . . .	1
1.1 Research Motivation . . . . .	2
1.1.1 Robotic Platforms and Their Relation to Non-GPS Navigation . . . . .	3
1.2 Problem Statement . . . . .	4
1.3 Equipment and Testing . . . . .	5
1.4 Research Scope . . . . .	5
1.5 Research Contributions . . . . .	5
1.6 Summary . . . . .	6
2. Electronic Image Stabilization Technical Background and Recent Work	7
2.1 Mathematical Notation . . . . .	7
2.2 Digital Video and Video Motion . . . . .	8
2.2.1 Digital Video Feeds . . . . .	8
2.2.2 Motion Models . . . . .	9
2.3 Causes of Motion Estimation Error . . . . .	13
2.4 Motion Parameter Estimation Using Template Matching . .	15
2.4.1 Error Metrics . . . . .	15
2.4.2 Search Methods . . . . .	17
2.5 Motion Parameter Estimation Using Feature Detection . . .	21

	Page
2.5.1 Hessian Operator . . . . .	21
2.5.2 Sobel Derivative Operator . . . . .	22
2.5.3 Harris Features . . . . .	23
2.5.4 Shi-Tomasi Good Features . . . . .	24
2.5.5 Scale Invariant Feature Transform (SIFT) . . . . .	24
2.5.6 Calculation of Motion Parameter Estimates . . . . .	29
2.6 Motion Parameter Estimation Using Optical Flow . . . . .	33
2.6.1 Sparse Optical Flow . . . . .	33
2.6.2 Dense Optical Flow . . . . .	38
2.7 Motion Parameter Estimation Using Inertial Measurement . . . . .	39
2.7.1 IMU Overview . . . . .	39
2.7.2 First-Order Gauss-Markov Error Process . . . . .	40
2.7.3 Integration and Estimate Determination . . . . .	41
2.8 Other Methods for Motion Parameter Estimation . . . . .	42
2.8.1 Phase Correlation . . . . .	42
2.8.2 Wavelets . . . . .	43
2.9 Outlier Compensation . . . . .	45
2.9.1 Median Filter . . . . .	45
2.9.2 LMedS . . . . .	46
2.9.3 RANSAC . . . . .	46
2.10 Kalman Filtering Overview . . . . .	49
2.11 OpenCV . . . . .	51
2.12 Current EIS Work . . . . .	51
2.12.1 Current Template Matching Methods . . . . .	51
2.12.2 Current Feature Matching Methods . . . . .	52
2.12.3 Current Optical Flow Methods . . . . .	53
2.12.4 Current Inertial Measurement Methods . . . . .	53
2.13 Conclusions on Current Image Stabilization Approaches . . . . .	53

	Page
3. Description of Stabilizer Algorithms . . . . .	55
3.1 Template Matching . . . . .	57
3.1.1 Template Generation . . . . .	57
3.1.2 Template Search . . . . .	57
3.1.3 Global Motion Detection . . . . .	58
3.1.4 Image Registration . . . . .	60
3.1.5 Template Matching Summary . . . . .	62
3.2 Feature Detection . . . . .	62
3.2.1 Feature Matching . . . . .	62
3.2.2 RANSAC Determination of Transformation Matrix . . . . .	62
3.2.3 Feature Detection Summary . . . . .	65
3.3 Optical Flow . . . . .	65
3.3.1 Optical Flow Summary . . . . .	65
3.4 Inertial Measurement . . . . .	65
3.4.1 Inertial Measurement Summary . . . . .	67
3.5 Optical Flow with Inertial Fusion . . . . .	67
3.5.1 Kalman Filter Development . . . . .	72
3.5.2 Optical Flow with Inertial Fusion Summary . . . . .	80
3.6 Summary . . . . .	81
4. Simulation and Experimental Analysis . . . . .	82
4.1 Truth Model Description . . . . .	83
4.2 Evaluation of Non-Inertial EIS Algorithms . . . . .	84
4.2.1 Translation Testing . . . . .	84
4.2.2 Translation and Rotation Testing . . . . .	87
4.2.3 Effects of Blurring . . . . .	89
4.2.4 Effects of Moving Objects . . . . .	89
4.2.5 Non-Inertial EIS Summary . . . . .	89

	Page
4.3 Evaluation of Inertial Algorithms . . . . .	93
4.3.1 Manual Determination of Motion Parameter Estimates	93
4.3.2 Hallway Testing . . . . .	95
4.3.3 Moving Object Simulation . . . . .	98
4.3.4 Inertial Algorithm Summary . . . . .	98
4.4 T-Significance Testing . . . . .	99
4.5 Summary . . . . .	99
5. Conclusions . . . . .	101
5.1 Future Work . . . . .	103
5.1.1 Speeding Up the Algorithm . . . . .	103
5.1.2 Determination of Global Motion Filter Coefficients .	103
5.1.3 Inertial Truth Model Development . . . . .	103
5.1.4 Manual Determination Accuracy . . . . .	103
5.1.5 Detector Value for the Feature Detection Algorithm	104
5.1.6 Numerical Analysis of Other Methods . . . . .	104
5.1.7 Console Bias Determination . . . . .	104
5.1.8 Effects of Angular Rotation . . . . .	104
5.1.9 Use of a Higher Grade IMU . . . . .	104
5.2 Summary . . . . .	105
Bibliography . . . . .	106

## *List of Figures*

Figure		Page
1.1	The EIS Concept . . . . .	2
1.2	The DAGSI Whegs Mobile Platform, with Camera/IMU Setup . . .	4
2.3	Pixel Coordinate Descriptions . . . . .	9
2.4	Effects of Translation and Scaling . . . . .	10
2.5	Homography Transform . . . . .	12
2.6	Two-Dimensional Transforms . . . . .	12
2.7	Template Matching . . . . .	15
2.8	Pixel Numbering System . . . . .	16
2.9	Hierarchical Image Pyramid . . . . .	18
2.10	The Stochastic Constraint Method . . . . .	18
2.11	One-Dimensional Edge . . . . .	20
2.12	Two-Dimensional Corner . . . . .	20
2.13	Sobel Kernel . . . . .	20
2.14	Corner/Edge Classifier . . . . .	23
2.15	Possible Keypoint . . . . .	26
2.16	Octaves of the Difference of Gaussian Functions over a Scale-Space	26
2.17	Heisenberg's Uncertainty Principle. . . . .	43
2.18	One-Dimensional Wavelet Analysis . . . . .	44
2.19	Sample Wavelet Decomposition . . . . .	45
2.20	Kalman Filter Equations . . . . .	50
3.21	Platform Coordinate Frame . . . . .	55
3.22	Template Matching Block Diagram . . . . .	56
3.23	Bode Response of the LPF . . . . .	59
3.24	Step Response of the LPF . . . . .	60

Figure		Page
3.25	Feature Detection Block Diagram . . . . .	61
3.26	Optical Flow Block Diagram . . . . .	64
3.27	Inertial Measurement Block Diagram . . . . .	66
3.28	IMU Gains Determination . . . . .	68
3.29	The EIS Concept . . . . .	69
3.30	Motion Estimation Block . . . . .	69
3.31	Optical Flow with Inertial Fusion Block Diagram . . . . .	70
3.32	Angular Rate Drift . . . . .	73
3.33	Example Autocorrelation and Parameter Determination . . . . .	73
3.34	Example Angular Rate Output . . . . .	74
3.35	Kalman Filter Equations . . . . .	78
3.36	R Value Error Plot . . . . .	79
3.37	Determination of Console Clock Bias . . . . .	80
4.38	Base Image for Truth Model . . . . .	84
4.39	Translation Test Performance . . . . .	85
4.40	Template Matching Estimation Error, Short Translation Test . . . .	86
4.41	Translation and Rotation Test Performance . . . . .	88
4.42	Translation, Rotation, and Blur Test Performance . . . . .	90
4.43	Translation, Rotation, Blur, and Moving Object Test Performance .	91
4.44	Video with Moving Object . . . . .	92
4.45	Hand Determination Snapshot . . . . .	92
4.46	Hallway Test Performance . . . . .	94
4.47	Hallway Test with Moving Object . . . . .	96
4.48	Hallway with Moving Object Test Performance . . . . .	97

## *List of Tables*

Table		Page
3.1	Autocorrelation Parameter Values . . . . .	74
4.2	Translation Test Performance . . . . .	85
4.3	Average Time for One Estimation Loop . . . . .	86
4.4	Translation and Rotation Test Performance . . . . .	88
4.5	Translation, Rotation, and Blur Test Performance . . . . .	90
4.6	Translation, Rotation, Blur, and Moving Object Test Performance .	91
4.7	Hallway Test Performance . . . . .	95
4.8	Hallway with Moving Object Test Performance . . . . .	96

# ELECTRONIC IMAGE STABILIZATION FOR MOBILE ROBOTIC VISION SYSTEMS

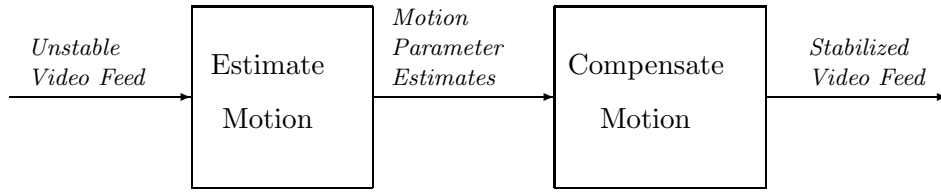
## I. Introduction to Electronic Image Stabilization

**S**tabilization is the process by which undesired output fluctuations are removed from a system. Stabilization is found in many areas of life. A robotic manipulator uses stabilization to maintain control of its mechanical arms. Traction controls on automotive vehicles stabilize the acceleration forces on the tires. The nation's economic leaders use financial stabilization tools to keep the American markets under control. Even the simple act of eating a bowl of cereal requires active biological stabilizers to successfully bring food from bowl to mouth.

Image stabilization is a specific type of stabilization with versatile and necessary use. Inherent to image stabilization is vision. For stationary vision systems image stabilization is unnecessary, but the vast majority of visual systems are mobile. Both biological and engineered vision systems have mobile elements which require stabilization. Walking down the street would be very painful without a natural internal image stabilizer allowing for accurate determination of object location and depth to aid in notifying you of a potential collision. A camera attached to an Unmanned Aerial Vehicle (UAV) requires image stabilization for the clearest video sequences.

Electronic image stabilization (EIS) is a specific type of image stabilization involving signal processing on a digital camera video feed. Other types of image stabilization are mechanical, involving the physical movement of the camera, and optical, involving the manipulation of the camera lenses [1:3-8]. This thesis focuses on EIS and its applications to highly dynamic mobile robotic platforms with onboard camera systems.

The most familiar form of EIS is found on commercial digital photographic cameras and digital video cameras. These stabilizers are used to counteract hand jitters which can



**Figure 1.1. The EIS Concept. Performing EIS involves the conversion of an unstable video feed into a stabilized video feed.**

ruin a photo or video. In addition, EIS is used for video overlay algorithms. If certain objects in a video feed are marked using an overlay, it is necessary for the overlay to move with the object for correct classification. EIS is used to estimate the movement of the images and correct the overlay as necessary. EIS is also used as a first step towards more complex video analysis. Object shape characterization, object detection and tracking, and simultaneous localization and mapping (SLAM) algorithms, among others, all require stabilized video in order to operate most effectively.

The aim of EIS is the elimination of unwanted camera motion effects from an image feed. It seeks to convert an unstable image feed into a stable image feed. EIS is comprised of two main functions, shown in Figure 1.1. These are motion estimation, followed by motion compensation. In the motion estimation stage, the physical movement of the video image pixels between frames are defined within the parameters of a particular motion model. Once a set of parameter estimates is found frame to frame, compensation is applied to counteract the perceived motion. This is done using digital warping of the image. The result is a stable image feed.

### ***1.1 Research Motivation***

The motivation for this thesis is non-Global Positioning System (GPS) navigation systems. Precision navigation in non-GPS coverage areas is becoming a necessity in modern combat operations. As warfare continues to move into urban areas, GPS navigation is becoming less reliable. The line-of-sight vector between the GPS receiver and the satellite is often blocked in urban areas due to buildings and other obstructions. Enter a building and the GPS signal is severely degraded or lost entirely. Further, the susceptibility of

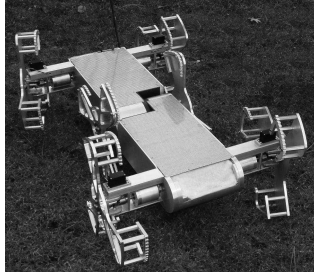
the GPS signal to jamming, intentional or unintentional, is an ever present concern [2:43]. These problems in GPS availability necessitate the development of non-GPS navigation systems.

The advantages of non-GPS navigation is an increase in military asset mobility and control, and additional navigation redundancy where GPS is available. It provides the capability to track and monitor people and material in non-GPS coverage areas where they currently cannot be tracked or monitored. Non-GPS navigation can be used to guide UAV's into regions of the urban air space they could not enter previously. Non-GPS navigation can also be used to guide teleoperated robotic platforms and Unmanned Ground Vehicles (UGV's) into regions of the battlefield where they currently cannot venture due to GPS constraints. In the event of sudden GPS denial or degradation on the battlefield, non-GPS navigation systems can ensure continued precision navigation.

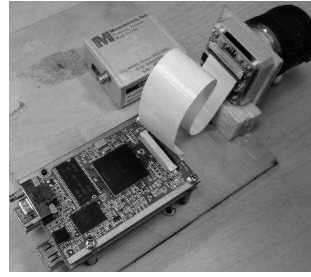
*1.1.1 Robotic Platforms and Their Relation to Non-GPS Navigation.* In recent years, it has become apparent that robotic platforms have a variety of beneficial uses to the combat soldier. They can be used for reconnaissance or as transport vehicles; they can be used as decoys; or they can be armed with munitions and engage the enemy, among other uses. The use of these valuable systems in GPS degraded areas may reveal new combat capabilities.

Recent developments in biologically inspired robotic platforms have allowed these systems to physically venture into areas robots have never gone before. Snake mimicking robots can navigate in and around very confined spaces. Insect mimicking robots can crawl or fly into any small opening. Walking robots can traverse large obstacles. The EIS algorithm developed for this thesis operates on the video feed of a camera mounted on the DAGSI Whegs<sup>TM</sup> mobile robotic platform [3], shown in Figure 1.2. This particular platform is capable of climbing stairs, rocks, and other difficult terrain. Ordinary wheeled platforms simply cannot navigate in environments where these types of biologically inspired robots thrive.

It has been proven that non-GPS navigation is achieved by optical and inertial sensor fusion [4]. However, this kind of complex video analysis requires a stabilized video feed in



(a) DAGSI Whegs



(b) Camera/IMU Setup

**Figure 1.2. The (a) DAGSI Whegs Mobile Platform, with  
(b) Camera/IMU Setup.**

order to achieve the best performance. Optical sensors mounted on biologically inspired robotic platforms have one major drawback; they undergo highly dynamic motion. Thus, if non-GPS navigation is implemented on one of these highly dynamic mobile robotic platforms, image stabilization is necessary. The EIS algorithm developed in this thesis is the preparatory step required for this type of navigation solution for highly dynamic mobile robotic platforms.

Biologically inspired robotic platforms using non-GPS navigation will allow the entrance of combat capabilities into areas of the battlefield never thought possible. Explosive devices could be sent secretly up sink drains to destroy enemy assets. Ammunition and supplies could be sent across a pile of rubble impassible by humans to the people who need it. An audio sensor could secretly find its way under the sofa of a prominent enemy leader and gather valuable intelligence. The list is endless. The integration of these types of robotic platforms with non-GPS navigation will allow our military forces to go farther and accomplish more than ever before.

## **1.2 Problem Statement**

The problem statement for this work is to develop an effective EIS algorithm capable of use on highly dynamic mobile robotic platforms. When traversing difficult terrain, the video feed undergoes a substantial amount camera movement, noise, and imaging effects. These aspects of dynamic motion make complex video feed analysis difficult to achieve.

An effective EIS algorithm is necessary in order to accurately utilize non-GPS navigation on these systems.

The use of the developed EIS algorithm is not contained to the video feed only. If other sensors are rigidly attached to the platform and face the same direction, the frame to frame motion estimates can be applied to them as well.

### ***1.3 Equipment and Testing***

The equipment used in this work is the Pixelink PL-A741 monochrome video camera, operating at 7 frames per second, and the MIDG IMU, operating at 50 Hz. Both devices are mounted on the front of the DAGSI Whegs<sup>TM</sup> mobile platform. These devices are shown in Figure 1.2. Testing was accomplished inside the Air Force Institute of Technology, located at Wright Patterson Air Force Base, Ohio.

### ***1.4 Research Scope***

This research develops the theory and numerically analyzes several current EIS techniques. An in depth presentation of applicable theory is provided, laying the foundational understanding of the four major classes of EIS; template matching, feature detection, optical flow, and inertial measurement. Algorithms representative of each of these methods are developed and their accuracies compared against a video truth model whose motion parameter values are known. Further, using the camera and IMU affixed to the DAGSI Whegs<sup>TM</sup> mobile platform, a live data collect was performed. This allows for real world performance analysis of the algorithms. Algorithm computation time is also compared.

### ***1.5 Research Contributions***

The main contribution of this thesis is a novel EIS algorithm capable of use on highly dynamic mobile robotic platforms. Optical sensor information is fused with inertial sensor information by way of Kalman filtering. Specifically, pyramidal Lucas-Kanade optical flow using Shi-Tomasi good features [5] is fused with inertial data provided by an IMU by way of a discrete Kalman filter. Previous work has used inertial systems to initialize EIS

algorithm motion estimates [6, 7, 8], but none have directly fused the two data sources by way of a discrete Kalman filter.

The secondary contribution of this work is the creation of an image stabilization algorithm that operates on the DAGSI Whegs<sup>TM</sup> platform. To date no image stabilization algorithm has been developed for this specific platform.

The third contribution of this work is the discovery of an algorithm that reduces the errors caused specifically by moving objects and image blurring. To date, these two effects have never been specifically addressed by EIS techniques.

The fourth contribution of this work is a numerical analysis comparing the performance of the four main classes of EIS. Template matching, feature detection, optical flow, and inertial measurement algorithms are evaluated and compared using video truth models in a fair and standard fashion. To date, a numerical comparison between these methods has never been done.

## **1.6 Summary**

Image stabilization is a necessary and exciting field of study, with applications in many different areas of life. Electronic image stabilization in particular is useful as the preparatory step for non-GPS navigation algorithms utilized on highly dynamic mobile robotic platforms. Integration of pyramidal Lucas-Kanade optical flow using Shi-Tomasi good features with inertial data from an IMU by way of a discrete Kalman filter is shown to be an effective EIS algorithm in this type of application.

This thesis is organized as follows. The second chapter addresses the necessary technical background information and related work pertaining to EIS. The third chapter details the EIS algorithms developed for this thesis. The fourth chapter presents the numerical analysis of the algorithms. Chapter five concludes the thesis with final results and ideas for future work.

## II. Electronic Image Stabilization Technical Background and Recent Work

**E**lectronic image stabilization is a subject of study which covers topics within the fields of computer vision, inertial navigation, and Kalman filtering. Several aspects of calculus, linear algebra, and statistics are necessary for a full and complete understanding of EIS and its methods. A brief introduction to EIS was given in chapter one. This chapter presents an in depth review of the relevant technical background. Knowledge of the topics covered in this chapter will prepare the reader for the algorithm descriptions in chapter three.

First the mathematical notation used in this thesis is explained. Then the fundamental topics of video and video motion are discussed. Next, causes of EIS errors are described. The four different motion parameter estimation techniques, template matching, feature detection, optical flow, and inertial measurement, are then presented. Outlier compensation techniques are given next, followed by a description of Kalman filtering and the OpenCV programming library. The chapter concludes with current work on EIS.

### *2.1 Mathematical Notation*

To provide insight regarding the mathematical equations at a quick visual inspection, the notation for this thesis is as follows:

**Scalars:** Scalars are denoted by lower or uppercase unitalicized Times New Roman font, for example  $h$  or  $D$ .

**Vectors:** Vectors are denoted by lower or uppercase Helvetica font, for example  $\mathbf{x}$  or  $\mathbf{T}$ .

**Matrices:** Matrices are denoted by boldface Bookman font, for example  $\mathbf{F}$ .

**Dimensional Parameters:** Dimensional parameters are denoted by lowercase italicized font, for example  $(x, y)$ .

**Estimated Variables:** Estimated variables are given a *hat* character directly above the variable, for example  $\hat{\mathbf{x}}$ .

**Functions:** Functions are denoted by uppercase unitalicized Times New Roman, with their parameters depicted in parentheses, for example  $E_{SSD}(\mathbf{u})$ .

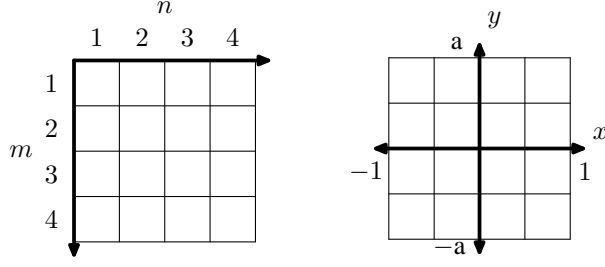
All other notation is straightforward and understood from context.

## 2.2 *Digital Video and Video Motion*

*2.2.1 Digital Video Feeds.* The video feed is the most important source of data for EIS. The video feed is made up of a consistent input stream of images. An image consists entirely of pixels, and it is the pixel and its manipulation which is the major concern in EIS. Two important descriptions of an image pixel are its intensity value and its location in the image. Pixel intensity is the numerical value assigned to a pixel, representing the amount of light entering into the camera lens at that point [4:35-37]. The pixel intensity is then matched to a particular color value using a color map. For example, with the grayscale color map, high pixel intensity is conveyed with white color, and low pixel intensity is conveyed with black color. Often color pictures are encoded using three separate sub images, each called an image channel, with each channel reflecting intensity values for one of three color maps. Use of a red color map, a green color map, and a blue color map comprise the well known RGB color format. The three image channels are portrayed on screen at the same time, and their combination forms the complete color picture.

Pixel location is the position of a pixel within an image. Pixel location is often described using raster coordinates, using the ( $m$  row,  $n$  column) description, where the origin lies in the top left corner of the image. This description works well for simple image and pixel manipulation such as translation, but for more complicated motion such as image rotation, Cartesian coordinates are used. Cartesian coordinates are expressed by an  $x$ - $y$  axis whose origin rests at the center of the image.

An additional step is normalization of the coordinates. Normalizing coordinates allow for the handling of images at different resolutions. The longer image axis is normalized to  $[-1, 1]$  and the shorter axis is normalized to  $[-a, a]$ , where  $a$  is the inverse of the aspect



**Figure 2.3. Pixel Coordinate Descriptions.** The left figure displays raster coordinates. The right figure displays Cartesian coordinates.

ratio [9:2]. Both raster and normalized Cartesian coordinate descriptions are shown in Figure 2.3. Aspect ratio is described as

$$\text{aspect ratio} = \frac{1}{a} = \frac{H}{W} \quad (2.1)$$

where  $H$  is the height of the image and  $W$  is the width.

Given an image pixel coordinate  $(m, n)$ , the normalized Cartesian coordinate pair becomes

$$x = \frac{2m - W}{S} \quad \text{and} \quad y = \frac{2n - H}{S} \quad (2.2)$$

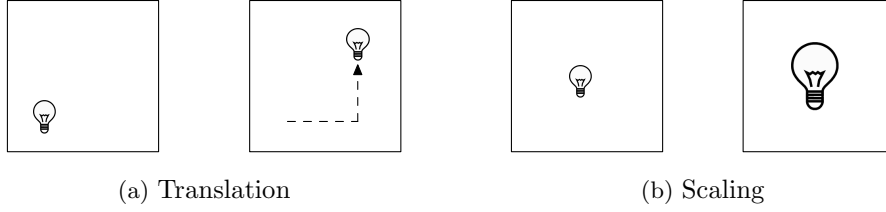
where  $S = \max(W, H)$ . The  $(x, y)$  coordinate pair is referred to in matrix form as  $\mathbf{c}$ , where

$$\mathbf{c} = \begin{bmatrix} x \\ y \end{bmatrix}. \quad (2.3)$$

The homogeneous representation is  $\tilde{\mathbf{c}}$ , where an untransformed image point  $(x, y)$  is represented as

$$\tilde{\mathbf{c}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \quad (2.4)$$

**2.2.2 Motion Models.** The motion model is the expression of pixel movement in the video feed. It is a transformation frame to frame of every pixel in the image. There are



**Figure 2.4. Effects of Translation and Scaling.** The left images portray a positive  $x$  translation and a positive  $y$  translation. The right images portray positive scaling.

several types of motion models. Simple motion models have few parameters to consider, and thus have a computational advantage, but require certain assumptions on the type of scenes depicted in the video frames for accurate use. Complex motion models have many parameters and can correctly describe complex motions, but are computationally expensive. Two-dimensional models are the simplest, but assume that the video feed is receiving images from a flat surface scene. This is a reasonable assumption when pixel movement is determined primarily by the camera motion (the scene is sufficiently far from the camera), but when camera movement is less dominant, three-dimensional scene effects become more important, and the two-dimensional model loses effectiveness. The most common two-dimensional motion models are described below [9:3-4].

*Translation:* This model describes movement in the  $x$  and  $y$  directions only. It assumes that frame to frame there is negligible rotation of the image, and negligible scaling (Figure 2.4 shows the effects of translation and scaling). It is governed by the equation

$$\mathbf{c}^+ = \begin{bmatrix} \mathbf{I} & \mathbf{T} \end{bmatrix} \tilde{\mathbf{c}}^- \quad (2.5)$$

where

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_x \\ \mathbf{T}_y \end{bmatrix} \quad (2.6)$$

is the pixel translation in the  $x$  and  $y$  directions, and  $\mathbf{I}$  is the  $2 \times 2$  identity matrix. The  $-$  and  $+$  superscripts refer to the pixel coordinates before and after the transformation,

respectively. Note that this transformation converts a homogeneous coordinate into an inhomogeneous coordinate. The same is true for the rigid body model, discussed next.

*Rigid Body:* This model accounts for both translation and rotation of the image, and is also known as the Euclidean transformation since Euclidean distances are maintained. It assumes negligible scaling, and is described as

$$\mathbf{c}^+ = \begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix} \tilde{\mathbf{c}}^- \quad (2.7)$$

where the rotation matrix  $\mathbf{R}$  is

$$\mathbf{R} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}. \quad (2.8)$$

The  $\alpha$  parameter is the rotation angle of the image. The sign of the angle  $\alpha$  is positive in the counter-clockwise direction.

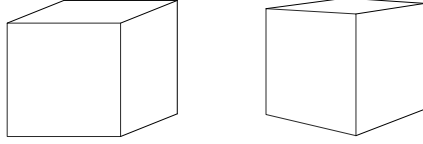
*Similarity:* This model accounts for translation, rotation, and image scaling. It preserves angles between lines in the image. The similarity transform is described as

$$\mathbf{c}^+ = \begin{bmatrix} \lambda \mathbf{R} & \mathbf{T} \end{bmatrix} \tilde{\mathbf{c}}^- \quad (2.9)$$

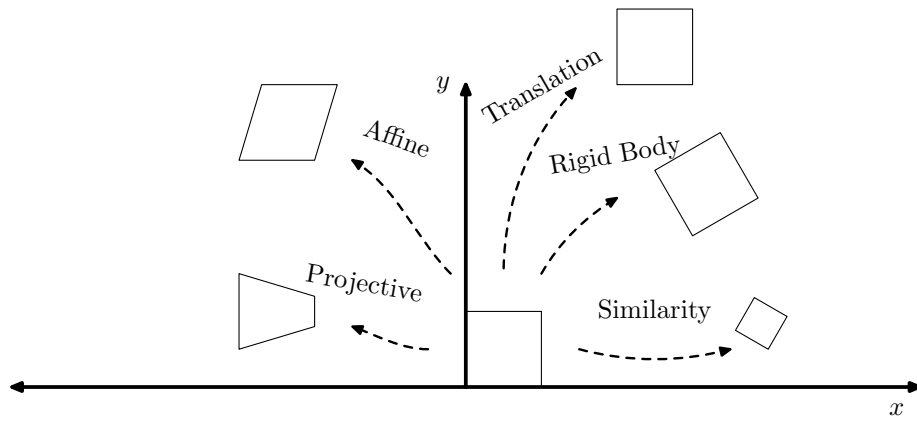
where  $\lambda$  is the scale factor for the transform. The similarity model is the model chosen for the development of the algorithms in chapter three, due to two desirable characteristics. Besides computational speed, the similarity model also only requires two pixel matches between frames to be calculated accurately. This will be shown later in Section 2.5.6. This transformation also requires a homogeneous coordinate and returns an inhomogeneous coordinate.

*Affine:* The affine transformation is a general transform that contains all of the above three more specific transforms. It is any  $2 \times 3$  matrix with arbitrary values for the variables  $a_{11} \dots a_{23}$ . The affine transformation preserves parallel lines, and is expressed as

$$\mathbf{c}^+ = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \tilde{\mathbf{c}}^- . \quad (2.10)$$



**Figure 2.5. Homography Transform.** As the cube is rotated, the front face of the cube has transformed shape with respect to the camera.



**Figure 2.6. Two-Dimensional Transforms.** A square undergoes several transformations: translation, rigid body, similarity, affine, and projective.

*Homography:* The homography, projective, or perspective transform, is even more general than the affine transform. It is represented by an arbitrary  $3 \times 3$  matrix comprised of components  $h_{11}...h_{33}$ . The homography preserves straight lines, both horizontal and vertical. It is used when characterizing motion in a three-dimensional environment, where image depth is important. Figure 2.5 shows the homography transformation for the face of a rotating cube. The transform is expressed as

$$\tilde{\mathbf{c}}^+ = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tilde{\mathbf{c}}^- . \quad (2.11)$$

To summarize, Figure 2.6 portrays graphically the different types of two-dimensional transforms.

### 2.3 Causes of Motion Estimation Error

Before presenting how motion estimation is performed, the major causes of estimation error will first be discussed. Moving objects, blurring, washout, occlusion, parallax, and shot noise are all effects that occur within video feeds. They are the major causes leading to inaccurate motion parameter estimation. These effects are due in part to the camera hardware and also due in part to the scene environment of the camera. At any given time, an EIS algorithm will experience some of these errors. The mark of a robust EIS algorithm is its resistance to estimation error in the presence of these effects.

A *moving object* is an object in the video feed with different motion than the background. Feature detection algorithms function very poorly in the presence of moving objects. When the surface of a moving object has strong corners, feature correspondences associated with the moving object are created. Because the motion of the object is not equivalent to the motion of the image, determination of the motion model parameters will be inaccurate if these correspondences are used to calculate the transformation matrix.

*Blurring* is caused when camera motion is too quick for the image capture rate of the camera. Objects are registered onto a larger pixel area than their true representation. The

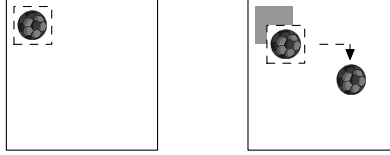
result is pixel intensity smearing within certain regions of the image. Bright objects which have high pixel intensity values, such as light fixtures, tend to bleed their pixel intensity into the surrounding pixels. Blurring makes features difficult to detect in an image by covering over potential corners.

*Washout* occurs when the video images become one uniform color. This is caused when the camera approaches closely to an object of solid color, such as a wall. Without strong gradients in the spatial scene, features are difficult to detect. Washout reduces the number of feature points that can be found in an image.

*Occlusion* occurs when part of the image scene creates edges that are caused by objects that are not depth-continuous. For example, consider two large buildings, one in the foreground and one in the distance. Set the camera so that both buildings can be seen, with the foreground building slightly overlapping the background building. As the camera moves laterally across the front building, the edge created between the front building and the back building will move. This edge is not associated with any real physical surface on an object, and is an unreliable feature point. The edge will disappear altogether as the foreground building passes completely in front of the background building. Also, the movement of the edge is not necessarily dependent upon camera motion. For these reasons, occlusions create unreliable features.

*Parallax* occurs as a camera moves laterally across a scene of objects with different depths. Objects far away appear to move slower than objects close to the camera. Features being tracked on distant objects will move slowly compared to features on closeup objects. This inconsistency makes determination of the transformation matrix inaccurate for simplified motion models such as the similarity model.

*Shot noise* is random Gaussian distributed error which is introduced onto random pixels in the image. It is due to anomalies in the video camera hardware and the intensity radiance of the scene objects. Shot noise can reduce the accuracy of template matching error scores, or can be misinterpreted as a strong feature. Either way, the result is a less accurate motion estimate.



**Figure 2.7. Template Matching.** A template is taken from the current frame and applied throughout the next frame. When the template match is found, the template displacement is noted and this is equivalent to the frame movement itself.

## ***2.4 Motion Parameter Estimation Using Template Matching***

Given two images, each affected by the above causes of error, there are several techniques available to determine the transformation matrix accurately characterizing the motion between the frames. There are four main classes of estimation algorithms; template matching, feature detection, optical flow, and inertial measurement. Each class has several variant algorithms, and combinations of techniques are also used.

In template matching algorithms [9:11-26], a small template window is copied out of the image, and this location is stored. On the next image, the template is moved up, down, left, and right, until a suitable match is found, as portrayed in Figure 2.7. The template movement required for the match is the image displacement. Two primary choices must be made for template matching; the error metric and the search method to use.

*2.4.1 Error Metrics.* The error metric determines how the best template match location is found. Some form of sum of squared difference or cross correlation are most commonly used.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

**Figure 2.8. Pixel Numbering System.** A template is comprised of pixels which can be numbered sequentially.

*2.4.1.1 Sum of Squared Difference (SSD).* This approach takes the intensity values of the template and subtracts them from the intensity values of the image at the particular template location [9:14]. This is expressed as

$$E_{SSD}(\mathbf{u}) = \sum_k \left[ \mathbf{I}_I(\mathbf{c}_k) - \mathbf{I}_T(\mathbf{c}_k + \mathbf{u}) \right]^2 = \sum_k e_k^2 \quad (2.12)$$

where  $\mathbf{I}_I$  represents the large image, and  $\mathbf{I}_T$  represents the template image. The vector  $\mathbf{u}$  is the displacement of the template from its original position, and  $\mathbf{c}_k$  is the location in the large image for pixel number  $k$ . Figure 2.8 shows an orientation for numbering the pixels in the template image. The last term,  $e_k$  is the residual error between the pixels.

A particular  $E_{SSD}$  value for a displacement  $\mathbf{u}$  is found by the following: each pixel in the template is subtracted from the corresponding pixel in the larger image. This is the residual error, which is squared. All of the pixel square residuals in the image-template correspondences are then added, and this value is the  $E_{SSD}$ .

The value  $\mathbf{u}$  which yields the lowest  $E_{SSD}$  is the best estimate for the true displacement of the frame. This least-squares approach is the optimal estimate when dealing with Gaussian noise, and thus most often used. However, depending on the vision environments, better results will be obtained using different error metrics.

*2.4.1.2 Cross Correlation Function.* Instead of minimizing, cross correlation seeks the maximum  $E_{CC}(\mathbf{u})$  value [9:18]. The cross correlation between the template and the image is found from

$$E_{CC}(\mathbf{u}) = \sum_k \left[ \mathbf{I}_I(\mathbf{c}_k + \mathbf{u}) * \mathbf{I}_T(\mathbf{c}_k) \right] \quad (2.13)$$

where  $*$  is the two-dimensional convolution operator.

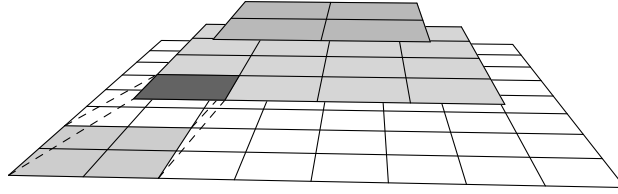
This process is sped up significantly by using the Fast Fourier Transform (FFT) [9:15]. Mathematically, the FFT version is

$$E_{CC}(\mathbf{u}) = \mathcal{F}^{-1} \left\{ \mathcal{I}_I(f) \mathcal{I}_T^*(f) \right\} \quad (2.14)$$

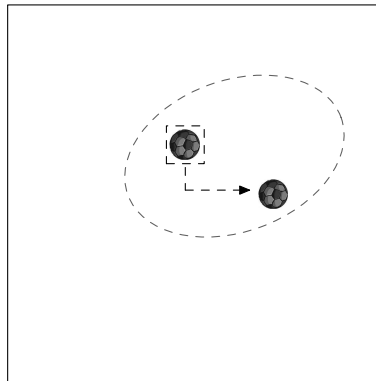
where  $\mathcal{I}_I(f)$  is the Fourier transform of the large image, and  $\mathcal{I}_T^*(f)$  is the complex conjugate of the Fourier transform of the template image.

*2.4.2 Search Methods.* Given a template and an image, the search method dictates the process by which the template is run across the image.

*2.4.2.1 Full Search.* In full search, the template is run across every single pixel in the image. The location yielding the best error value is used. This incurs a large computational cost. For an  $m \times n$  sized image, and an  $k \times l$  sized template, the computational cost is  $O(mnkl)$ . As the template increases in size to the complete image, the cost approaches  $O(mn)^2$  [9:14].



**Figure 2.9. Hierarchical Image Pyramid.** Each level up in the pyramid is a downsampled version of the level below it. The intensity values of the four pixels in the bottom level are averaged to produce the intensity value of the pixel above it.



**Figure 2.10. The Stochastic Constraint Method.** A gyro sensor is used to initialize and bound the template search area to within one standard deviation in the  $x$  and  $y$  directions.

*2.4.2.2 Hierarchical Image Pyramid.* A much faster method is the hierarchical pyramid approach, shown in Figure 2.9. In this method, a pyramid of down-sampled images is constructed [10]. The lowest level of the pyramid is the unaltered image. In the next level up, the image is downsampled by two. In each subsequent level the image is progressively downsampled. For example, if a  $256 \times 256$  image is used, the lowest level will be the unaltered image, and the next level up will be a  $128 \times 128$  image. Each pixel value at a particular level is the mean value of the four pixels below it in the lower level. The downsampling continues until a level is reached which does not benefit the template search.

Once an image pyramid is constructed, a template pyramid is made. Then, using the highest level of the image and template pyramids, a full search is implemented. Once the best match is found for a higher level, the displacement value is used to initialize the level underneath it. As long as the highest level displacement estimate is accurate, each subsequent lower level only requires a search space of two pixels. Thus with a pyramid approach computation is significantly reduced.

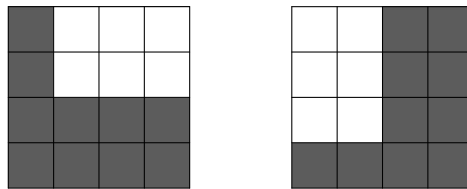
*2.4.2.3 Stochastic Constraint Method.* The stochastic constraint method, shown in Figure 2.10, constrains the search space on the image by predicting the location for the template match. In [4:139-141], the stochastic constraint method is used to limit the search space for feature matching discussed next, but it can be used for template searching as well. The angular displacements frame to frame, output by the gyro sensor, are converted to pixel movement. The search space is then constrained to  $\pm 1$  standard deviation around the the template. In optical and inertial sensor fusion, this process is used in what is known as tightly coupled integration. It results in a fast and accurate template search.

Template matching provides an intuitive process of motion estimation that is simple to implement. Some forms of template matching are more computationally expensive than others, but for many EIS needs, template matching is a viable option.

142	142	76	76
-----	-----	----	----

142	142	142	142
-----	-----	-----	-----

**Figure 2.11. One-Dimensional Edge.** The left line depicts a distinct edge. The right line has no edge.



**Figure 2.12. Two-Dimensional Corner.** Image rotation is determined by applying a two-dimensional corner detector.

a	b	c
d	e	f
g	h	i

**Figure 2.13. Sobel Kernel.** This  $3 \times 3$  matrix is convolved with an image to determine directional derivatives.

## 2.5 Motion Parameter Estimation Using Feature Detection

An alternative to template matching techniques is feature detection. Frame displacement is determined from tracking prominent and unique points in successive frames.

The fundamental concept of feature detection is the one-dimensional edge. The edge is any change in value along a line as. An edge is shown in Figure 2.11, which depicts a line of pixel intensities in an image.

Applying a local derivative along the line is a simple solution to detect this type of edge. This is effective for discontinuous lines. The derivative returns a zero value along the line until it comes across the discontinuity. At the discontinuity, the derivative will spike and yield a value of high magnitude. It will then immediately return to zero along the rest of the line.

If this derivative detector is implemented on a line of steady gradient, an issue arises. There are no points of prominence along the line, yet the derivative will return a steady nonzero value along the entire length of the line.

A solution to this problem is the second derivative. The second derivative still spikes at the discontinuity, but neither uniform values nor a constant gradient affect it. This is a more robust edge detector.

A two-dimensional edge is known as a corner. The corner is a powerful feature because it relates information about orientation. A corner rotation of  $90^\circ$  is shown in Figure 2.12.

*2.5.1 Hessian Operator.* The two-dimensional second derivative is known as the Hessian operator [11:317]. The Hessian operator is described as

$$\mathbf{H}(\mathbf{p}) = \begin{bmatrix} \frac{\partial^2 \mathbf{I}(\mathbf{p})}{\partial x^2} & \frac{\partial^2 \mathbf{I}(\mathbf{p})}{\partial x \partial y} \\ \frac{\partial^2 \mathbf{I}(\mathbf{p})}{\partial y \partial x} & \frac{\partial^2 \mathbf{I}(\mathbf{p})}{\partial y^2} \end{bmatrix}. \quad (2.15)$$

At each point  $\mathbf{p}$  in the image  $\mathbf{I}$ , the  $2 \times 2$  Hessian matrix provides information about the principal curvatures at that point. The principal curvatures are the magnitudes of the increase in pixel intensity as the corner is crossed over. A good corner has two large

principal curvatures. This means that good corner points will have two large eigenvalues in the Hessian matrix. Points with small eigenvalues are not good corners. Implementation of the Hessian requires an image gradient operator, discussed next.

*2.5.2 Sobel Derivative Operator.* One popular image gradient operator is the Sobel derivative [12]. The Hessian operator can be calculated by using the Sobel derivative. This gradient is calculated by convolving a two-dimensional kernel with the image. The kernel can be of any square size, but for demonstration a  $3 \times 3$  kernel is shown in Figure 2.13. The point in the kernel in which the final convolution value is placed on the image is known as the anchor point.

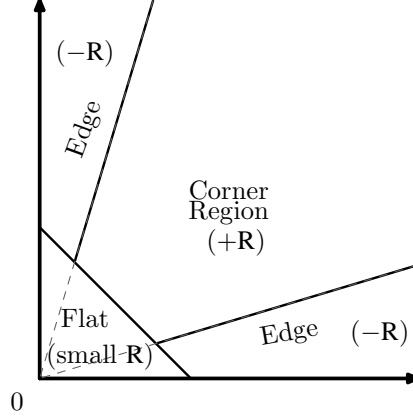
One implementation of the Sobel operator, at the image point corresponding the anchor  $e$ , approximates the gradients by

$$\frac{\partial}{\partial x} = (c + 2f + i) - (a + 2d + g) \quad (2.16)$$

$$\frac{\partial}{\partial y} = (g + 2h + i) - (a + 2b + c). \quad (2.17)$$

The gradients in the  $x$  and  $y$  directions are thus obtained. To obtain the second derivatives, convolve the Sobel filter again over the resulting gradient image. The Sobel derivative is a fast and simple gradient operator.

The Sobel concept is expanded into many different applications. Different kernel sizes and weightings provide different results. For example the Scharr kernel uses a unique weighting designed to allow more accurate gradient angle determination than the original Sobel kernel [11:150].



**Figure 2.14. Corner/Edge Classifier.** The classification and quality of a corner/edge can be determined by calculating its response  $R$ . Positive  $R$  corresponds to a corner, negative  $R$  to an edge, and small  $R$  to flat region. The  $R = 0$  lines are dashed.

*2.5.3 Harris Features.* Harris features [13] are found by using a localized two-dimensional autocorrelation function to determine the Hessian matrix. This is described as

$$\mathbf{M}(\mathbf{c}) = \begin{bmatrix} \sum_{-K \leq i, j \leq K} \mathbf{I}_x^2(x+i, y+j) & \sum_{-K \leq i, j \leq K} \mathbf{I}_x(x+i, y+j) \mathbf{I}_y(x+i, y+j) \\ \sum_{-K \leq i, j \leq K} \mathbf{I}_x(x+i, y+j) \mathbf{I}_y(x+i, y+j) & \sum_{-K \leq i, j \leq K} \mathbf{I}_y^2(x+i, y+j) \end{bmatrix} \quad (2.18)$$

where  $\mathbf{c}$  is the pixel coordinate, and  $K$  is the maximum range of the autocorrelation around the point. The Harris feature detector looks for points where the eigenvalues of the Harris operator are both large. This is the mark of a strong corner.

Harris further developed a corner/edge response function which relates corner and edge classification and quality. Given the autocorrelation matrix,

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \quad (2.19)$$

the trace and determinant of the autocorrelation are

$$\text{Tr}(\mathbf{M}) = m_{11} + m_{22} \quad (2.20)$$

$$\text{Det}(\mathbf{M}) = m_{11}m_{22} - m_{12}^2. \quad (2.21)$$

Note that  $m_{12}$  and  $m_{21}$  are equivalent. The corner/edge response function is then given as

$$\mathbf{R} = \text{Det} - \text{Tr}^2. \quad (2.22)$$

Classification and quality of the corner/edge is found by examining the graph in Figure 2.14. The parameter  $\mathbf{R}$  is positive in the corner region, negative in the edge regions, and small in the flat region.

*2.5.4 Shi-Tomasi Good Features.* Shi and Tomasi simplified the Harris corner [5]. The two eigenvalues of the  $\mathbf{M}$  matrix are taken, and as long as they are both larger than some threshold, the point is a good corner. Thus calculation of the Trace and Determinant is not needed. Further, Shi-Tomasi uses Sobel derivatives to calculate the  $\mathbf{M}$  matrix, instead of the localized autocorrelation. These differences reduce computation, while maintaining good feature detection.

*2.5.5 Scale Invariant Feature Transform (SIFT).* SIFT belongs to a separate class of feature detectors apart from the simple edge and corner detectors discussed above [14]. It was initially designed for object recognition in images. The two main components are the keypoint and the descriptor. The keypoint is a blob type feature, and the descriptor contains information about the feature. A blob is point of extremum in an image region. The descriptor is a vector containing information about directional gradients about the keypoint. Working together, they provide robust feature matching. SIFT involves four stages: scale-space extrema detection, keypoint localization, orientation assignment, and keypoint description.

**Scale-space extrema detection:** Potential keypoints are found in an image by detecting the local maxima and minima of an image transformation understood as a difference

of Gaussian (DoG) function over a scale-space upon the image. First, the unaltered image,  $\mathbf{I}(x, y)$ , is convolved with a set of Gaussian functions  $\mathbf{G}(x, y, \sigma)$  with variable scale,  $\sigma$ . The resulting collection of images are called  $\mathbf{L}(x, y, \sigma)$ .

$$\mathbf{G}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}. \quad (2.23)$$

$$\mathbf{L}(x, y, \sigma) = \mathbf{G}(x, y, \sigma) * \mathbf{I}(x, y). \quad (2.24)$$

Scale-space is determined by a constant multiplier  $k$ , such that the difference of Gaussian between two neighboring scales,  $\mathbf{D}(x, y, \sigma)$ , can be computed from

$$\mathbf{D}(x, y, \sigma) = \left[ \mathbf{G}(x, y, k\sigma) - \mathbf{G}(x, y, \sigma) \right] * \mathbf{I}(x, y). \quad (2.25)$$

Equation (2.24) is substituted into Equation (2.25),

$$\mathbf{D}(x, y, \sigma) = \mathbf{L}(x, y, k\sigma) - \mathbf{L}(x, y, \sigma). \quad (2.26)$$

These smoothed images are placed together into an octave. Each octave consists of a set number of images  $s$  such that the multiplier  $k$  is defined as

$$k = 2^{\frac{1}{s}} \quad (2.27)$$

so that each octave contains double the scale-space of the octave below it.

Local extrema are detected by analyzing the entire stack of the DoG images. A sample point is compared with the surrounding eight neighboring points and the 18 neighboring points in the scales directly above and below it. If the sample point is the maximum or minimum of its neighbors, it is considered a possible keypoint. See Figure 2.15. The number of local extrema detectable is dependent upon the frequency of sampling the image and the number of definite scales used in each octave.

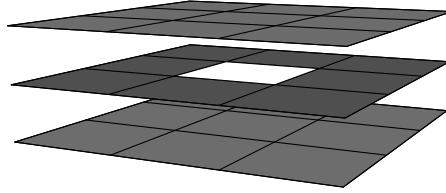


Figure 2.15. Possible Keypoint. This sample point is a local maximum among its immediate neighbors.

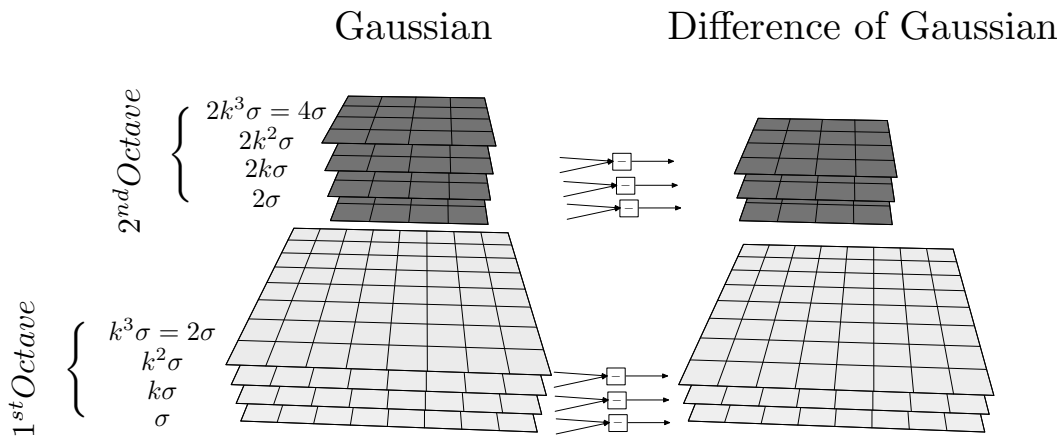


Figure 2.16. Octaves of the Difference of Gaussian Functions over a Scale-Space. Gaussian blurred images using the variable scale  $\sigma$  are stacked on the left. On the right are stacked the subtractions between the blurred images [14].

The octave of DoG's is the particular image sampling undergoing the DoG over the scale-space. As shown in Figure 2.16, each octave is an image downsampling of the octave below it.

Once all possible keypoints have been found, they are localized to exact positions in the space  $(x, y, \sigma)$ , and invalid keypoints are rejected.

**Keypoint localization:** Originally, the explicit value of the sample point  $(x, y, \sigma)$  was used to localize the point. However, a more accurate three-dimensional quadratic function is now used, providing better matching and stability in the algorithm. This quadratic is expressed as

$$\mathbf{D}(x, y, \sigma) = \mathbf{D}(\mathbf{x}) = \mathbf{D} + \frac{\partial \mathbf{D}^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 \mathbf{D}}{\partial \mathbf{x}^2} \mathbf{x}. \quad (2.28)$$

Here,  $\mathbf{x} = (x, y, \sigma)^T$ , and is the offset of the sample point from the origin. the extremum  $\hat{\mathbf{x}}$  are determined by setting Equation (2.28) to zero. This results in

$$\hat{\mathbf{x}} = - \left[ \frac{\partial^2 \mathbf{D}}{\partial \mathbf{x}^2} \right]^{-1} \frac{\partial \mathbf{D}}{\partial \mathbf{x}}. \quad (2.29)$$

By substituting Equation (2.29) into Equation (2.28), a useful function  $\mathbf{D}(\hat{\mathbf{x}})$  is created for rejecting low contrast unstable sample points,

$$\mathbf{D}(\hat{\mathbf{x}}) = \mathbf{D} + \frac{1}{2} \left[ \frac{\partial \mathbf{D}}{\partial \mathbf{x}} \right]^T \hat{\mathbf{x}}. \quad (2.30)$$

If image pixels are constrained to values between  $[0, 1]$ , a good threshold value is  $|\mathbf{D}(\hat{\mathbf{x}})| = 0.03$  [14].

After low contrast error checks have been performed, weak corners are considered. If the DoG image has strong principal curvature in one direction, but weak principal curvature in the perpendicular direction, the point is not a strong corner and the point is rejected. This corner strength determination is accomplished by dividing the square trace of the Hessian matrix by the determinant of the Hessian, expressed as

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} < \frac{(r+1)^2}{r} \quad (2.31)$$

where  $r$  is the maximum ratio allowed between the principal curvatures. If the ratio is less than some threshold value, the sample point is rejected.

**Orientation assignment:** Next a gradient magnitude  $m$  and orientation  $\theta$  are computed for an image  $\mathbf{L}(x, y)$  at the closest scale for a keypoint. A 36 bin orientation histogram is created for the keypoint, representing  $360^\circ$  of orientation. In the immediate pixel range around the keypoint, a Gaussian window is used upon the gradient magnitudes of each pixel, and this is added to that particular pixel's corresponding orientation bin. Then the three highest valued orientation bins are used to interpolate the peak orientation, and this value is the final determination of the keypoint orientation. If the orientation histogram has two peaks where the one is greater than 80% of the other, then two keypoints are created at that point, with two separate orientations.

**Keypoint descriptor:** Finally, each keypoint is given a feature vector descriptor which contains information concerning the gradient magnitudes and orientations of the pixels immediately surrounding the keypoint. The regions immediately around the keypoint are subdivided, and local orientation histograms are created in each subdivision. Trilinear interpolation is used when a pixel gradient slides across into another bin, thereby reducing boundary effects. This provides a robust method of describing a unique keypoint in the presence of affine and projective transforms of the keypoint.

For EIS, SIFT is performed upon the previous image and the current image in the video feed. The result is two sets of keypoint pixel locations, with possible correspondences among them. Feature matching is performed by finding the descriptors which have the smallest inverse cosine of the dot product between them. For example, the descriptor

from the previous image is taken and compared against every descriptor in the current image. The match that provides the lowest valued inverse cosine of the dot product is most likely the equivalent feature. A threshold distance ratio is used to maintain that a descriptor from the previous image is distinctly matched to a descriptor in the current image. If one descriptor from the previous frame is too similar to two different descriptors in the current frame, the keypoint is not matched. This process results in accurate feature correspondences between the previous frame and the current frame.

*2.5.6 Calculation of Motion Parameter Estimates.* The transformation matrix can be determined once features have been matched between the previous frame and the current frame. The motion parameter estimates are then extracted from the transformation matrix. Given a set of  $n$  feature correspondences, a matrix  $\mathbf{C}^+$  is created with the set of homogeneous feature pixel coordinates in the current frame,

$$\mathbf{C}^+ = \begin{bmatrix} \tilde{\mathbf{c}}_1^+ \cdots \tilde{\mathbf{c}}_n^+ \end{bmatrix} \quad (2.32)$$

and a matrix of homogeneous feature pixel coordinates in the previous frame,

$$\mathbf{C}^- = \begin{bmatrix} \tilde{\mathbf{c}}_1^- \cdots \tilde{\mathbf{c}}_n^- \end{bmatrix}. \quad (2.33)$$

The general homography transformation is then described as

$$\mathbf{C}^+ = \mathbf{H} \mathbf{C}^- \quad (2.34)$$

where  $\mathbf{H}$  is the homography matrix. By post multiplying both sides by  $\mathbf{C}^{-T}$ , and solving for  $\mathbf{H}$ , Equation (2.34) is rearranged into

$$\mathbf{H} = \mathbf{C}^+ \mathbf{C}^{-T} \left[ \mathbf{C}^- \mathbf{C}^{-T} \right]^{-1} \quad (2.35)$$

A solution for  $\mathbf{H}$  exists as long as  $\left[ \mathbf{C}^- \mathbf{C}^{-T} \right]$  is invertible, which is true if there are at least four vectors in  $\mathbf{C}^+$  and  $\mathbf{C}^-$ , where at least three of the point are not collinear [15:88].

An alternative method is known as the discrete linear transform (DLT) [15:88-91]. Let Equation (2.34) is be expressed as

$$\mathbf{C}_i^+ \times \mathbf{H} \mathbf{C}_i^- = 0 \quad (2.36)$$

where  $i$  is the  $i^{\text{th}}$  correspondence homogeneous point. If  $\mathbf{h}^1$ ,  $\mathbf{h}^2$ , and  $\mathbf{h}^3$  are the first, second, and third rows of the  $\mathbf{H}$  matrix, respectively, then this cross product becomes

$$\begin{bmatrix} [0 \ 0 \ 0] & -\mathbf{C}_i^{-\text{T}} & y_i^+ \mathbf{C}_i^{-\text{T}} \\ \mathbf{C}_i^{-\text{T}} & [0 \ 0 \ 0] & -x_i^+ \mathbf{C}_i^{-\text{T}} \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = 0. \quad (2.37)$$

where  $x_i^+$  and  $y_i^+$  are from the homogeneous point of the feature correspondence in the current image,  $\tilde{\mathbf{c}}_i^+ = \begin{bmatrix} x_i^+ & y_i^+ & 1 \end{bmatrix}^{\text{T}}$ . Now let

$$\mathbf{A}_i = \begin{bmatrix} [0 \ 0 \ 0] & -\mathbf{C}_i^{-\text{T}} & y_i^+ \mathbf{C}_i^{-\text{T}} \\ \mathbf{C}_i^{-\text{T}} & [0 \ 0 \ 0] & -x_i^+ \mathbf{C}_i^{-\text{T}} \end{bmatrix} \quad (2.38)$$

and

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix}. \quad (2.39)$$

Then Equation (2.37) can be written as

$$\mathbf{A}_i \mathbf{h} = 0 \quad (2.40)$$

where  $\mathbf{A}_i$  is a  $2 \times 9$  matrix.

The matrix  $\mathbf{A}$  is constructed as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \quad (2.41)$$

where  $\mathbf{A}_1 \cdots \mathbf{A}_n$  are the matrices for each of the  $n$  correspondences. The final solution for  $\mathbf{H}$  is found by determining the vector  $\mathbf{h}$  which lies in the null-space of the  $\mathbf{A}$  matrix. The matrix  $\mathbf{A}$  is of rank 8, and thus has only a one-dimensional null-space [15:90]. The null-space can be determined using singular value decomposition (SVD) [16:427], which decomposes the  $\mathbf{A}$  matrix into two normal matrices,  $\mathbf{U}$  and  $\mathbf{V}^T$ , and a non-negative diagonal matrix  $\mathbf{D}$ .

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T. \quad (2.42)$$

If SVD is implemented so that  $\mathbf{D}$  is in descending order, then the last column of  $\mathbf{V}$  is the vector  $\mathbf{h}$ . The final  $\mathbf{H}$  is then reconstructed from this  $\mathbf{h}$  vector.

When the homography matrix is simplified into the similarity transform, the DLT can be simplified enough so that only two correspondences are necessary to calculate the transform. The similarity homography transform is

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \lambda \cos \alpha & -\lambda \sin \alpha & T_x \\ \lambda \sin \alpha & -\lambda \cos \alpha & T_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.43)$$

Thus the parameters  $h_{31}$ ,  $h_{32}$ ,  $h_{33}$ , are already determined. Further,

$$\begin{aligned} h_{11} &= h_{22} \\ h_{12} &= -h_{21}. \end{aligned} \quad (2.44)$$

This allows for a reduction in Equation (2.37). Instead of solving for nine unknowns, it is necessary to solve for only four. This is shown if Equation (2.37) is expanded into

$$\begin{aligned} [0 \ 0 \ 0] \mathbf{h}^1 + \mathbf{C}_i^{-T} \mathbf{h}^2 + y_i^+ \mathbf{C}_i^{-T} \mathbf{h}^3 &= 0 \\ \mathbf{C}_i^{-T} \mathbf{h}^1 + [0 \ 0 \ 0] \mathbf{h}^2 - x_i^+ \mathbf{C}_i^{-T} \mathbf{h}^3 &= 0 \end{aligned} \quad (2.45)$$

and then reduced to

$$\begin{aligned} -x_i^- \mathbf{h}^{21} - y_i^- \mathbf{h}^{22} - \mathbf{h}^{23} + y_i^+ x_i^- \mathbf{h}^{31} + y_i^+ y_i^- \mathbf{h}^{32} + y_i^+ \mathbf{h}^{33} &= 0 \\ x_i^- \mathbf{h}^{11} + y_i^- \mathbf{h}^{12} + \mathbf{h}^{13} - x_i^+ x_i^- \mathbf{h}^{31} - x_i^+ y_i^- \mathbf{h}^{32} - x_i^+ \mathbf{h}^{33} &= 0. \end{aligned} \quad (2.46)$$

Because of the simplifications offered by Equation (2.44), this reduces again to

$$\begin{aligned} -y_i^- \mathbf{h}^{11} + x_i^- \mathbf{h}^{12} - \mathbf{h}^{23} + y_i^+ &= 0 \\ x_i^- \mathbf{h}^{11} + y_i^- \mathbf{h}^{12} + \mathbf{h}^{13} - x_i^+ &= 0 \end{aligned} \quad (2.47)$$

which can be represented in matrix form as

$$\begin{bmatrix} -y_i^- & -x_i^- & 0 & 1 \\ x_i^- & y_i^- & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{h}_{11} \\ \mathbf{h}_{12} \\ \mathbf{h}_{13} \\ \mathbf{h}_{23} \end{bmatrix} = \begin{bmatrix} -y_i^+ \\ x_i^- \end{bmatrix}. \quad (2.48)$$

The values for  $\mathbf{h}_{11}, \mathbf{h}_{12}, \mathbf{h}_{13}, \mathbf{h}_{23}$  are then found by row reduction. The final  $\mathbf{H}$  matrix becomes

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_{11} & \mathbf{h}_{12} & \mathbf{h}_{13} \\ -\mathbf{h}_{12} & \mathbf{h}_{11} & \mathbf{h}_{23} \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.49)$$

Once the homography matrix has been determined, the motion parameters are extracted. Returning back to the original transformation matrix description,

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (2.50)$$

the motion parameters are extracted according to

$$\Delta x = h_{13} \quad (2.51)$$

$$\Delta y = h_{23} \quad (2.52)$$

$$\Delta \lambda = \sqrt{h_{11}^2 + h_{21}^2} \quad (2.53)$$

$$\Delta \alpha = \arcsin \frac{-h_{21}}{\Delta \lambda}. \quad (2.54)$$

The motion parameters have now been determined, given two image correspondences.

## 2.6 Motion Parameter Estimation Using Optical Flow

An alternative method for determining feature correspondences between two images is optical flow. Optical flow calculates pixel velocity between two frames. However, for the purpose of EIS, pixel velocity is equivalent to the actual frame displacement. There are two strategies to optical flow; sparse and dense. In sparse optical flow, a select group of prominent feature points are tracked between frames. In dense optical flow, every pixel in the image is tracked.

*2.6.1 Sparse Optical Flow.* A popular sparse technique is the Lucas-Kanade method [17]. It relies on three assumptions; brightness constancy, temporal persistence, and spatial coherence. The fundamental optical flow equation is

$$I_x a + I_y b + I_t = 0 \quad (2.55)$$

where  $I_x$  and  $I_y$  are the spatial derivatives of the feature point in the  $x$  and  $y$  directions, respectively,  $a$  and  $b$  are the feature point velocities (displacements) in the  $x$  and  $y$  directions, respectively, and  $I_t$  is the image time derivative. The optical flow equation is constructed using the brightness constancy assumption and the temporal persistence assumption. However, the optical flow equation is fundamentally inaccurate because it has two unknowns. Multiple points must be considered for an accurate solution, and this is acceptable only when spatial coherence is assumed. Finally, solving the overdetermined system, the values for the displacement estimates  $a$  and  $b$  are found using a method of least-squares. These ideas are now presented in more detail.

The initial two-dimensional image is understood as

$$\mathbf{I}(x, y). \quad (2.56)$$

Adding a time dependence to the image results in

$$\mathbf{I}(x, y, t). \quad (2.57)$$

Brightness constancy states that

$$\mathbf{I}^+(x + a, y + b, t + \Delta t) = \mathbf{I}^-(x, y, t). \quad (2.58)$$

Thus for any image movement between frames, a particular feature does not change intensity value as it moves around the image in time. This can be understood in terms of a feature point  $F$ . Define a feature point  $F$ , whose movement can be tracked over time within an image  $\mathbf{I}(x, y, t)$  according to

$$F(\mathbf{g}(t), \mathbf{h}(t), t) \quad (2.59)$$

where  $\mathbf{g}(t)$  and  $\mathbf{h}(t)$  are arbitrary functions describing the motion of the feature over time along the  $x$  and  $y$  directions, respectively. Thus for any time  $t$ , the exact  $x$  and  $y$  coordinates for a feature are known. Equation (2.59) can then be understood as

$$F(x, y, t), \quad \text{where } x = \mathbf{g}(t), \text{ and } y = \mathbf{h}(t). \quad (2.60)$$

The brightness constancy assumption may now be restated in terms of this feature point

$$\frac{\partial F(x, y, t)}{\partial t} = 0. \quad (2.61)$$

Again, this equation states that the feature exhibits no change in intensity as it moves around the image over time.

Temporal persistence now becomes important. Assuming that velocities in the  $x$  and  $y$  directions are differentially small, then  $a$  and  $b$  are understood as

$$\begin{aligned} a &= \frac{dx}{dt} \\ b &= \frac{dy}{dt}. \end{aligned} \quad (2.62)$$

Further, temporal persistence dictates that the time derivative of the image at the feature point is differentially small. Define a particular feature location  $(i, j)$ , outputs of  $g(t)$  and  $h(t)$  at a particular time  $t = \tau$ , as

$$\begin{aligned} i &= g(\tau) \\ j &= h(\tau). \end{aligned} \quad (2.63)$$

Then the image time derivative is some value  $I_t$ ,

$$I_t = \left. \frac{\partial F(x, y, t)}{\partial t} \right|_{\substack{x=i, \\ y=j}}. \quad (2.64)$$

The image time derivative  $I_t$  for a feature point at pixel location  $\mathbf{c}$  in the previous image  $\mathbf{I}^-$ , given the current image  $\mathbf{I}^+$ , is calculated as

$$I_t = \mathbf{I}^+(\mathbf{c}) - \mathbf{I}^-(\mathbf{c}). \quad (2.65)$$

Combining the brightness constancy assumption and the temporal persistence assumption allows for the final determination of the optical flow equation. Referring back to Equation (2.61), it is clearly a multivariate function. Applying the chain rule for multivariate functions, the result is

$$\frac{\partial F(x, y, t)}{\partial t} = \left[ \frac{\partial F(x, y, t)}{\partial x} \Big|_{\substack{y=j \\ t=\tau}} \frac{dx}{dt} \right] + \left[ \frac{\partial F(x, y, t)}{\partial y} \Big|_{\substack{x=i \\ t=\tau}} \frac{dy}{dt} \right] + \left[ \frac{\partial F(x, y, t)}{\partial t} \Big|_{\substack{x=i \\ y=j}} \frac{dt}{dt} \right] \quad (2.66)$$

which simplifies to

$$\frac{\partial F(x, y, t)}{\partial t} = \left[ \frac{\partial F(x, y, t)}{\partial x} \Big|_{\substack{y=j \\ t=\tau}} \frac{dx}{dt} \right] + \left[ \frac{\partial F(x, y, t)}{\partial y} \Big|_{\substack{x=i \\ t=\tau}} \frac{dy}{dt} \right] + \left[ \frac{\partial F(x, y, t)}{\partial t} \Big|_{\substack{x=i \\ y=j}} \right]. \quad (2.67)$$

Now let the following definitions be made,

$$\begin{aligned} I_x &= \frac{\partial F(x, y, t)}{\partial x} \Big|_{\substack{y=j \\ t=t}} \\ I_y &= \frac{\partial F(x, y, t)}{\partial y} \Big|_{\substack{x=i \\ t=t}} \end{aligned} \quad (2.68)$$

where the parameters  $I_x$  and  $I_y$  are the spatial instantaneous derivatives of the feature point in the  $x$  direction and  $y$  directions, respectively. These remain constant for all time, assuming the feature does not alter its appearance, as follows the brightness constancy assumption. Substituting the definitions for  $I_x$ ,  $I_y$ ,  $a$ ,  $b$ , and  $I_t$  back into Equation (2.61), the final optical flow equation is formed as

$$I_x a + I_y b + I_t = 0. \quad (2.69)$$

This can be rewritten into

$$\nabla I^T \mathbf{u} = -I_t \quad (2.70)$$

where

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix}. \quad (2.71)$$

This equation has two unknowns, the  $a$  and  $b$  values. Thus, in order to determine this value, optical flow equations for multiple points must be generated. The spatial coherence assumption allows for this.

*Spatial coherence* states that neighboring pixels belong to the same object in the image, and move in a similar fashion. Thus the pixels immediately surrounding the feature point are used as the additional points necessary to solve Equation (2.70). An  $n \times n$  window is created, depending on how many additional pixels are used. A value of  $n = 5$  is typical.

The overdetermined system is now

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{n^2}) & I_y(p_{n^2}) \end{bmatrix} \mathbf{u} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{n^2}) \end{bmatrix}. \quad (2.72)$$

This can be rewritten as

$$\mathbf{A}\mathbf{u} = -\mathbf{d} \quad (2.73)$$

where

$$\mathbf{A} = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{n^2}) & I_y(p_{n^2}) \end{bmatrix} \quad (2.74)$$

and

$$\mathbf{d} = \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{n^2}) \end{bmatrix}. \quad (2.75)$$

For an overdetermined system, the least-squares minimum error is determined by solving

$$\min ||\mathbf{A}\mathbf{u} - \mathbf{d}||^2. \quad (2.76)$$

This is accomplished by

$$\left[ \mathbf{A}^T \mathbf{A} \right] \mathbf{u} = \mathbf{A}^T \mathbf{d}. \quad (2.77)$$

Rearranging and solving for  $\mathbf{u}$ , the final displacement estimate is

$$\mathbf{u} = \left[ \mathbf{A}^T \mathbf{A} \right]^{-1} \mathbf{A}^T \mathbf{d}. \quad (2.78)$$

As long as  $\left[ \mathbf{A}^T \mathbf{A} \right]$  is invertible, a unique solution for  $\mathbf{u}$  exists. This is true when at least two corners with large principle curvature are used as features.

Once the displacement vector  $\mathbf{u}$  has been calculated for a feature correspondence, it is added to the feature location in the previous image, and the new feature coordinate in the current image is obtained. The set of feature correspondences are then used to calculate the transformation matrix and the motion parameter estimates are extracted, as presented in Section 2.5.6.

*2.6.2 Dense Optical Flow.* Sparse optical flow is concerned with tracking a few prominent features in the image. Dense optical flow attempts to calculate the velocities for every pixel in the image. The Horn-Schunck method is a popular dense optical flow method [18]. It uses the brightness constancy assumption and an additional smoothness constraint, and also uses Laplacians in the  $x$  and  $y$  directions. The optical flow equations for the Horn-Schunck method are

$$\begin{aligned} \frac{\partial}{\partial x} \frac{\partial a}{\partial x} - \frac{1}{\alpha} \mathbf{I}_x \left[ \mathbf{I}_x a + \mathbf{I}_y b + \mathbf{I}_t \right] &= 0 \\ \frac{\partial}{\partial y} \frac{\partial b}{\partial y} - \frac{1}{\alpha} \mathbf{I}_y \left[ \mathbf{I}_x a + \mathbf{I}_y b + \mathbf{I}_t \right] &= 0 \end{aligned} \quad (2.79)$$

where  $\alpha$  is the regularization coefficient, which controls the smoothness of the vectors. displacements  $\mathbf{a}$  and  $\mathbf{b}$  are determined by using Lagrange multipliers to minimize the errors in the optical flow equations for an overdetermined system. Once displacements have been calculated, they are applied to every pixel and the correspondences are made.

As with feature detection and sparse optical flow, at least two correspondences must be matched between two images, then the similarity homography matrix  $\mathbf{H}$  is calculated, and finally the motion parameter estimates extracted.

## 2.7 Motion Parameter Estimation Using Inertial Measurement

Motion estimation algorithms dealing with optical sensors have been presented. Next inertial devices are presented. Estimation using an IMU is determined by integrating the output of a gyro sensor and multiplying it by the appropriate constant. First a brief overview of IMU's is presented, followed by the steps to determine the motion parameter estimates.

*2.7.1 IMU Overview.* Inertial measurement units provide data concerning acceleration along the three orthogonal axes, and angular rates about the three axes [19]. A reasonably accurate micro electromechanical systems (MEMS) based IMU can be purchased for a few hundred dollars. More precise IMU's are purchased for strategic systems like ICBM's, but cost hundreds of thousands of dollars and are beyond the scope of this effort.

The IMU is comprised of accelerometers and gyro sensors. Accelerometers measure specific force, which is the acceleration with respect to the inertial space plus the gravity vector. The inertial space is the reference frame which is non-rotating with respect to the stars, and whose origin lies at the center of the earth [19:21]. Gyro sensors measure angular rates with respect to the body reference frame.

For image stabilization, when the camera is viewing long distance scenes, angular rates are far more significant than accelerations. Translation effects of the camera do not alter the scene significantly, whereas angular displacement produces strong effects upon the scene. By using direct integration, image displacement  $(x, y, \alpha)$  caused by yaw, pitch, and roll can be directly calculated.

Gyro sensors suffer from several sources of error. These include g-insensitive bias, g-sensitive bias, anisoelastic bias, scale-factor error, cross-coupling, and zero-mean random bias [19:76]. The full measurement model for a rate integrating gyroscope is

$$\tilde{\omega}_x = (1 + \mathbf{S}_x)\omega_x + \mathbf{M}_y\omega_y + \mathbf{M}_z\omega_z + \mathbf{B}_{f_x} + \mathbf{B}_{g_x}\mathbf{a}_x + \mathbf{B}_{g_z}\mathbf{a}_z + \mathbf{B}_{a_{xz}}\mathbf{a}_x\mathbf{a}_z + \mathbf{n}_x \quad (2.80)$$

where  $\tilde{\omega}_x$  is the angular rate output by the gyro sensor,  $\mathbf{S}_x$  is the scale factor,  $\mathbf{M}_y$  and  $\mathbf{M}_z$  are cross-coupling coefficients,  $\mathbf{B}_{f_x}$  is the g-insensitive bias coefficient,  $\mathbf{B}_{g_x}$  and  $\mathbf{B}_{g_z}$  are the g-sensitive bias coefficients,  $\mathbf{B}_{a_{xz}}$  is the anisoelastic bias coefficient, and  $\mathbf{n}_x$  is a zero-mean random bias. The variables  $\omega_x, \omega_y$ , and  $\omega_z$  are the true rotation rates, and  $\mathbf{a}_x, \mathbf{a}_z$  are the accelerations of the gyroscope.

In general, these negative effects are most disastrous in low frequency long duration use of the gyro sensor. For high frequency short duration use, the gyro sensor is quite accurate. Depending the specific use of the IMU, this comprehensive measurement model is simplified for reduced computation. The simplification of these combined errors as a first-order Gauss-Markov noise process was used in [4].

*2.7.2 First-Order Gauss-Markov Error Process.* A first-order Gauss-Markov process is a stationary Gaussian process [20:94]. The autocorrelation function of a first-order Gauss-Markov process is of the form

$$\mathbf{R}(\tau) = \rho e^{-\beta|\tau|} \quad (2.81)$$

and of differential form

$$\dot{\mathbf{n}}(t) = \frac{-1}{\tau}\mathbf{n}(t) + \mathbf{w}_n(t) \quad (2.82)$$

where  $\mathbf{n}(t)$  is the current value of the noise,  $\tau$  is the time constant for the noise process, and  $\mathbf{w}_n(t)$  is zero-mean additive white Gaussian noise of strength  $\sigma^2$ . The time constant  $\tau$  and  $\beta$  value are related by

$$\tau = \frac{1}{\beta} \quad (2.83)$$

The Gauss-Markov process is often used to describe errors because it appears to fit much natural error phenomena, and the process is completely described by the parameters of Equation (2.81) [20:95].

*2.7.3 Integration and Estimate Determination.* The angular rate output by the gyro sensor is integrated to determine angular displacement. It is shown in [21:32] that for a constant velocity  $v$  over time  $t$ , the total displacement  $x(t)$  is given as

$$x(t) = x(0) + tv \quad (2.84)$$

where  $x(0)$  is the initial displacement a time  $t = 0$ . This is understood in angular terms as

$$\Omega(t) = \Omega(0) + t \frac{d}{dt} \Omega(t) \quad (2.85)$$

where  $\Omega(t)$  is the angular displacement, and  $\frac{d}{dt} \Omega(t)$  is the angular rate output by the gyro sensor. If a constant time step  $\Delta_t$  is assumed, then the displacement between time steps is represented as

$$\Omega_k = \Omega_{k-1} + \Delta_t \frac{d}{dt} \Omega_{k-1} \quad (2.86)$$

where  $k$  is the index of the time step. Thus for a vector of angular rates, the vector of angular displacements is generated.

Between the times of image capture, these angular displacements are summed. The result is the angular displacements,  $\phi, \theta, \psi$ , between frames. Multiplying these by an associated coefficient results in the final motion parameter estimate expressed as

$$\begin{bmatrix} m \\ n \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{K}_\theta & 0 \\ 0 & 0 & \mathbf{K}_\psi \\ \mathbf{K}_\phi & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \quad (2.87)$$

where  $\mathbf{K}_\phi$ ,  $\mathbf{K}_\theta$ , and  $\mathbf{K}_\psi$  are the displacement coefficients. These coefficients describe how much pixel movement is caused by a particular amount angular displacement.

Because inertial measurement is not affected by video image error, it has a unique benefit against template matching, feature detection, and optical flow.

## 2.8 Other Methods for Motion Parameter Estimation

Template matching, feature detection, optical flow, and inertial measurement comprise the main classes of EIS. However there are some alternative techniques which do not fit exactly into one of the main classes of algorithms.

*2.8.1 Phase Correlation.* Phase correlation is used to estimate translation and rotation [22]. It is based on the Fourier transform. A time difference  $\tau$  of a function  $f$  is equivalent to multiplication of  $e^{-\tau\omega j}$ , expressed as

$$\mathcal{F}\{f(t - \tau)\} = e^{-\tau\omega j} \mathcal{F}\{f(t)\}. \quad (2.88)$$

This is extended to images. Given a translated image  $\mathbf{I}^+(x, y) = \mathbf{I}^-(x - r, y - s)$ , where  $r$  and  $s$  are arbitrary displacements, then

$$\mathcal{F}\{\mathbf{I}^+(x, y)\} = e^{-(r\omega + s\omega)j} \mathcal{F}\{\mathbf{I}^-(x, y)\}. \quad (2.89)$$

A function,  $D(x, y)$ , is created such that

$$\phi\{D(x, y)\} = \phi\{\mathbf{I}^+(x, y)\} - \phi\{\mathbf{I}^-(x, y)\} = -(r\omega + s\omega) \quad (2.90)$$

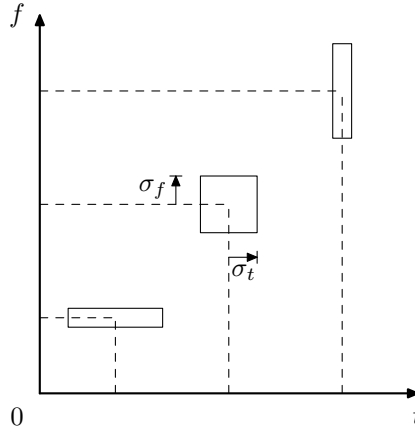
where  $\phi$  denotes phase. Thus

$$\mathcal{F}\{D(x, y)\} e^{-(r\omega + s\omega)j} \quad (2.91)$$

which leads to the useful Dirac-Delta function,

$$D(x, y) = \delta(x - r, y - s). \quad (2.92)$$

The final displacements  $r$  and  $s$  are determined by the peak value for  $D(x, y)$ .



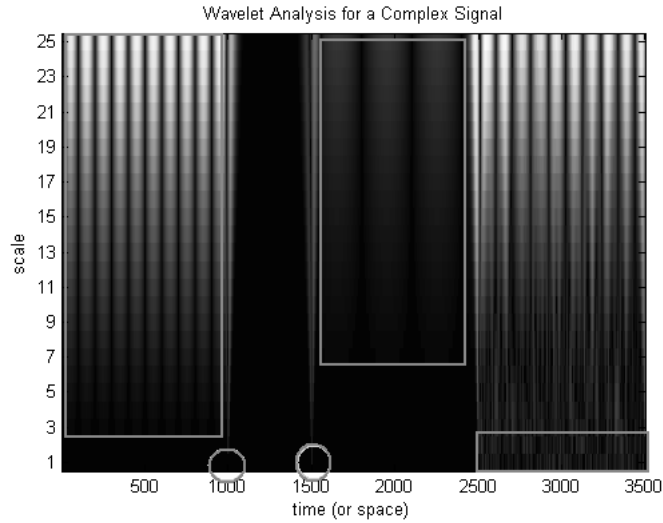
**Figure 2.17. Heisenberg’s Uncertainty Principle. As certainty in the frequency domain is increased, time domain certainty decreases, and vice versa [23:46].**

Phase correlation is simple and fast, but lacks accuracy in calculating translation and rotation; especially in the presence of blurring.

*2.8.2 Wavelets.* Wavelet analysis is a recent development in digital signals processing [23]. Traditionally, temporal and frequency characteristics of a signal are determined with separate analysis. For instance, Bode plots are used to determine spectrum information within a signal, but this finds the spectrum characteristics of the *entire* signal, and particular frequency/time relationships are not determined. With time domain analysis, it is difficult to understand the spectrum information for a particular length of the signal, due to the presence of noise and several different frequencies acting simultaneously. Wavelets analysis allows insight into *both* temporal and frequency characteristics.

A form of Heisenberg’s uncertainty principle is observed in this type of analysis. As more certainty in the frequency domain occurs, it results in less certainty in the time domain, as shown in Figure 2.17. The contrary holds as well.

The wavelet works by interrogating a signal with an asymmetric pulse at different scales. The scale determines the width of the pulse. High frequency interrogation is performed by a small scaled pulse, and low frequency interrogation is performed by a large scaled pulse. Signal response from corresponding scales are put together to form a graph

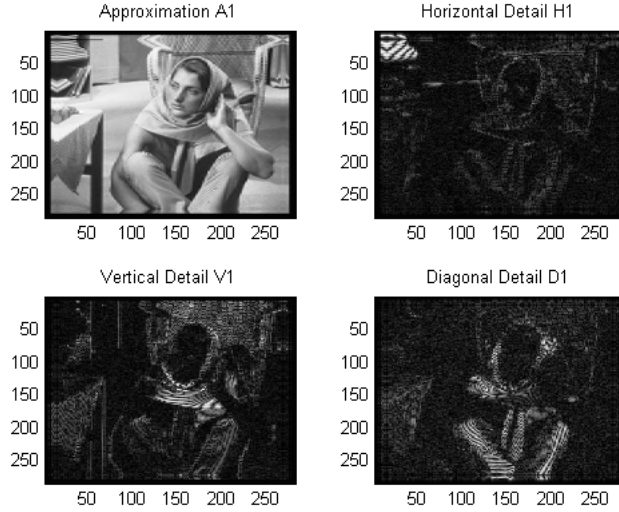


**Figure 2.18. One-Dimensional Wavelet Analysis.** Wavelet allows for time and frequency analysis simultaneously. Depicted is the wavelet analysis of a complex sinusoid.

with time as the  $x$ -axis and scale as the  $y$ -axis. A sample one-dimensional wavelet analysis on a complex signal is shown in Figure 2.18. Five regions are delineated. The upper left region shows the existence of a simple sinusoid of high frequency. The upper center region shows a simple sinusoid of low frequency. The two circled regions in the bottom left of the figure depict two signal discontinuities; the beginning and end of a zero-frequency signal. The final region in the bottom right shows the existence of noise added onto a high frequency sinusoid.

Wavelets are used upon two-dimensional data as well as one-dimensional signals. For images, they perform well as edge detectors. Wavelet decomposition is a technique used to separate an image into isolated edge response images. A horizontal, vertical and diagonal decomposition can be performed, which results in horizontal, vertical, and diagonal edge detection, as shown in Figure 2.19.

Wavelet decomposition can be used to simplify an image while still maintaining good features, and it is computationally fast.



**Figure 2.19. Sample Wavelet Decomposition.** Two-dimensional wavelet decomposition can be used as a horizontal, vertical and diagonal edge detector.

## 2.9 Outlier Compensation

The algorithms used for motion estimation have been presented. It was shown that both feature detection and optical flow result in sets of correspondences between the previous frame and the current frame. Motion estimation algorithms that result in a large collection of data values must obtain a single set of parameters. Using the mean average of the set is not suitable because when a small set of values is averaged, one outlier will render the estimate inaccurate. Some type of outlier compensation is necessary. For feature detection and optical flow parameter estimation, the combination of mismatched features along with well-matched features between images must be dealt with in order to determine an accurate transformation matrix. An outlier compensating algorithm provides a way to handle this problem.

*2.9.1 Median Filter.* A simple outlier compensation technique is the median filter. The median filter applies a window to a set of data, and replaces the data set value corresponding to the center value of the window with the median of the window. The filter size can be altered to change the performance.

*2.9.2 LMedS.* Least median of squares (LMedS) regression is a variant of the sum of square difference (SSD) [24]. Instead of minimizing the sum of square residuals  $e_i^2$ , the median of the values of  $e_i^2$  is minimized.

$$\min [\text{med}_i e_i^2] \quad (2.93)$$

the displacements resulting in the lowest median value are used as the motion parameter estimates.

*2.9.3 RANSAC.* Random sample and consensus (RANSAC) is capable of handling large outliers disrupting up to 50% of the data points [25]. The fundamental idea behind RANSAC is model fitting [26].

A vector of parameters,  $\theta$ , capable of satisfying some model, must be determined from a given data set. In the data set, several of the values are corrupted. These corrupted values are the outliers, and if they are included in the final model fitting by averaging, these outliers have a great deal of leverage to throw off the final parameter determination. Thus RANSAC is used to model fit in the presence of great number of outliers. It involves a repeated two step process: hypothesis and test.

**Hypothesis:** In the hypothesis stage, a minimal sample set (MSS) is chosen at random from the whole of the data. The size of the MSS is the lowest number required to generate a complete model parameter vector  $\theta$ .

**Test:** In the testing stage, this model is applied to all the data points. The set of data points which are consistent with the parameters is called the consensus set (CS).

This process of hypothesis and test is repeated until the probability of finding a better CS falls below a certain threshold.

Let  $q$  be the probability that for a random MSS, all data points are inliers (points which match the true model parameters). Then  $(1 - q)$  is the probability that the MSS will contain at least one outlier. Applying the hypothesis-test process several times will increase the probability of obtaining a complete inlier data set.

After  $h$  iterations, the probability of an outlier being chosen is equal to  $(1 - q)^h$ . Thus an alarm rate,  $\epsilon$ , is determined such that

$$(1 - q)^h \leq \epsilon. \quad (2.94)$$

Once the probability of an outlier drops below the alarm rate, no more iteration is necessary. Solving this equation for  $h$ ,

$$h \geq \frac{\log \epsilon}{\log(1 - q)}. \quad (2.95)$$

The value of  $q$  is needed. In [26], the value of the probability  $q$  is best estimated as

$$q \approx \left( \frac{N_I}{N} \right)^k \quad (2.96)$$

where  $N_I$  is the total number of inliers and is best approximated as  $\hat{N}_I$ , the number of inliers found so far.  $N$  is the total number of data points, and  $k$  is the number of data points selected for each hypothesis.

Once  $q$  and  $\epsilon$  are set, the algorithm repeats  $h$  times and results in a ranking of the best fitting model parameters. The original ranking criteria was the actual number of data points in the CS, but other methods have also been developed.

Before the MSS, comprised of the data points  $\mathbf{C} = [\mathbf{c}_1 \cdots \mathbf{c}_n]$ , with associated parameter vector,  $\theta_c$ , can be included in the CS, its consistency with the current model must be rated and pass some measure of similarity to the parameters that fit the model. Specifically, define a model  $\mathbf{M}(\theta)$ , with a parameter set  $\theta_0$  that perfectly fits the model. The associated minimum set of data points that instantiate  $\theta_0$  are denoted as  $\mathbf{C}_0$ . A distance measure  $D(\mathbf{C}, \mathbf{C}_0)$  is defined such that the difference between the current MSS  $\mathbf{C}$  and the zero-level MSS  $\mathbf{C}_0$  can be found. A threshold  $\delta$  is created such that as long as

$$D(\mathbf{C}, \mathbf{C}_0) \leq \delta \quad (2.97)$$

then the parameter vector  $\theta_c$  is considered consistent with the model and the associated data points  $\mathbf{C}$  are included in the CS.

Using the Euclidean norm as the distance measure,

$$D(\mathbf{C}, \mathbf{C}_0) = \sqrt{\sum_{i=1}^n (\mathbf{c}_i - \mathbf{c}_i^*)^2}, \quad (2.98)$$

where  $\mathbf{c}_i^*$  is an element of  $\mathbf{C}^*$ , the orthogonal projection of  $\mathbf{C}$  onto  $\mathbf{C}_0$ . A probability  $P_{\text{in}}$ , is then associated to this threshold,

$$P[D(\mathbf{C}, \mathbf{C}_0) \leq \delta] = P_{\text{in}}. \quad (2.99)$$

Simply stated,  $P_{\text{in}}$  is the probability that if the MSS  $\mathbf{C}$  has a distance below or equal to  $\delta$ , then the MSS is an inlier.

Assuming that the data points are corrupted by a zero-mean Gaussian noise  $\eta$  of variance  $\sigma^2$ , then

$$P[D(\mathbf{C}, \mathbf{C}_0) \leq \delta] = P\left[\sum_{i=1}^n \eta_i \leq \delta\right]. \quad (2.100)$$

Multiplying each side of the inequality by  $\frac{\eta_i}{\sigma^2}$ ,

$$P\left[\sum_{i=1}^n \eta_i \leq \delta\right] = P\left[\sum_{i=1}^n \left(\frac{\eta_i}{\sigma}\right)^2 \leq \frac{\delta^2}{\sigma^2}\right] = P_{\text{in}}. \quad (2.101)$$

The expression  $(\frac{\eta_i}{\sigma})^2$  represents a  $\chi^2$  distribution [26]. Thus rearranging Equation (2.101),

$$\delta = \sigma \sqrt{F_{\chi_n^2}^{-1}(P_{\text{in}})} \quad (2.102)$$

where  $F_{\chi_n^2}^{-1}$  is the inverse cumulative distribution function associated with a  $\chi_n^2$  random variable. It follows from [15:119], that for homographies,  $n = 2$  for the  $\chi^2$  distribution.

Once the qualification for the CS has been made, the hypothesis-test procedure is conducted  $h$  times or until a maximum iteration threshold is passed.

For EIS, RANSAC can be used to calculate the transformation matrix between several feature correspondences more accurately than a least-squares method. Using the similarity transform, only two data points are necessary for the MSS.

## 2.10 Kalman Filtering Overview

The Kalman filter is a method of state estimation which fuses information from multiple sensors [27]. Each sensor produces an estimate of a particular state, or combination of states, with a known variance. The filter combines the estimates according to variance in a way that optimally reduces Gaussian distributed errors. The Kalman filter requires development of a state dynamics model and a measurement model. The discrete time dynamics model is of the form

$$\mathbf{x}_{k+1} = \phi \mathbf{x}_k + \mathbf{B} \mathbf{u}_k + \mathbf{G} \mathbf{w}_k \quad (2.103)$$

where  $\mathbf{x}_k$  is the state vector,  $\mathbf{u}_k$  is the control input vector, and  $\mathbf{w}_k$  is a vector of zero-mean additive white Gaussian noise. The matrix  $\phi$  is the state transition matrix,  $\mathbf{B}$  is the control matrix, and  $\mathbf{G}$  is the noise matrix. The state transition matrix is calculated from the state dynamics matrix  $\mathbf{F}$ , as

$$\phi = e^{\mathbf{F}\Delta_t} \quad (2.104)$$

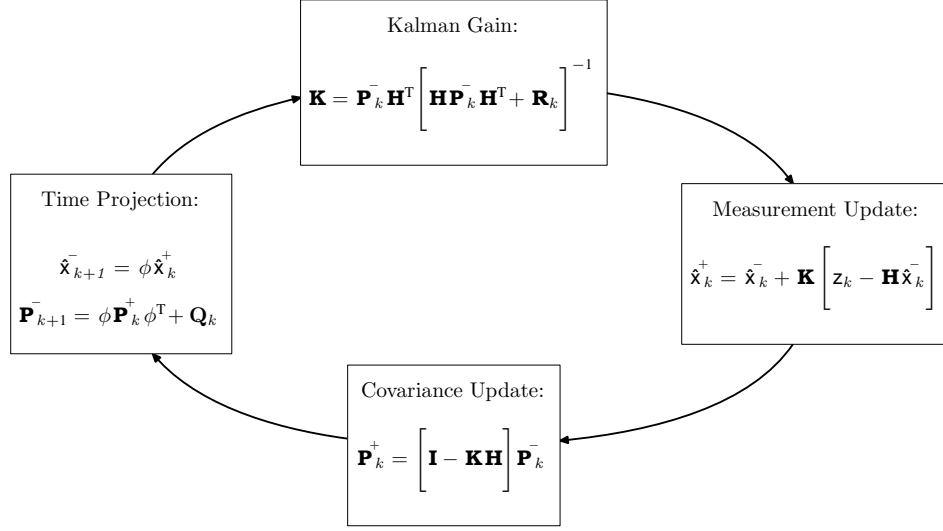
where  $\Delta_t$  is the time step for the system.

The measurement model is defined as

$$\mathbf{z}_k = \mathbf{H} \mathbf{x}_k + \mathbf{v}_k \quad (2.105)$$

where the discrete time measurement vector  $\mathbf{z}_k$  relates to the discrete state vector  $\mathbf{x}_k$  through the observation matrix  $\mathbf{H}$ , and is corrupted by zero-mean additive white Gaussian noise  $\mathbf{v}_k$ .

The general Kalman filter equations for this type of state space model are shown in Figure 2.20.



**Figure 2.20. Kalman Filter Equations [27:219].**

The dynamic model is modified into the dynamic perturbation model

$$\delta \dot{\mathbf{x}}_k = \mathbf{F} \delta \mathbf{x}_k + \mathbf{B} \mathbf{u}_k + \mathbf{G} \mathbf{w}_k \quad (2.106)$$

where  $\delta \mathbf{x}_k$  is the error in the state and is defined as

$$\delta \mathbf{x}_k = \tilde{\mathbf{x}}_k - \mathbf{x}_{k_{\text{true}}} \quad (2.107)$$

where  $\tilde{\mathbf{x}}_k$  is the nominal state value. For EIS, this value is assumed to be the output of an IMU. The associated perturbation measurement model is

$$\delta \mathbf{z}_k = \mathbf{H} \delta \mathbf{x}_k + \mathbf{v}_k \quad (2.108)$$

where  $\mathbf{H}$  is the observation matrix of the current measurement, and  $\delta \mathbf{z}_k$  is the perturbation measurement. The perturbation measurement  $\delta \mathbf{z}_k$  is defined as

$$\delta \mathbf{z}_k = \mathbf{z}_k - \mathbf{H} \tilde{\mathbf{x}}_k \quad (2.109)$$

where  $\mathbf{z}_k$  is the actual measurement collected. Identification of state uncertainty matrix  $\mathbf{Q}_k$  and measurement uncertainty matrix  $\mathbf{R}_k$  is required for implementation of the Kalman

filter equations as presented in Figure 2.20. Matrices  $\mathbf{Q}_k$  and  $\mathbf{R}_k$  are found from state variances and measurement variances, and must be determined from hardware testing. Calculation of these values is often the most significant and challenging task for a Kalman filter application.

Kalman filtering can be used for the fusion of optical and inertial sensor data [4].

### 2.11 *OpenCV*

OpenCV is an open source C/C++ library specially designed for computer vision applications [11]. The official release occurred in 2006, but the project began in 1999 from an Intel Research initiative with influence from the MIT Media Lab. The majority of the library work was developed under Intel’s Russian team. OpenCV has now experienced over 2 million downloads and it’s users group has expanded to 20,000 members. It’s libraries cover a multitude of computer vision topics, from machine learning and three-dimensional vision to image processing and motion tracking. It is an invaluable resource for the computer vision engineer, and was used in the development of this work.

### 2.12 *Current EIS Work*

*2.12.1 Current Template Matching Methods.* Hierarchical distributed template matching (HDTM) [28] is a novel approach to the template matching algorithm. Instead of using a full template to search across the image, only a select number of pixels from the template are used. This is called a distributed template. The pixels chosen are determined from an optimization algorithm that attempts to find the distributed template which has the most difference with it’s surroundings. Hierarchical image pyramids are used for a computationally fast template search. This approach was used on static photos in which a very a small template containing some object was run across the image. For example one test was a specific electronics chip on a silicon board filled with much electronics clutter.

The implementation found in [29] is an image stabilization algorithm for a surveillance camera. It uses phase correlation to obtain an initial image translation value. Next, a hierarchical image pyramid is generated and a search template. Using SSD, the template

is matched using a coarse-to-fine approach for model parameters. This algorithm uses color space instead of grayscale.

In [30], the camera is attached to a mine tunnel robot experiencing violent bumping motion. The proposed algorithm uses a grid of templates all performing sum of absolute difference template matching. Each grid is sized so that the maximum motion does not cause the template to jump outside its grid. Once the motion vectors from the different grids are determined, the true global motion of the frame is recovered from a least squares estimation method. The motion model used in this approach is the Similarity model.

*2.12.2 Current Feature Matching Methods.* The work in [31] uses Harris features and RANSAC for video mosaicing. The video mosaic was constrained to temporally local frames, instead of a large mosaic covering all time. The camera was affixed to a micro-UAV ( $\mu$ UAV), notorious for jittery feeds. The algorithm uses correspondence points from two frames via Harris features and inputs these into the RANSAC routine. The best estimate for the frame displacement is then determined. A Euclidean motion model is used.

The image stabilizer found in [32] uses feature points to determine frame displacement. A two dimensional rigid body motion model is used. A hierarchical image pyramid is used, where the levels are downsampled and further smoothed using a Laplacian of Gaussian technique. A template match is performed at each level to determine frame displacement.

Stationary wavelet transform decomposition is used in [33]. The image is decomposed to the second level, and the vertical and horizontal detail images are then projected into the vertical and horizontal axis, respectively. One-dimensional projection matches are then performed on the images to provide an initial estimate of image translation. A gradient-based approach is then used around a select number of feature points to refine the frame displacement. The feature points are selected by a unique algorithm that combines the vertical and horizontal detail information, yielding strong two-dimensional corners.

A novel affine model based approach is found in [1]. Feature tracking is performed upon six sub images. The sub images are constructed from binary pixels. An intensity histogram is first generated from the original image. The histogram is then divided into six

regions. Each region contains the information for one sub image. Each sub image is formed from thresholding the histogram values associated with it. The feature detector used is a binary image centroid extractor. Several centroids are extracted from each sub image. A data clustering approach is then used to calculate final parameter estimates for the frame displacement. Additionally, an alternative phase correlation algorithm is presented.

*2.12.3 Current Optical Flow Methods.* In [7], airborne video mosaicing is accomplished. A three step process is implemented; first phase correlation is used to provide initial frame movement estimates, second an optical flow algorithm refines these estimates, and third an SSD minimization technique is used to register the image to the mosaic. The inertial estimate is used to initialize the starting position of the SSD minimization upon the mosaic.

The approach in [6] uses mechanical and electronic image stabilization for a head mounted vision system. The Lucas-Kanade optical flow method is used to calculate displacement. A discrete Kalman filter is used to estimate camera pan and tilt from controlled inputs and gyro sensor data. However, it does not fuse EIS measurements into the Kalman filter. Pan and tilt servos, in conjunction with EIS, are used for motion compensation.

*2.12.4 Current Inertial Measurement Methods.* In IMU based methods, the gyro sensor is either stand-alone or integrated with an additional EIS algorithm. For instance, [34] presents a stand-alone gyro sensor used to predict camera motion aboard a moving ship. Angular displacement is multiplied by a constant to produce frame movement estimates. In [8], the camera is attached to a walking robot. The algorithm works by integrating gyro sensor information with template matching. Frame movement estimates, provided by inertial measurement, are used to initialize the template starting position for a local search.

## ***2.13 Conclusions on Current Image Stabilization Approaches***

The consistent philosophy for the non-IMU stabilizer is to develop an accurate single or cascaded motion estimation method. The consistent philosophy for IMU use in EIS is

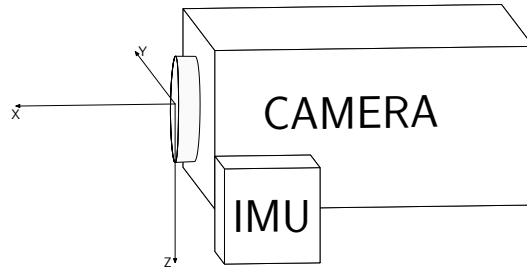
limited to initialization or constraining of the EIS algorithm. This application of the IMU is aimed at reducing the computational costs for the EIS algorithm. None of the current work seeks to address algorithmically the main causes of estimation error such as noise, moving objects, image, blur, and washout. In the following chapter, the development of the novel optical flow with inertial fusion algorithm is presented, with the aim of increasing the tolerance of standard EIS against noise, moving objects, image blurring, and image washout.

### III. Description of Stabilizer Algorithms

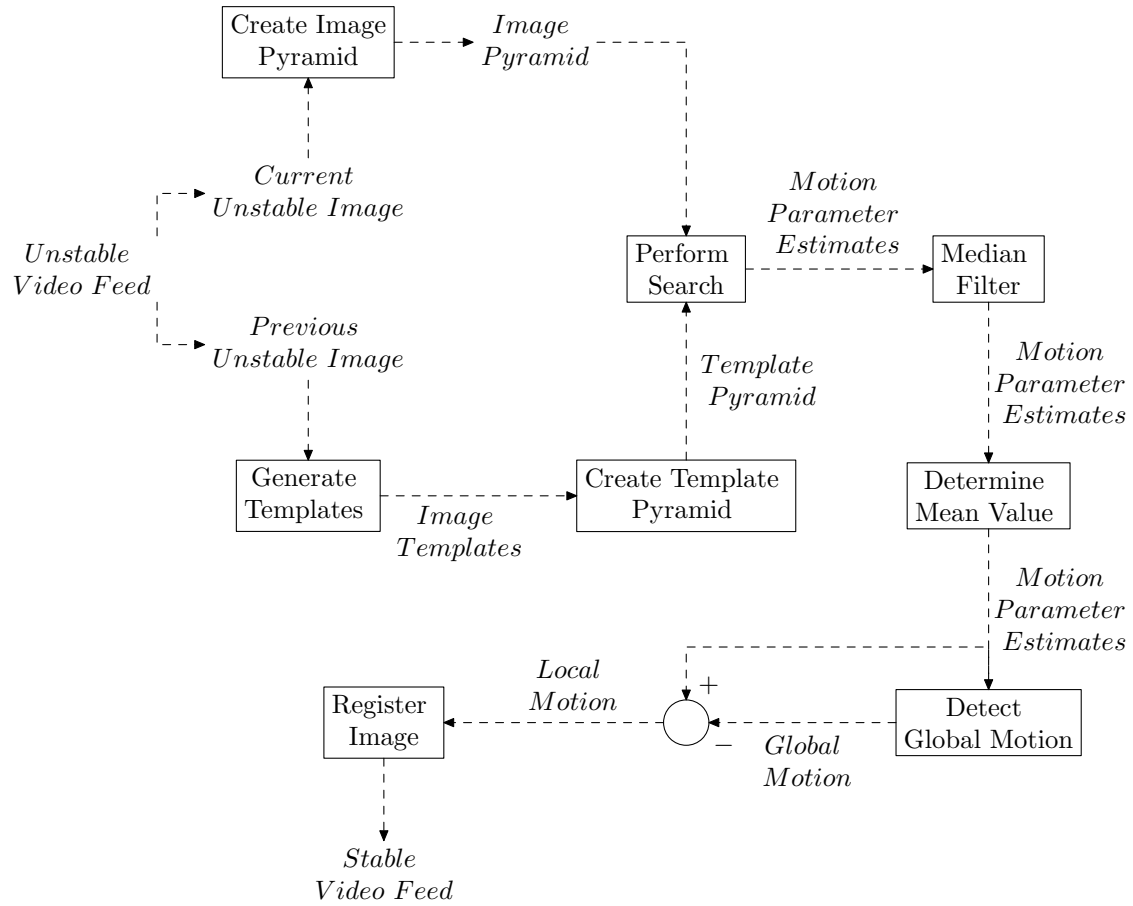
**E**lectronic image stabilization can be accomplished by myriad ways, as presented in chapter two. In this chapter five EIS algorithms are presented; one algorithm from each of the main classes of EIS and one algorithm which is a combination of two of them. The template matching algorithm is presented first, followed by the feature detection algorithm. The optical flow algorithm is then explained, followed by the inertial measurement algorithm. Finally, the novel contribution of this thesis, the optical flow with inertial fusion algorithm, is presented as the optimal solution to the image stabilization problem faced by camera systems affixed to highly dynamic mobile robotic platforms.

The coordinate frame for the system is shown in Figure 3.21. The  $\phi$ ,  $\theta$ , and  $\psi$  angular directions follow the right hand rule and coincide with the  $x$ ,  $y$ , and  $z$  directions, respectively. Observe that in the video feed, positive  $\phi$  displacement generates negative  $\alpha$  image rotation, positive  $\theta$  displacement generates negative  $m$  pixel movement, and positive  $\psi$  displacement generates positive  $n$  pixel movement.

Note that because the camera and IMU are both mounted on a rigid body, each experience the same rotation. Also, translation is of negligible significance to this particular application of EIS because distant scenes are assumed. Based on this truth and this assumption, no transformation matrix is necessary between the IMU and camera.



**Figure 3.21. Platform Coordinate Frame.**



**Figure 3.22. Template Matching Block Diagram.**

### 3.1 Template Matching

*3.1.1 Template Generation.* The block diagram for the template matching algorithm is shown in Figure 3.22. The development will start on the left hand side. The current unstable image and the previous unstable image are taken from the video feed. Six templates evenly spaced within the previous image, each of size  $64 \times 64$  pixels, are copied out of the image. The number of templates, the size of templates, and the number of pyramid levels to use were all determined by parameter sweeps to optimize the accuracy of the algorithm. The accuracy of the algorithm saturates at six templates of size  $64 \times 64$ , utilizing four-level image and template pyramids. Enough space from the edges of the image allow for detection of up to 128 pixels of movement in the  $m$  and  $n$  directions. This was chosen because the live test data collected had an approximate maximum displacement of 100 pixels.

Once the templates have been generated, they are decimated and collected into the template pyramid. Likewise, the current unstable image is transformed into its four-level pyramid. Once the image and template pyramids have been generated the search begins.

*3.1.2 Template Search.* Template searches are performed across the images of the pyramid. At the highest level, the search is constrained to 16 pixels in the  $m$  and  $n$  directions, corresponding to the maximum translation detection amount set by the spacing of the templates. 16 pixels works its way down to 128 pixel movement in the lowest level of the pyramid. Each subsequent level then performs a more localized search, until the final match is achieved. The search is for the  $m$  and  $n$  displacement which brings the correlation value  $C$  as close to unity as possible. The equation for the  $C$  score is

$$C(m, n) = \frac{\sum_{j=0}^{N-1} \sum_{i=0}^{M-1} \left\{ \left[ \mathbf{I}_I(m+i, n+j) - \overline{\mathbf{I}}_I \right] \left[ \mathbf{I}_T(i, j) - \overline{\mathbf{I}}_T \right] \right\}}{\sqrt{\sum_{j=0}^{N-1} \sum_{i=0}^{M-1} \left\{ \left[ \mathbf{I}_I(m+i, n+j) - \overline{\mathbf{I}}_I \right]^2 \right\} \cdot \sum_{j=0}^{N-1} \sum_{i=0}^{M-1} \left\{ \left[ \mathbf{I}_T(i, j) - \overline{\mathbf{I}}_T \right]^2 \right\}}} \quad (3.110)$$

where  $\mathbf{I}_I$  is the large image and  $\mathbf{I}_T$  is the template image. The parameter  $\overline{\mathbf{I}_I}$  is the mean intensity for the image, and  $\overline{\mathbf{I}_T}$  is the mean intensity for the template. The parameters  $M$  and  $N$  are the height and width, respectively, for the template. The variables  $m$  and  $n$  are the pixel location of the top left corner of the template on the image. The variables  $i$  and  $j$  are the pixel coordinates inside the template.

The  $(m, n)$  location yielding the highest  $C$  value is then subtracted from the known template corner position from the previous frame. The difference is the  $m$  and  $n$  pixel movement of the frame. This is performed with all six templates. Once the six different estimates have been determined, they are passed through a median filter of window size three. The last step is to average these filtered estimates, and the motion parameter estimates are found. These estimates are the values of the total motion of the camera, however. Undesired local motion must be separated from the desired global motion.

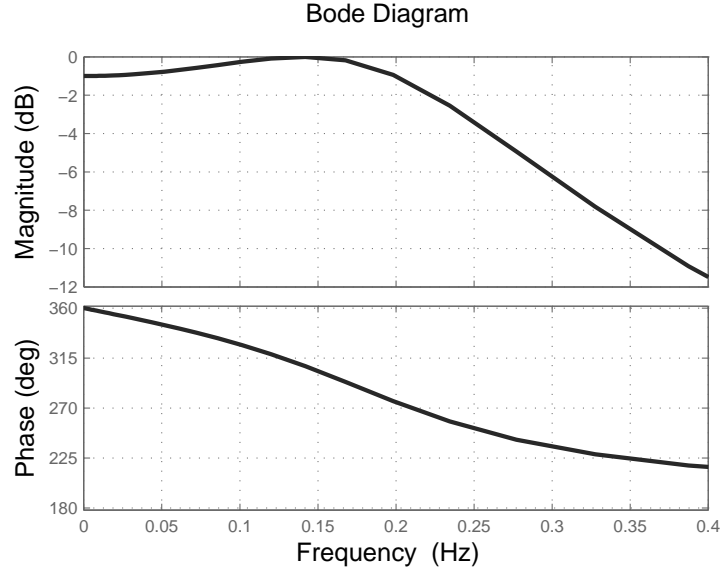
*3.1.3 Global Motion Detection.* Image stabilization on mobile platforms require a distinction between undesired camera jitter, known as local motion, and desired motion, known as global motion. For example, as the platform rounds a corner, the stabilized image should follow the turn, while removing unwanted high frequency bumps and jitters. This separation is accomplished by low pass filtering.

A low pass filter of the lowest order is desirable to keep filter latency as low as possible. By constraining the filter to second-order, the correction delay is limited to two frames. Stated another way, the correction for the current frame is dependent upon the current frame and the last two frames.

A second-order infinite impulse response (IIR) filter is expressed as a transfer function in the Laplace domain as

$$H(s) = \frac{b_0 s^2 + b_1 s + b_2}{a_0 s^2 + a_1 s + a_2} \quad (3.111)$$

where  $b_0, b_1, b_2, a_0, a_1$ , and  $a_2$  are the filter coefficients. The parameter  $a_0$  is usually set to unity. Out of the many different types of IIR and Finite Impulse Response (FIR) filters, the IIR elliptic filter is the most efficient in terms of amplitude response [35:485]. For a given filter specification, it results in the lowest filter order. Thus, the IIR elliptic filter



**Figure 3.23. Bode Response of the LPF.**

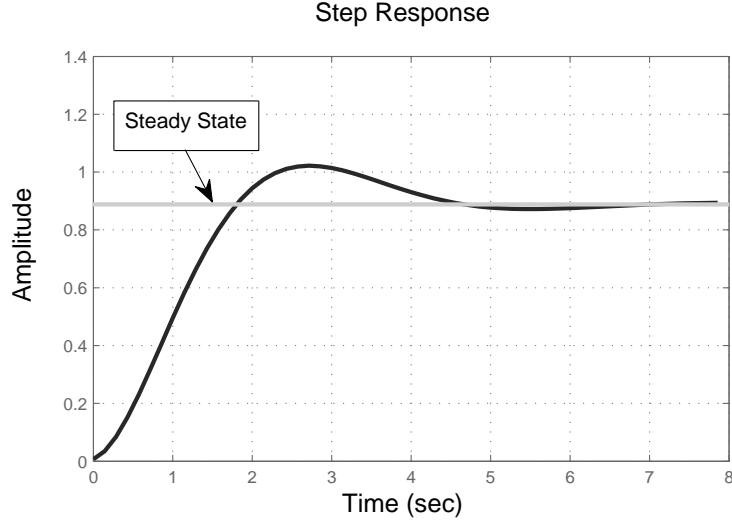
is used for this application. However, calculation of the coefficients of an elliptic filter is difficult. In practice, it is best to use a table lookup, or use filtering software [35:485].

Using the filtering toolbox in MATLAB, a second-order infinite impulse response elliptic filter was designed, with a sampling frequency of 7 Hz (The frame rate of the camera), and a cutoff frequency of 0.2 Hz. This cutoff frequency was determined by finding a frequency which allowed the video to remain approximately still on straightaways, and track turns around corners. The resulting coefficients are

$$\mathbf{b} = \begin{bmatrix} 7.3 \times 10^{-3} & 14.2 \times 10^{-2} & 7.3 \times 10^{-3} \end{bmatrix} \mathbf{a} = \begin{bmatrix} 1 & -1.789 & 0.822 \end{bmatrix}. \quad (3.112)$$

This filter is used for the  $m$ ,  $n$ , and  $\alpha$  directions. The Bode plot and step response of the filter are shown in Figure 3.23 and Figure 3.24. The system is stable, and the true cutoff is close to 0.2 Hz. The 5% settling time is approximately four seconds.

The step response can be understood as follows. When the LPF receives an image displacement, initially it considers the displacement completely as jitter and compensates completely for it. However, as long as the camera does not move from this new orientation, the image will center itself about the new orientation after four seconds.

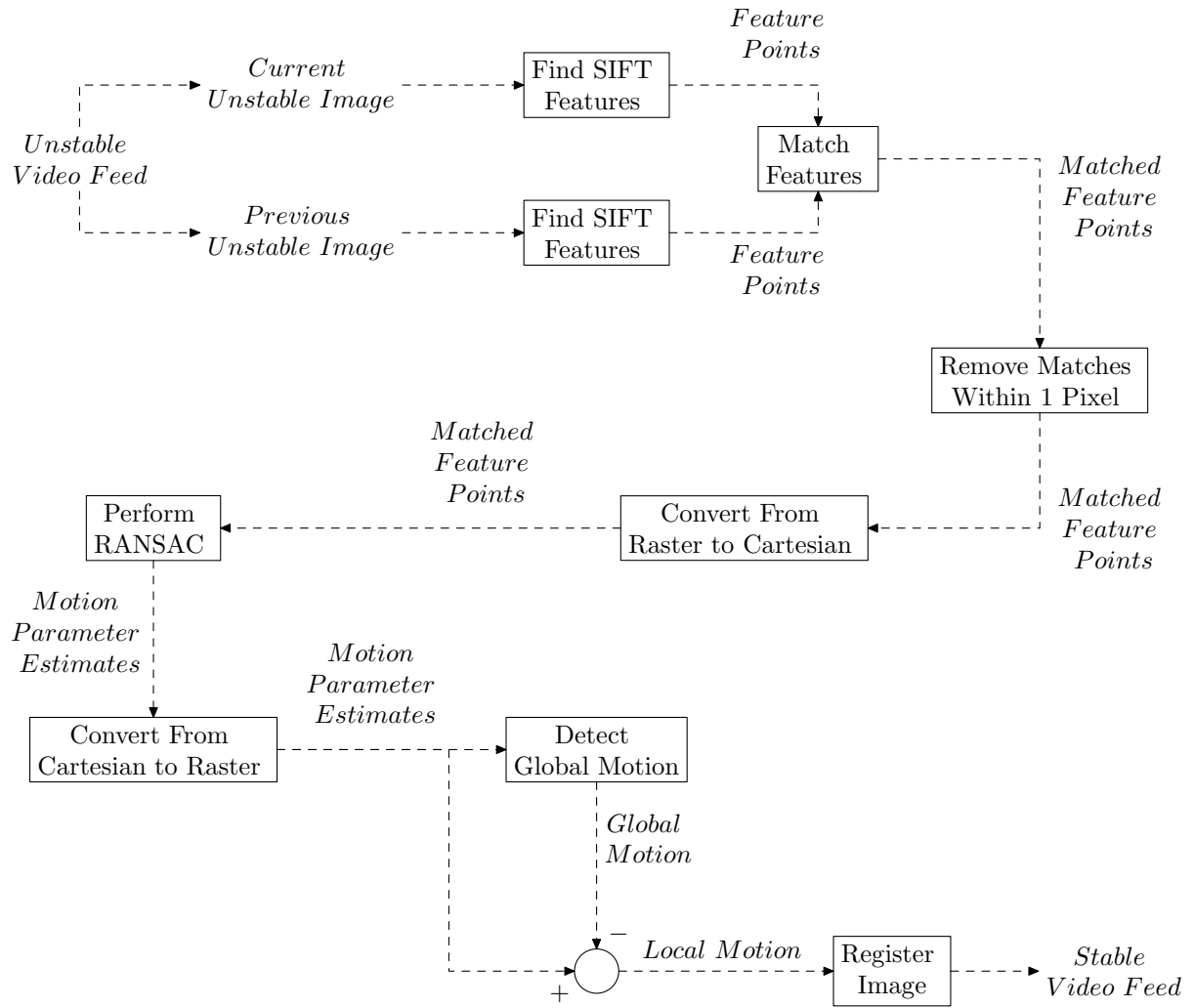


**Figure 3.24. Step Response of the LPF. The dotted line is the steady state value.**

The desired cutoff frequency will change depending on what the desired motion of the platform is. For example, a platform with slow side to side turning allows for a lower cutoff frequency. A platform with fast side to side turning will require a higher cutoff frequency to maintain video tracking during turns. A platform that is expected to track significant roll angle will require a higher cutoff frequency for its rotation filter. Each of the  $m$ ,  $n$ , and  $\alpha$  directions can modify its LPF filter cutoff frequency for desired performance.

Once the global motion is calculated for a frame using the low pass filter, it is subtracted out of the current set of optimal motion parameter estimates. This prevents compensation of the global motion, while performing compensation for the undesired local motion.

*3.1.4 Image Registration.* The last stage is to compensate for the undesired motion. This is accomplished by transforming the image by the negatives of the local motion parameter estimates. The result is a stable frame capable of tracking the low frequency motion of the platform.



**Figure 3.25. Feature Detection Block Diagram.**

*3.1.5 Template Matching Summary.* The template matching algorithm is straightforward and can be effective in certain uses of EIS where the video feed undergoes small frame displacement. However, for EIS on highly dynamic mobile robotic platforms, small frame displacement cannot be guaranteed. Also, the algorithm operates at an average speed compared to the other methods.

### **3.2 Feature Detection**

The next algorithm considered is the feature detection algorithm. The block diagram for the feature detection algorithm is shown in Figure 3.25. Again, the development will proceed from the top left of the block diagram.

*3.2.1 Feature Matching.* The SIFT algorithm is performed on both the current image and the previous image. Each feature has a particular pixel location and a directional gradient descriptor.

Feature matching is performed by finding the descriptors which have the smallest inverse cosine of the dot product between them. For example, the descriptor from the previous image is taken and compared against every descriptor in the current image. The match that provides the lowest value of the inverse cosine of the dot product is most likely the equivalent feature. A threshold distance ratio of 0.5 is used, which maintains that a descriptor from the previous image is distinctly matched to the current image. The next closest match distance must be at least 0.5 greater than the first match distance in order to be a valid match. This value of 0.5 was found by using a parametric sweep to determine the best tradeoff between the total number of matches and the number of incorrect matches.

The SIFT algorithm often produces multiple matches for a single pixel. This happens if the descriptor for a particular pixel has very distinct attributes that can be separated into two features. These are not useful for calculating the transformation matrix, however. We reduce the vector of feature matches to those which are at least one pixel apart.

*3.2.2 RANSAC Determination of Transformation Matrix.* The vectors of the feature correspondences are put through the RANSAC algorithm as created by [26], and the

output of RANSAC is the most probable transformation matrix. The algorithm uses the RANSAC parameters shown in Equation (3.113). The  $\epsilon$  and  $P_{in}$  values are recommended in [26]. The  $\sigma$  value was determined by a parameter sweep for the most accurate performance during the testing in chapter four. The number of iterations to use was determined by first evaluating the performance of RANSAC with a very high number of iterations, and then lowering the value just until the point performance was noticeably degraded.

$$\begin{aligned}
\sigma &= 1 \times 10^{-3} \\
\epsilon &= 1 \times 10^{-3} \\
P_{in} &= 0.99 \\
\text{minimum iteration} &= 0 \\
\text{maximum iteration} &= 1000
\end{aligned} \tag{3.113}$$

Once RANSAC outputs the most probable transformation matrix,  $\mathbf{H}$ , the  $x, y$ , and  $\alpha$  motion estimates are determined from the elements of the matrix.

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}. \tag{3.114}$$

$$\Delta x = h_{13}. \tag{3.115}$$

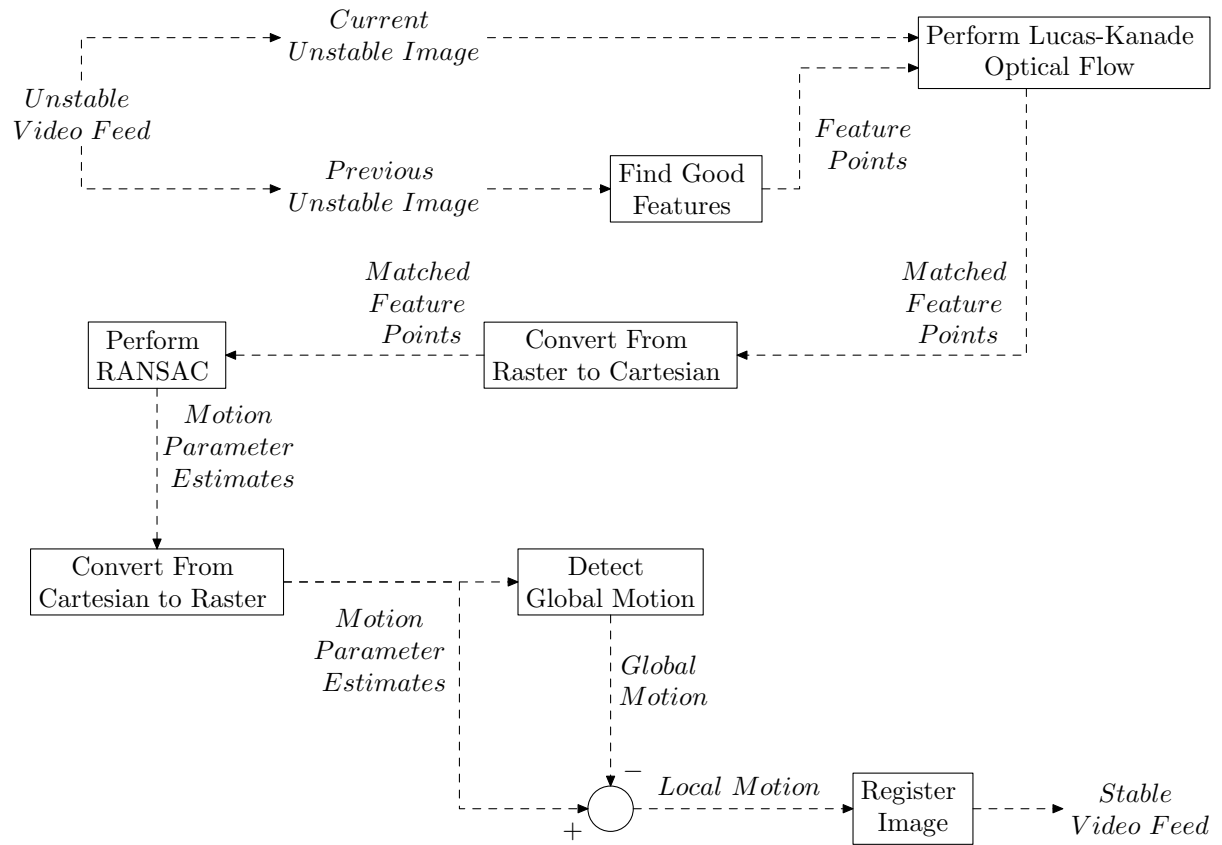
$$\Delta y = h_{23}. \tag{3.116}$$

$$\Delta \lambda = \sqrt{h_{11}^2 + h_{21}^2}. \tag{3.117}$$

$$\Delta \alpha = \arcsin \frac{-h_{21}}{\Delta \lambda}. \tag{3.118}$$

Scale  $\Delta \lambda$  is negligible since this image stabilizer assumes long distance scenes, thus consideration of the  $\Delta \lambda$  value is only necessary for except the calculation of  $\Delta \alpha$ .

These estimates are then converted back to raster coordinates, and the motion parameter estimates are achieved. Next, low pass filtering is accomplished to determine local motion, and finally the local motion of the video feed is removed by warping the image.



**Figure 3.26. Optical Flow Block Diagram.**

*3.2.3 Feature Detection Summary.* The feature detection algorithm is an accurate method of motion estimation, but is not reliable in the presence of image blurring and moving objects, as will be seen in the next chapter. Additionally, it incurs a high computational cost. However, The feature detection algorithm is still an effective stabilizer for many EIS applications.

### **3.3 Optical Flow**

The block diagram for the optical flow algorithm is shown in Figure 3.26. Development will begin from the unstable video feed in the upper left corner of the block diagram.

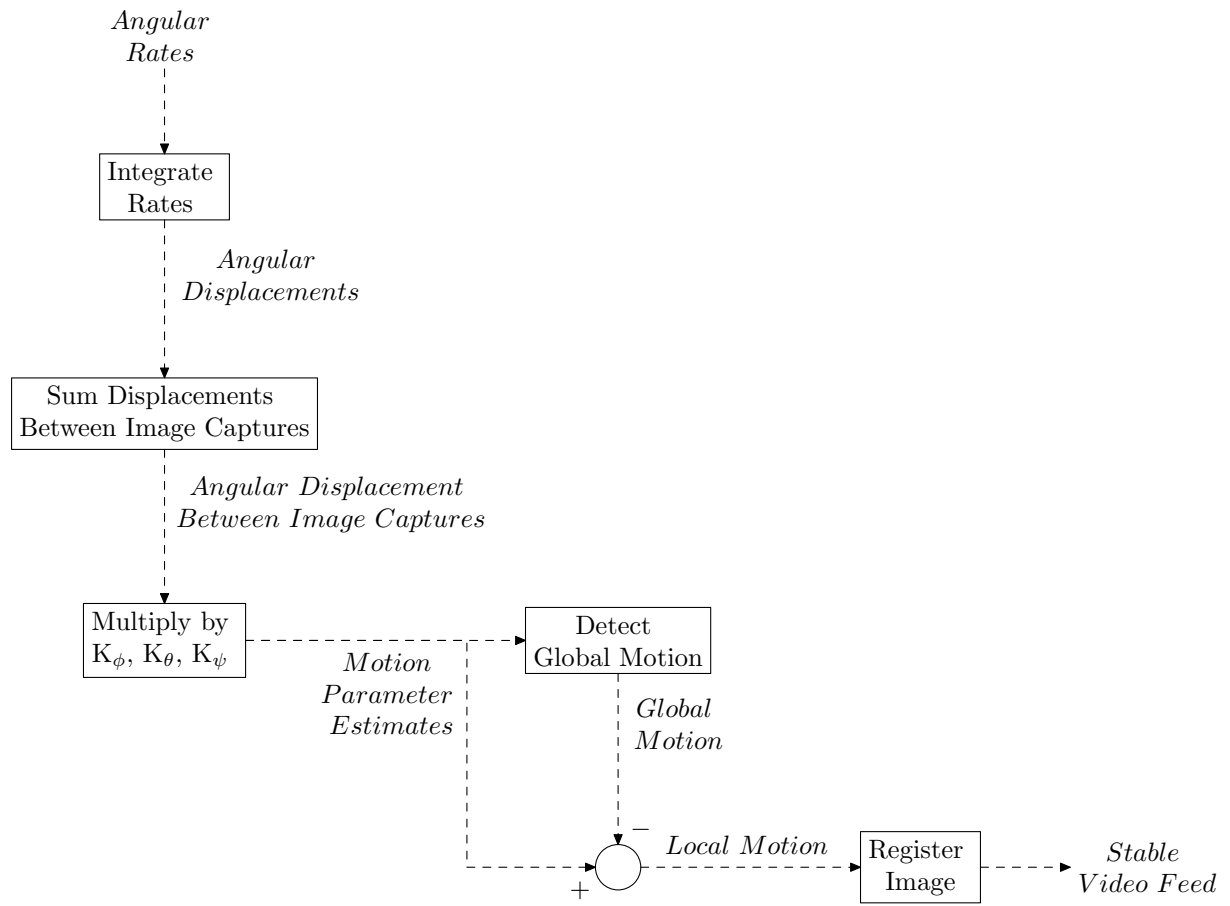
Motion estimation at the current time requires both the current image and the previous image. The good features are found using the Shi-Tomasi technique. Pyramidal Lucas-Kanade optical flow is then performed. The algorithm uses the find good features function and the pyramidal Lucas-Kanade optical flow function provided by the OpenCV library.

The vectors of matched points in the image, output by the Lucas-Kanade algorithm in raster coordinates, must be converted into Cartesian in order for RANSAC determination of the transformation matrix. The motion parameters are then extracted from the transformation matrix, and converted back into raster coordinates. Finally, the images are registered by applying the negative values of the calculated local motion.

*3.3.1 Optical Flow Summary.* Optical flow is an accurate EIS algorithm that is capable of robust estimation even in the presence of image blur. It is also computationally fast. Because of these reasons, optical flow was chosen for integration with inertial data to provide the optimal EIS motion parameter estimates. The optical flow with inertial fusion algorithm is discussed in Section 3.5.

### **3.4 Inertial Measurement**

The next algorithm considered is inertial measurement. The block diagram is shown in Figure 3.27. Inertial measurement is unique because it does not rely upon an optical sensor. All that is necessary is an IMU.



**Figure 3.27. Inertial Measurement Block Diagram.**

Inertial measurement works as follows. First the inertial data stream is received, and the angular rates,  $\mathbf{p}$ ,  $\mathbf{q}$ , and  $\mathbf{r}$  are taken. These angular rates are then integrated according to

$$\mathbf{\Omega}_k = \mathbf{\Omega}_{k-1} + \Delta_t \frac{d}{dt} \mathbf{\Omega}_{k-1}. \quad (3.119)$$

where  $\mathbf{\Omega}_k$  is the angular displacement at time step  $k$ ,  $\mathbf{\Omega}_{k-1}$  is the angular displacement at the previous time step  $k - 1$ ,  $\Delta_t$  is the time step magnitude, and  $\frac{d}{dt} \mathbf{\Omega}_{k-1}$  is the angular rate at the previous time step  $k - 1$ . This results in vectors of angular displacements,  $\phi$ ,  $\theta$ , and  $\psi$ . These displacements are then summed between image capture times, and multiplied by their respective coefficients,  $\mathbf{K}_\phi$ ,  $\mathbf{K}_\theta$ , and  $\mathbf{K}_\psi$ . The results are the motion parameter estimates frame to frame.

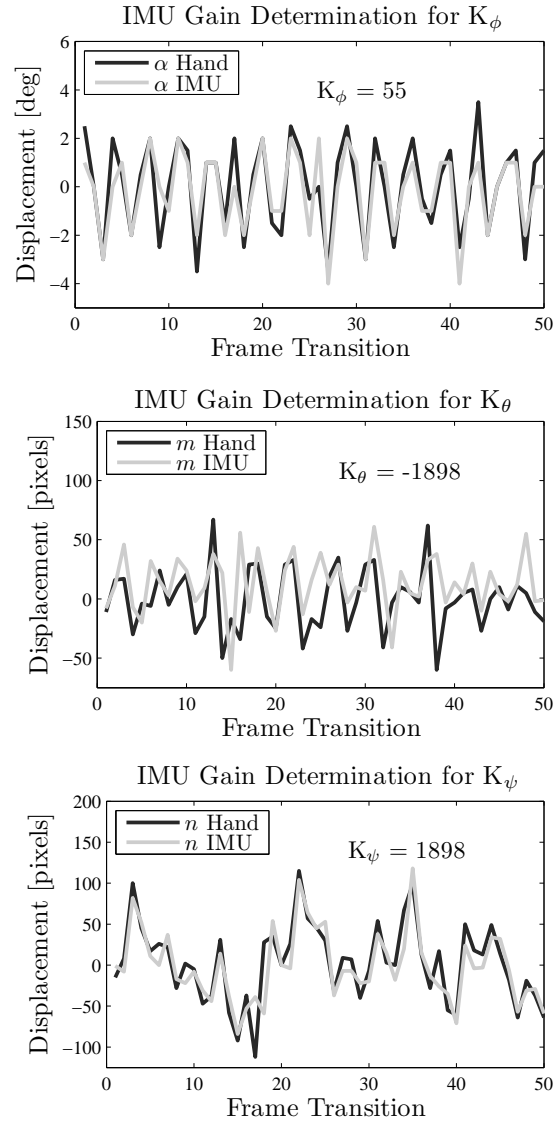
The values for  $\mathbf{K}_\phi$ ,  $\mathbf{K}_\theta$ , and  $\mathbf{K}_\psi$  were found by performing parametric sweeps and visually matching the output plots to a section of video of known displacement. The final fittings are shown in Figure 3.28. The hand determination of the video displacement is discussed in Section 4.3.1.

Once the motion parameter estimates are found, global motion is detected and then subtracted from the motion parameter estimates. This leaves the local motion for the final compensation. Once the local motion is compensated, a stabilized video feed is the result.

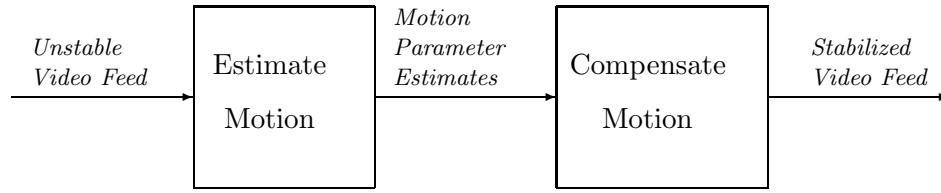
*3.4.1 Inertial Measurement Summary.* Inertial measurement is the only algorithm that is invariant to errors caused by image blur and moving objects. However, its accuracy depends on the specific IMU used. The errors in the MIDG IMU used in this thesis degraded the accuracy of inertial measurement. With the use of better IMU's, more accurate results can be achieved. Because of the the algorithm's invariance to image blur and moving objects, it is chosen for integration with optical flow.

### **3.5 Optical Flow with Inertial Fusion**

The final algorithm, the novel optical flow with inertial fusion, is the optimal solution to the image stabilization problem. Shown again in Figure 3.29, EIS contains a motion estimation block and a motion compensation block. The motion estimation block is further



**Figure 3.28. IMU Gains Determination.**

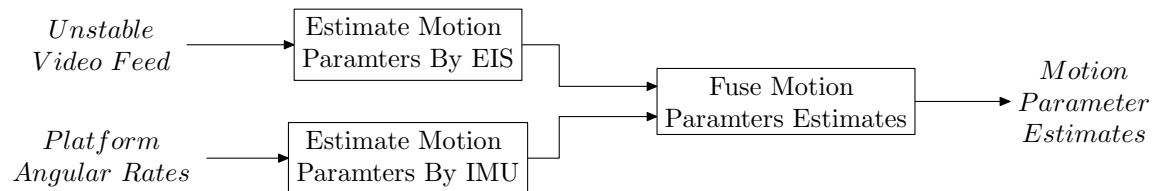


**Figure 3.29. The EIS Concept. Performing EIS involves the conversion of an unstable video feed into a stabilized video feed.**

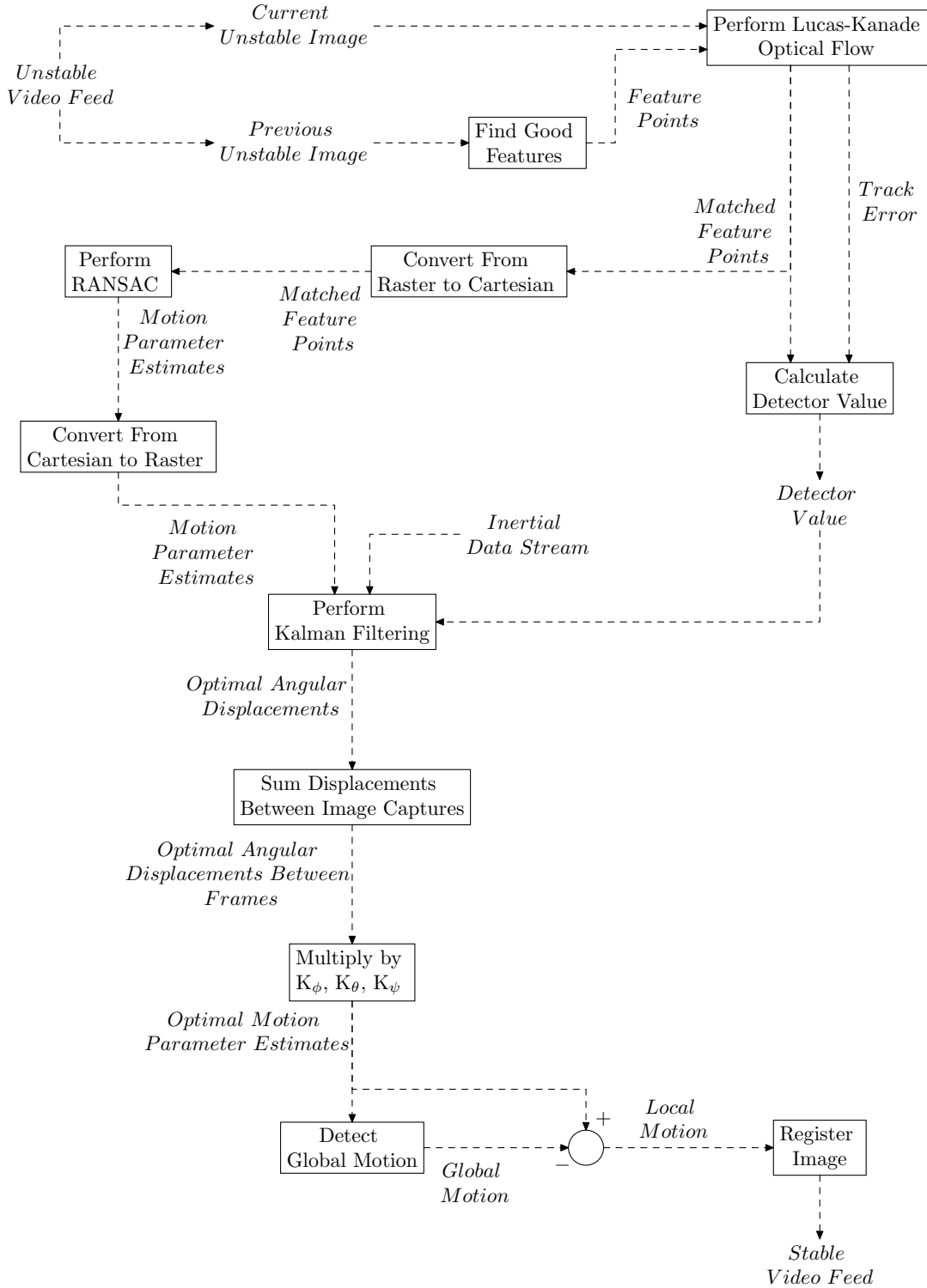
defined in Figure 3.30. The EIS algorithm performs directly upon the input video feed, while the IMU operates concurrently and is affected by the motion of the platform. For each frame time, a fusion step is performed to combine information from both estimation methods. In this way, the unique and useful traits of optical flow and inertial measurement can be utilized to provide optimal motion parameter estimates from frame to frame.

The block diagram for the optical flow with inertial fusion algorithm is found in Figure 3.31. Development begins from the top left corner of the diagram.

Motion estimation at the current time requires both the current image frame and the last captured image frame. The good features are found using the Shi-Tomasi technique. Pyramidal Lucas-Kanade optical flow is then performed. The algorithm uses the find good features function and the pyramidal Lucas-Kanade optical flow function provided by the OpenCV library.



**Figure 3.30. Motion Estimation Block.**



**Figure 3.31. Optical Flow with Inertial Fusion Block Diagram.**

The track error vector is used with the number of feature correspondences to generate a detector value. The detector yields values close to unity when there is significant disturbance and differences between the current frame and the previous frame, and close to zero when there is great similarity between frames. The equation used to generate the detector value is

$$D(T, C) = \left(\frac{T}{1000}\right)\left(1 - \frac{C}{500}\right) \quad (3.120)$$

where  $T$  is the average track error for the frame, and  $C$  is the number of correspondences. The track error ratio,  $\frac{T}{1000}$ , and the correspondence ratio,  $(1 - \frac{C}{500})$ , are multiplied to calculate the detector value. The number 1000 is used in the track error ratio because the maximum track error value during testing was approximately 1000. The track error ratio is close to unity when there is significant track error, and close to zero when there is not. The value of 500 was selected for the correspondence ratio because the maximum number of correspondences allowed in the algorithm is 500. This choice of a 500 correspondence maximum was chosen because the approximate maximum number of correspondences between two frames during live testing was found to be 500. When there is a high number of correspondences, the correspondence ratio is close to zero, and when there is a low number of correspondences, the correspondence ratio is close to unity. Multiplying the two ratios, the final detector value gives insight into the reliability of the motion parameter estimates found by optical flow.

Note that even though the detector value is correlated to actual performance of the stabilizer, a high detector value does not necessarily mean features could not be accurately matched. It states only that there is significant pixel intensity difference between the image, whatever the reason for the difference could be.

The vectors of matched points in the image, output by the Lucas-Kanade algorithm in raster coordinates, are converted into Cartesian coordinates in order for RANSAC determination of the transformation matrix. The motion parameter values are then extracted from the transformation matrix. The  $x$ ,  $y$ , and  $\alpha$  displacements are then fused with gyro sensor data from the inertial device to generate the optimal motion estimate. This fusion is accomplished by way of Kalman filtering.

*3.5.1 Kalman Filter Development.* A Kalman filter is developed to generate the optimal estimates of the angular displacements of the camera given data from the IMU and the EIS motion parameter estimate. The particular filter used is a nine state discrete Kalman filter using a perturbation model to estimate the IMU output errors. The IMU output errors are sufficiently characterized and modeled as a first-order Gauss-Markov process.

The state vector for the system is shown in Equation (3.121). There are three angular displacement states,  $\phi$ ,  $\theta$ , and  $\psi$ , three angular rate states,  $\mathbf{p}$ ,  $\mathbf{q}$ , and  $\mathbf{r}$ , and three drift bias states  $\mathbf{b}_\phi$ ,  $\mathbf{b}_\theta$ , and  $\mathbf{b}_\psi$ . The translational states have negligible significance to this particular application of EIS and are ignored.

$$\mathbf{x} = \begin{bmatrix} \phi & \theta & \psi & \mathbf{p} & \mathbf{q} & \mathbf{r} & \mathbf{b}_\phi & \mathbf{b}_\theta & \mathbf{b}_\psi \end{bmatrix}^T \quad (3.121)$$

*3.5.1.1 Characterization of IMU Error.* The IMU errors affecting the angular outputs are approximated as a first-order Gauss-Markov process. This is described as a noise source whose autocorrelation function is of the form

$$\mathbf{R}(\tau) = \rho e^{-\beta|\tau|} \quad (3.122)$$

and of differential form

$$\dot{\mathbf{n}}(t) = \frac{-1}{\tau} \mathbf{n}(t) + \mathbf{w}_n(t) \quad (3.123)$$

where  $\mathbf{n}(t)$  is the current value of the noise,  $\tau$  is the time constant for the noise process, and  $\mathbf{w}_n(t)$  is zero-mean additive white Gaussian noise of strength  $\sigma^2$ . The time constant  $\tau$  and  $\beta$  value are related by

$$\tau = \frac{1}{\beta} \quad (3.124)$$

To determine the values of  $\rho$  and  $\beta$  in Equation (3.122) for the particular IMU used in the experiment, the device was left motionless on a table and three sets of data were collected, each of duration of at least 20 minutes. Each angular rate was affected by a random drifting bias. These drifts are shown in Figure 3.32, which were arrived at by cumulative summing of the angular rate vector provided by the IMU.

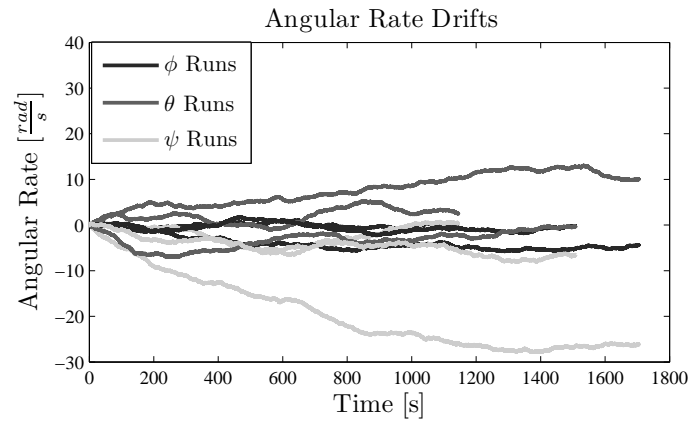


Figure 3.32. Angular Rate Drift.

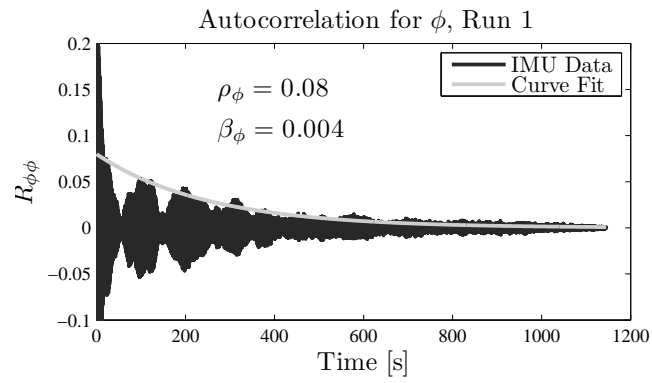
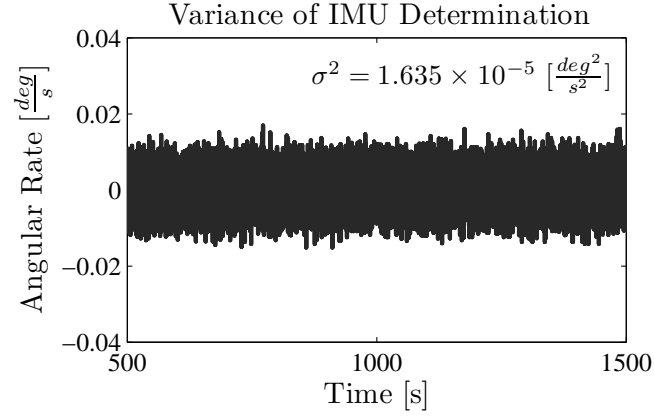


Figure 3.33. Example Autocorrelation and Parameter Determination.

**Table 3.1. Autocorrelation Parameter Values.**

<i>Run 1</i>	<i>Run 2</i>	<i>Run 3</i>	<b>Average</b>	
0.08	0.12	0.17	<b>0.123</b>	$\rho_\phi$
$4.0 \times 10^3$	$3.0 \times 10^3$	$3.0 \times 10^3$	<b><math>3.33 \times 10^{-3}</math></b>	$\beta_\phi$
0.025	0.025	0.035	<b>0.0283</b>	$\rho_\theta$
$2.5 \times 10^{-3}$	$1.5 \times 10^{-3}$	$1.5 \times 10^{-3}$	<b><math>1.83 \times 10^{-3}</math></b>	$\beta_\theta$
0.15	0.045	0.035	<b>0.04</b>	$\rho_\psi$
$2.5 \times 10^{-3}$	$1.3 \times 10^{-3}$	$1.0 \times 10^{-3}$	<b><math>1.115 \times 10^{-3}</math></b>	$\beta_\psi$



**Figure 3.34. Example Angular Rate Output.**

Autocorrelations of the unsummed angular rate vector are calculated, and  $\rho$  and  $\beta$  values determined by exponential curve fitting. An example plot with its curve fit is shown in Figure 3.33. The complete collection of Gauss-Markov parameter values is found in Table 3.1. IMU sensor noise is characterized by using the averages of these values.

*3.5.1.2 Dynamic Model Development.* The general form for the dynamic perturbation model of the platform is

$$\delta \dot{\mathbf{x}}_k = \mathbf{F}\delta \mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{G}\mathbf{w}_k \quad (3.125)$$

where  $\delta \mathbf{x}_k$  is the perturbed state vector,  $\mathbf{u}_k$  is the deterministic input vector, and  $\mathbf{w}_k$  is a vector of zero-mean additive white Gaussian noise.  $\mathbf{F}$  is the system dynamics matrix,  $\mathbf{B}$  is the control matrix, and  $\mathbf{G}$  is the noise matrix. There is no control mechanism for the system, so  $\mathbf{B} = 0$ . The  $\mathbf{G}$  matrix is the identity matrix. The  $\mathbf{F}$  matrix is

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\beta_\phi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta_\theta & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta_\psi \end{bmatrix}. \quad (3.126)$$

The state uncertainty matrix,  $\mathbf{Q}_k$ , is comprised of the variances of the different states. The variances of the three bias states,  $\mathbf{b}_\phi$ ,  $\mathbf{b}_\theta$ , and  $\mathbf{b}_\psi$ , are found from using Equation (3.127) for each run and averaging the results. The units for the variance of the bias states are degrees squared. The three angular rate state variances,  $\mathbf{p}$ ,  $\mathbf{q}$ , and  $\mathbf{r}$ , are found from the variance of the output of the motionless IMU and averaging the results from the three runs. The units for the variance of the angular rates are degrees squared per second squared. An example plot of the angular rate value over time is shown in Figure 3.34. The variance

of the angular displacement states,  $\phi$ ,  $\theta$ , and  $\psi$ , are determined by integrating the angular rates and averaging the variances of the results from the three runs. The units of the angular displacement variances are degrees squared. The final values for  $\mathbf{Q}_k$  are found in Equation (3.128).

$$\sigma^2 = 2\beta\rho \quad (3.127)$$

$$\mathbf{Q}_k = \begin{bmatrix} 4.3E-9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4.3E-9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9.2E-9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.6E-5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.6E-5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.6E-5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8.2E-4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0E-4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.2E-5 \end{bmatrix} \quad (3.128)$$

3.5.1.3 *Measurement Model Development.* The measurement model is de-

fined as

$$\delta \mathbf{z}_k = \mathbf{H} \delta \mathbf{x}_k + \mathbf{v}_k \quad (3.129)$$

with observation matrix  $\mathbf{H}$  defined as

$$\mathbf{H} = \begin{bmatrix} -55 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1898 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1898 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (3.130)$$

These values were found by performing parametric sweeps and visually matching the output plots, as described in the inertial measurement algorithm. Note that for a perturbation measurement model, the  $\mathbf{H}$  matrix is the negative of the true observation matrix.

The  $\mathbf{z}_k$  vector in the measurement model is comprised of the frame to frame displacement estimates provided by EIS. EIS is a particularly unique kind of measurement, because it offers relative measurements and not absolute. The displacement is given from the  $\mathbf{x}_{\text{true}}$  at the last time of frame capture, which is estimated by  $\hat{\mathbf{x}}$  in the Kalman filter. The perturbation model must account for this. The new  $\delta \mathbf{z}_k$  is found to be

$$\delta \mathbf{z}_{k_{\text{cf}}} = \mathbf{z}_{k_{\text{cf}}} - \mathbf{H} \left[ \mathbf{x}_{k_{\text{cf}}} - \hat{\mathbf{x}}_{k_{\text{pf}}} \right]. \quad (3.131)$$

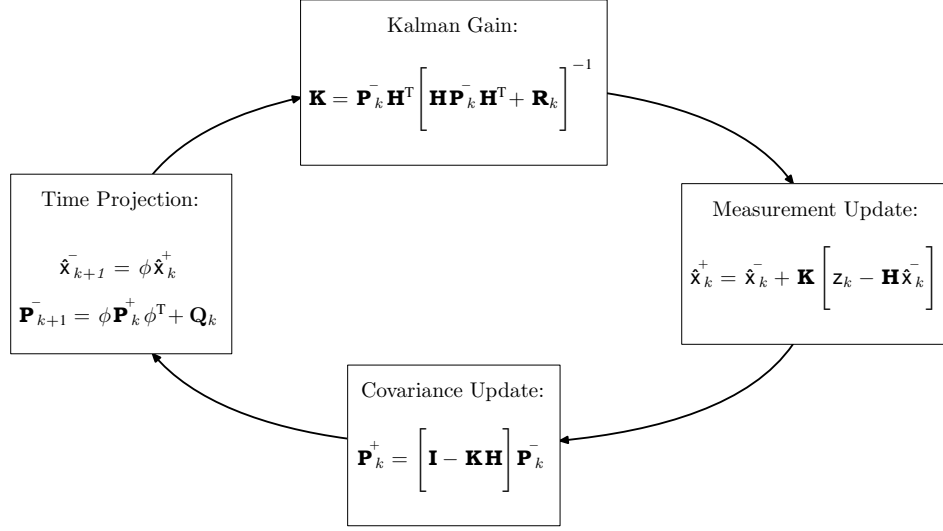
where  $k_{\text{cf}}$  is the current frame time and  $k_{\text{pf}}$  is the previous frame time. The full perturbation measurement model is then

$$\delta \mathbf{z}_{k_{\text{cf}}} = \mathbf{H} \delta \mathbf{x}_{k_{\text{cf}}} - \mathbf{H} \hat{\mathbf{x}}_{k_{\text{pf}}} + \mathbf{v}_{k_{\text{cf}}}. \quad (3.132)$$

To make the appropriate changes to the Kalman filter equations in Figure 3.35, replace every occurrence of  $\hat{\mathbf{x}}_k$  with  $\delta \hat{\mathbf{x}}_k$ , and every occurrence of  $\mathbf{z}_k$  with  $\delta \mathbf{z}_k$ , and proceed as normal.

3.5.1.4 *Measurement Variance Determination.* The  $\mathbf{R}_k$  matrix values de-

fine how much of EIS measurement is incorporated, or how little. When the detector value

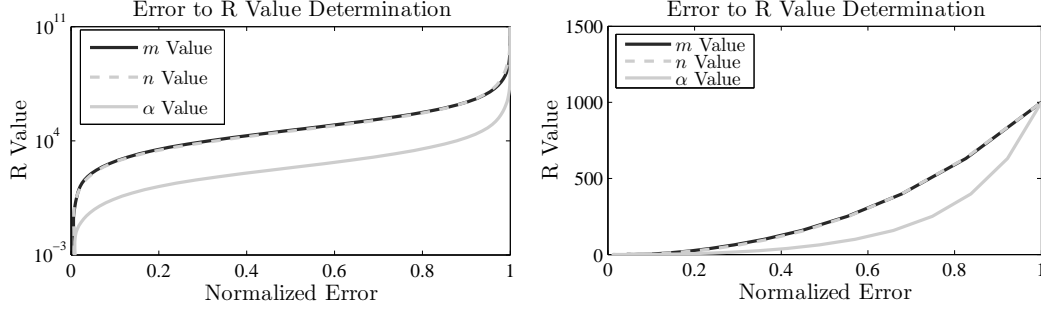


**Figure 3.35. Kalman Filter Equations [20:219].**

is near zero, all of the EIS measurement is desired. When the detector value is close to unity, all of the IMU measurement is desired.

The data from one of the three stationary test sets is used to determine the precise relationship between  $\mathbf{R}$  and the amount of EIS measurement to incorporate. The IMU was left motionless on a table, thus the true values of the angular rates are zero for all time. EIS measurements are simulated by a zero vector. The state estimate  $\hat{\mathbf{x}}_k$  of the filter should then be close to zero if EIS is fully incorporated, and deviate from zero if none of the EIS measurement is incorporated. This corresponds to the total error. The state estimate should have minimum error if all of the EIS estimate is fully incorporated, and maximum error if none of the EIS estimate is incorporated.

RMS errors were collected for values of  $\mathbf{R}$  ranging from  $1 \times 10^{-5}$  to  $1 \times 10^{15}$ . Maximum RMS error occurs when no EIS information is incorporated into the measurement, and minimum RMS error when EIS is incorporated into the measurement. The left plot in Figure 3.36 shows the normalized error to  $\mathbf{R}$  value relation. These complex curves are difficult to match with a simple equation. However if the view is constrained to the linear portions of the curves, as shown in the right plot in Figure 3.36, the curves resemble



**Figure 3.36. R Value Error Plot.**

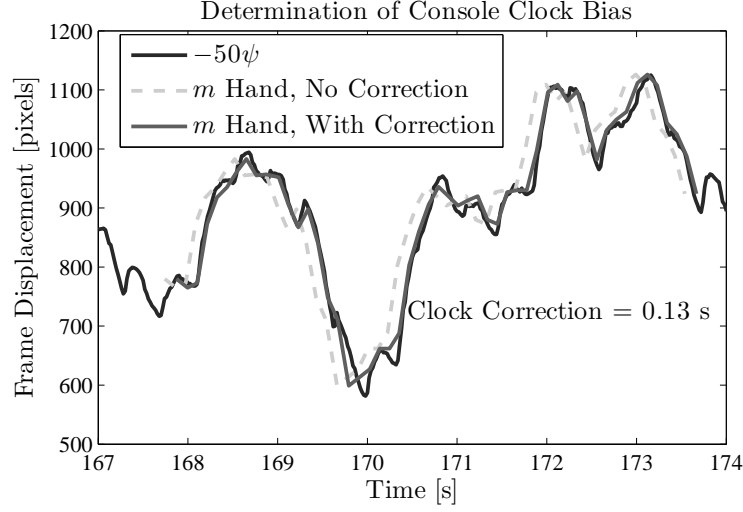
exponential form. Thus the  $R(D)$  equations should be of some exponential form

$$\eta e^{\gamma D}. \quad (3.133)$$

The final values for the  $\eta$  and  $\gamma$  coefficients are determined by using parametric sweeps. these sweeps were conducted during the moving object hallway test, as presented in Section 4.3.3. The coefficients are chosen so as to minimize the total RMS error for the run. These values are thus optimized for the specific environment of building hallways. Depending on the scene environment of the platform, these values may be altered to provide better results. The final  $R(D)$  equations are

$$\begin{aligned} R_m &= (1 \times 10^{-3}) \times e^{8D} \\ R_n &= 0.1 \times e^{6D} \\ R_\alpha &= (1 \times 10^{-3}) \times e^{18D}. \end{aligned} \quad (3.134)$$

*3.5.1.5 Camera to IMU Bias Determination.* A bias exists between the camera clock and the IMU clock. This bias prevents effective fusion of optical and inertial data. It is unknown whether this bias is random at turn on, or whether it is constant for every run as a hardware latency. The bias must be known in order to match up frame capture times to the appropriate IMU times. This bias was determined to be 0.13 seconds by matching up the hand calculated displacement to the displacement plot output the IMU, as shown in Figure 3.37.



**Figure 3.37. Determination of Console Clock Bias.**

With the necessary Kalman filter components now specified, optimal fusion of the optical flow estimates and inertial data is achieved. The Kalman filter corrects the output of the IMU by integrating optical flow measurements. The result is a more accurate angular rate and angular displacement estimate contained in  $\hat{\mathbf{x}}$ . To determine the optimal frame to frame displacements, sum the angular displacements in between frame times and multiply the result by the appropriate  $\mathbf{K}_\phi$ ,  $\mathbf{K}_\theta$ , or  $\mathbf{K}_\psi$  coefficient, as was done in the inertial measurement Algorithm.

These motion parameter estimates are then low pass filtered, and the local motion extracted. Compensation on the image is then accomplished by warping the image by the negative values of the local motion estimate. The result is a stabilized video feed.

*3.5.2 Optical Flow with Inertial Fusion Summary.* This concludes the development of the novel optical flow with inertial fusion algorithm. EIS is implemented in an optimal fashion by integrating information from optical and inertial sensors. The optical flow with inertial fusion algorithm is both fast and accurate, and is resistant to errors caused by imaging effects such as blurring and moving objects.

### 3.6 *Summary*

Five solutions to the motion estimation problem are presented, encompassing all four of the major classes of EIS. The five algorithms are template matching, feature detection, optical flow, inertial measurement, and the novel optical flow with inertial fusion algorithm. Each algorithm is well-structured, and presents a unique approach to the EIS problem. The template matching algorithm is conceptually straightforward, does not incur a large computational cost, and performs well in estimating small image displacement, however for the purpose of EIS on highly dynamic mobile robotic systems it should not be used. The feature detection algorithm is effective when image blurring and moving objects are not significant, but it is computationally expensive. Optical flow is both fast and effective even in the presence of image blurring. However it too incurs large errors in the presence of moving objects. Inertial measurement is unique because it is inherently invariant to image blurring and moving objects, but its accuracy completely depends upon the specific IMU used. Optical flow with inertial fusion is both the fastest and most accurate method of EIS for use on highly dynamic mobile robotic platforms, even in the presence of image blurring and moving objects.

## IV. Simulation and Experimental Analysis

**T**he performance of the five stated algorithms are now considered. It is necessary to compare the algorithms using a standard evaluation procedure in order to determine the best algorithm for EIS on highly dynamic mobile robotic systems. Optical flow with inertial fusion is shown to be the most effective algorithm, and also the fastest.

First, non-inertial EIS is tested. A video truth model is developed to simulate camera motion of a robot walking through a hallway. The truth model generates precise frame to frame motion parameter values for image translation and rotation, with the added capability of injecting image blur and moving objects into the video sequence. Each non-inertial EIS algorithm is utilized upon the truth model, and its estimated values compared to the true values. In this way an accurate and standard evaluation is achieved.

To begin, only translation is considered in the video truth model. Then translation and rotation is combined in the model. Then translation, rotation, and blurring is incorporated. Finally, translation, rotation, blurring, and moving objects are used. It is shown that optical flow performs the best of the non-inertial EIS algorithms.

Next, inertial EIS is tested. This is done by live test data of a camera and IMU affixed to the DAGSI Whigs<sup>TM</sup> mobile platform as it navigates through a hallway. True motion parameter values are estimated using a manual determination. These values are compared against optical flow, feature detection, inertial measurement and the novel optical flow with inertial fusion. Additionally, a simulated moving object is injected into the hallway video and the algorithm performance is rated for each of the methods. Of the four methods, optical flow with inertial fusion offers superior performance.

Note that global motion detection is disregarded for the purposes of evaluating the algorithms. The effectiveness of estimating complete image motion is desired. For perfect stabilization, the final video is completely stationary.

### 4.1 Truth Model Description

To rate the effectiveness of a particular EIS algorithm, a simulated truth model is created. This truth model allows for the observation of the exact motion parameters between frames. Knowing these values, we can describe the errors associated with a given EIS algorithm.

The base image used for the truth model is shown in Figure 4.38. It is a hallway scene, which captures the scene environment of the live test data. Thus results from the truth model are comparable to expected results of EIS operating on the live test data.

The input parameters of the truth model generator function are the maximum values for rotation,  $\theta$  translation,  $T$ , blur angle,  $\gamma$ , blur length,  $\xi$ , and blur frequency,  $f$ . The function then generates random values for these parameters according to a uniform distribution from the negative maximum to the positive maximum. The function takes a large image,  $1024 \times 1280$ , and rotates it by random  $\theta$ . The rotated image is cropped to match the original  $1024 \times 1280$  size. A window of size  $512 \times 640$  is then taken from the center of the rotated image, plus the random horizontal and vertical translation values  $T$ . This allows for perfect simulation of  $m$ ,  $n$ , and  $\alpha$  for the duration of the video.

In the truth model, the actual displacement values of  $m$ ,  $n$ , and  $\alpha$  are determined as an offset from the original base image. In order to calculate the true frame to frame displacements, the current frame value is subtracted by the previous frame value. This results in accurate frame to frame values, which are useful to compare against the outputs of the EIS algorithms.

The effects of blur are captured by parameters  $\gamma$ ,  $\xi$ , and  $f$ . The values of  $\gamma$  and  $\xi$  are input into a point spread function (PSF) which is applied to the image at a regular interval  $\frac{1}{f}$ .

The determination of error between the EIS parameter estimate and the true parameter value is accomplished by subtracting the estimated frame to frame displacement vector from the true frame to frame displacement vector. The residual vector is the estimation error for the duration of the test. Using the root-mean-squared (RMS) value of this error



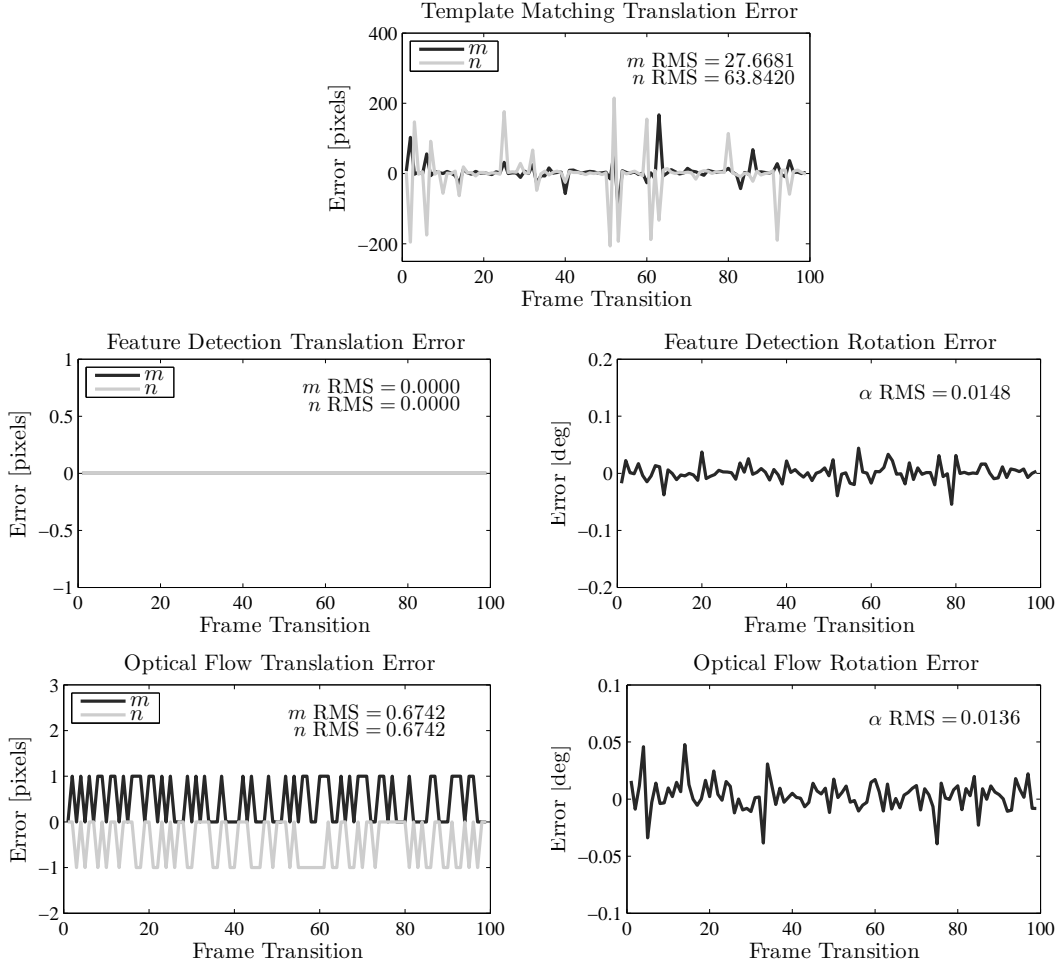
**Figure 4.38. Base Image for Truth Model.**

vector is effective to compare the overall performance of an EIS algorithm for a particular run.

Note that algorithms utilizing inertial data cannot be used on this truth model. This is because estimated IMU values are not determined for a particular video sequence. However, the development of such a model would be useful.

## ***4.2 Evaluation of Non-Inertial EIS Algorithms***

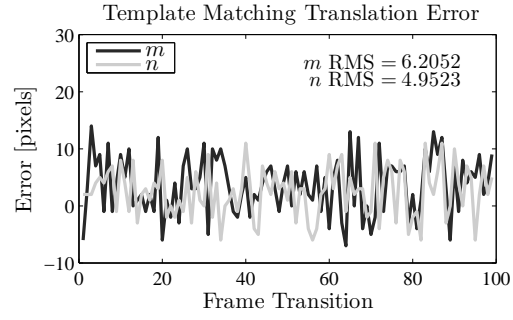
*4.2.1 Translation Testing.* With the truth model described, simulation and evaluations are now considered. Translation testing is performed upon each of the three algorithms. The truth model is given a uniformly sampled 100 pixel maximum movement in the  $m$  and  $n$  directions. This value of pixel movement was chosen because it approximates the actual maximum pixel movement for the live data set.



**Figure 4.39. Translation Test Performance.**

**Table 4.2. Translation Test Performance.**

	$m$ Error		$n$ Error		$\alpha$ Error	
	RMS	$\sigma$	RMS	$\sigma$	RMS	$\sigma$
<b>Template Matching</b>	27.67	27.41	63.84	64.06	N/A	N/A
<b>Feature Detection</b>	0.0	0.0	0.0	0.0	0.01°	0.01°
<b>Optical Flow</b>	0.67	0.50	0.67	0.50	0.01°	0.01°



**Figure 4.40. Template Matching Estimation Error, Short Translation Test.**

**Table 4.3. Average Time for One Estimation Loop**

	Time [s]
Template Matching	2.5
Feature Detection	4.6
Optical Flow	1.1
Inertial Measurement	0.6

The results are shown in Figure 4.39 and Table 4.2. Both feature detection and optical flow prove to be reliable algorithms for detecting translation and rotation. Both algorithms have translation RMS error value less than one pixel, and rotation RMS error near zero. The template matching algorithm fairs far worse, with RMS error of approximately 28 pixels in the  $m$  direction and 64 pixels in the  $n$  direction. There is no rotation error for template matching because the algorithm does not detect rotation.

The template matching algorithm is capable of better performance when the image translation is not as extreme. See Figure 4.40 for the performance of the template matching algorithm using a 50 pixel movement maximum. Here the RMS error is approximately 6 in the  $m$  direction and 5 in the  $n$  direction. Thus the template matching algorithm can be effectively used for small displacement estimation.

The speeds of each method are also determined from the translation testing. The mean required time for optical flow is 1.1 seconds for each estimation loop. Feature detection requires 4.6 seconds. Template matching requires 2.5 seconds per loop. Inertial measurement is evaluated later during hallway test in Section 4.3.2, but its value of 0.6 seconds is listed here for consistency. These times are presented in Table 4.3. The simulations were conducted on a personal laptop running a 2.1 MHz processor. With faster hardware and speed optimized code the algorithms will run much faster, however these loop time values provide insight into relative time requirements between the three approaches.

*4.2.2 Translation and Rotation Testing.* A rotation of  $6^\circ$  is now incorporated in the video truth model. This choice of rotation is based upon the approximate maximum rotation of the live data set. The resulting error plots and data are shown in Figure 4.41 and Table 4.4. Even in the presence of rotation, both the feature detection algorithm and the optical flow algorithm do an excellent job of estimating motion. The RMS error is held to 5 pixels in the  $m$  and  $n$  directions for both algorithms. Errors in rotation are negligibly small.

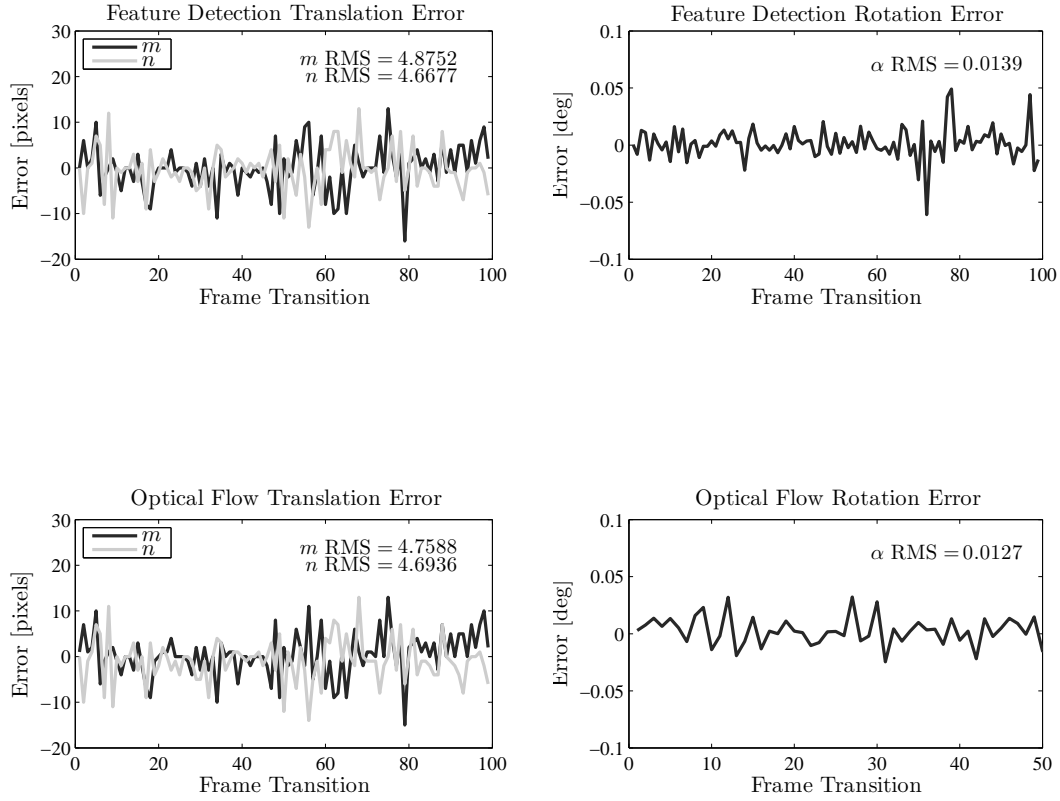


Figure 4.41. Translation and Rotation Test Performance.

Table 4.4. Translation and Rotation Test Performance.

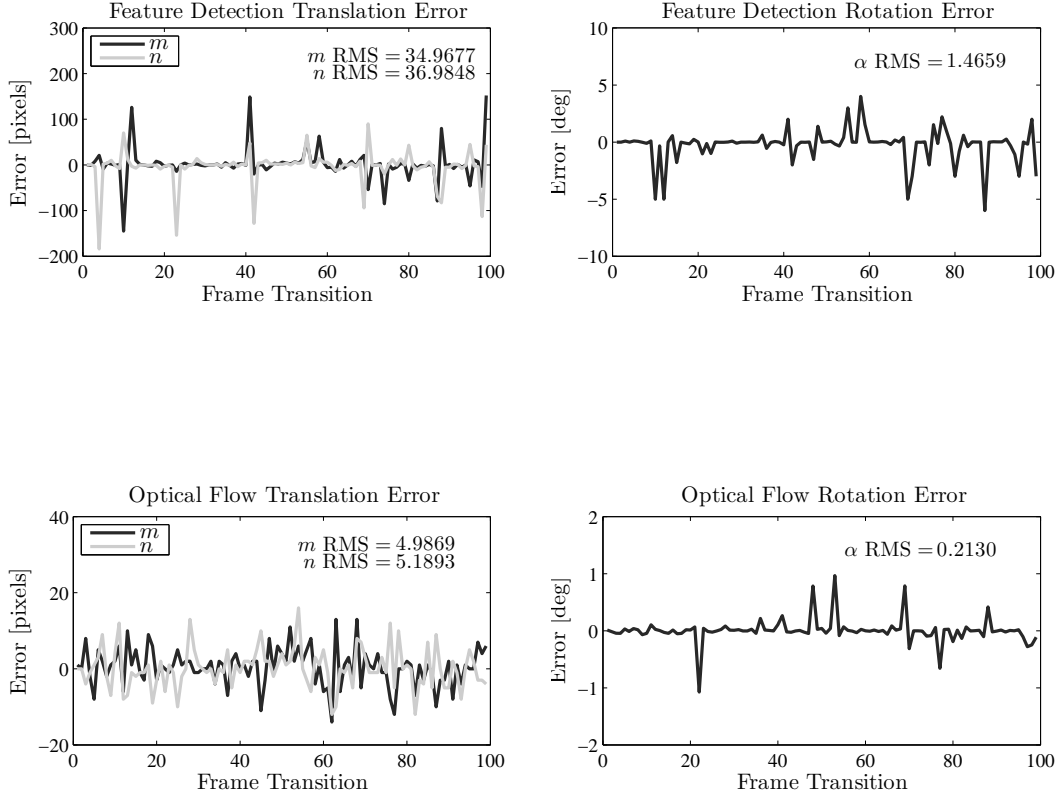
	$m$ Error		$n$ Error		$\alpha$ Error	
	RMS	$\sigma$	RMS	$\sigma$	RMS	$\sigma$
Feature Detection	4.88	4.90	4.67	4.67	0.01°	0.01°
Optical Flow	4.76	4.77	4.69	4.63	0.01°	0.01°

*4.2.3 Effects of Blurring.* When a random blur function is incorporated into the video truth model, the results shown in Figure 4.42 and described in Table 4.5 are observed. Blurring produces large errors in the feature detection algorithm. RMS errors are increased to 35 pixels in the  $m$  direction, and 37 pixels in the  $n$  direction. Significant rotation error also results, with an RMS value of  $1.5^\circ$ .

The optical flow algorithm maintains good performance, however. RMS errors are held to 5 pixels in the  $m$  and  $n$  directions, and rotation error is approximately  $0.2^\circ$ .

*4.2.4 Effects of Moving Objects.* A moving object is now added to the video truth model. A large black box of pixel size  $300 \times 300$  is sent across the screen at a constant speed. The block first enters the screen at frame 51, and exits around frame 90. A picture of the box is shown in Figure 4.44. Errors are greatly increased in the presence of the moving object. For feature detection, RMS pixel errors are 38 and 39 in the  $m$  and  $n$  directions, respectively, and  $8.5^\circ$  in rotational RMS error. Optical flow experiences RMS pixel errors of 24 in the  $m$  direction, 38 in the  $n$  direction, and  $3.1^\circ$  in RMS rotational error.

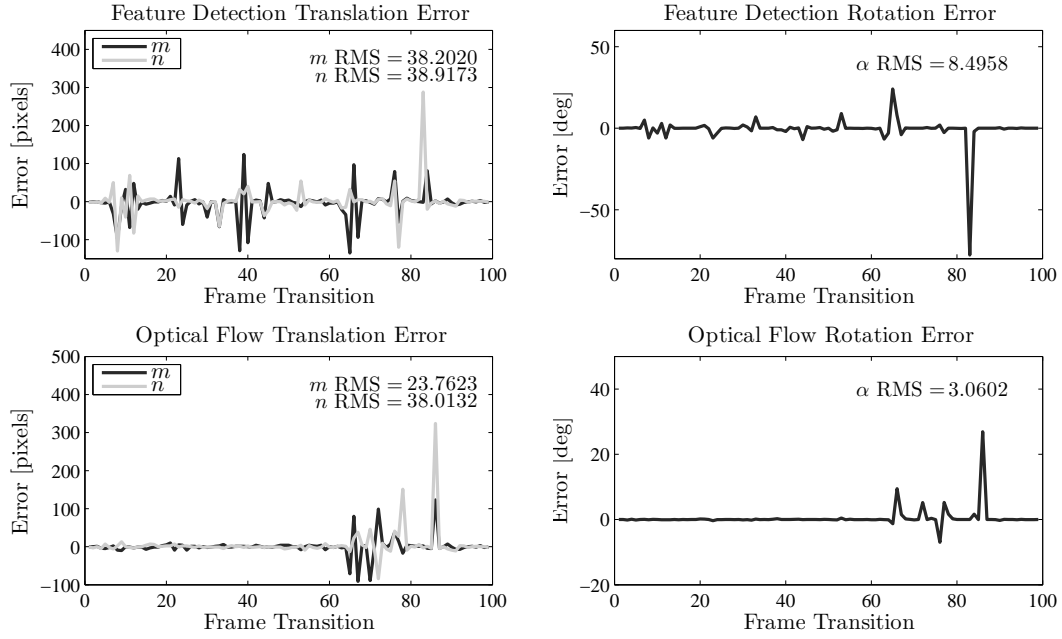
*4.2.5 Non-Inertial EIS Summary.* This concludes the evaluation of non-inertial EIS algorithms. Of the non-inertial algorithms, optical flow is the only algorithm robust against image blurring, whereas feature detection estimates are significantly degraded. However, both optical flow and feature detection are effective for estimating large image displacements, both in translation and rotation. The template matching algorithm is shown to be the weakest performer. However, when image displacements are constrained to 50 pixels performance is acceptable with template matching. None of the non-inertial EIS algorithms are capable of effective motion estimation in the presence of moving objects. Inertial measurement performance is rated next, followed by optical flow with inertial fusion.



**Figure 4.42. Translation, Rotation, and Blur Test Performance.**

**Table 4.5. Translation, Rotation, and Blur Test Performance.**

	$m$ Error		$n$ Error		$\alpha$ Error	
	RMS	$\sigma$	RMS	$\sigma$	RMS	$\sigma$
<b>Feature Detection</b>	34.97	35.09	36.98	36.93	1.47°	1.45°
<b>Optical Flow</b>	4.99	4.99	5.19	5.21	0.21°	0.21°



**Figure 4.43.** Translation, Rotation, Blur, and Moving Object Test Performance.

**Table 4.6.** Translation, Rotation, Blur, and Moving Object Test Performance.

	$m$ Error		$n$ Error		$\alpha$ Error	
	RMS	$\sigma$	RMS	$\sigma$	RMS	$\sigma$
<b>Feature Detection</b>	38.20	38.15	26.01	26.05	3.33°	3.43°
<b>Optical Flow</b>	28.93	29.06	38.44	38.41	4.21°	4.21°



Figure 4.44. Video with Moving Object

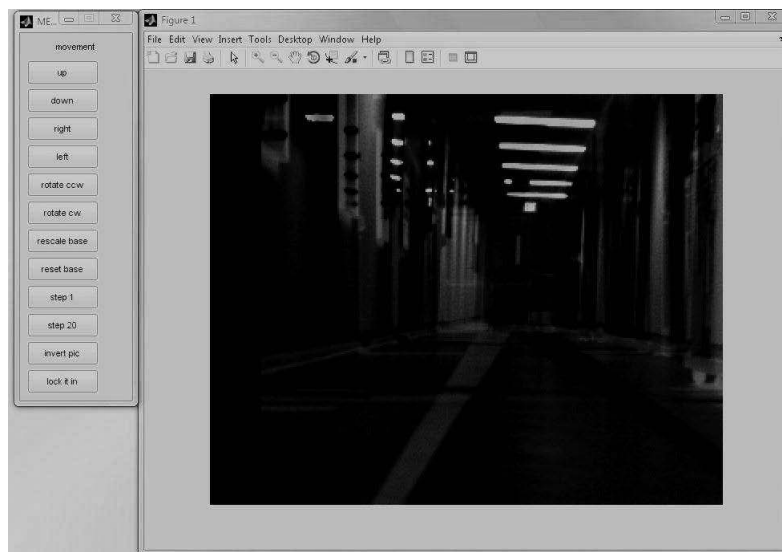


Figure 4.45. Hand Determination Snapshot.

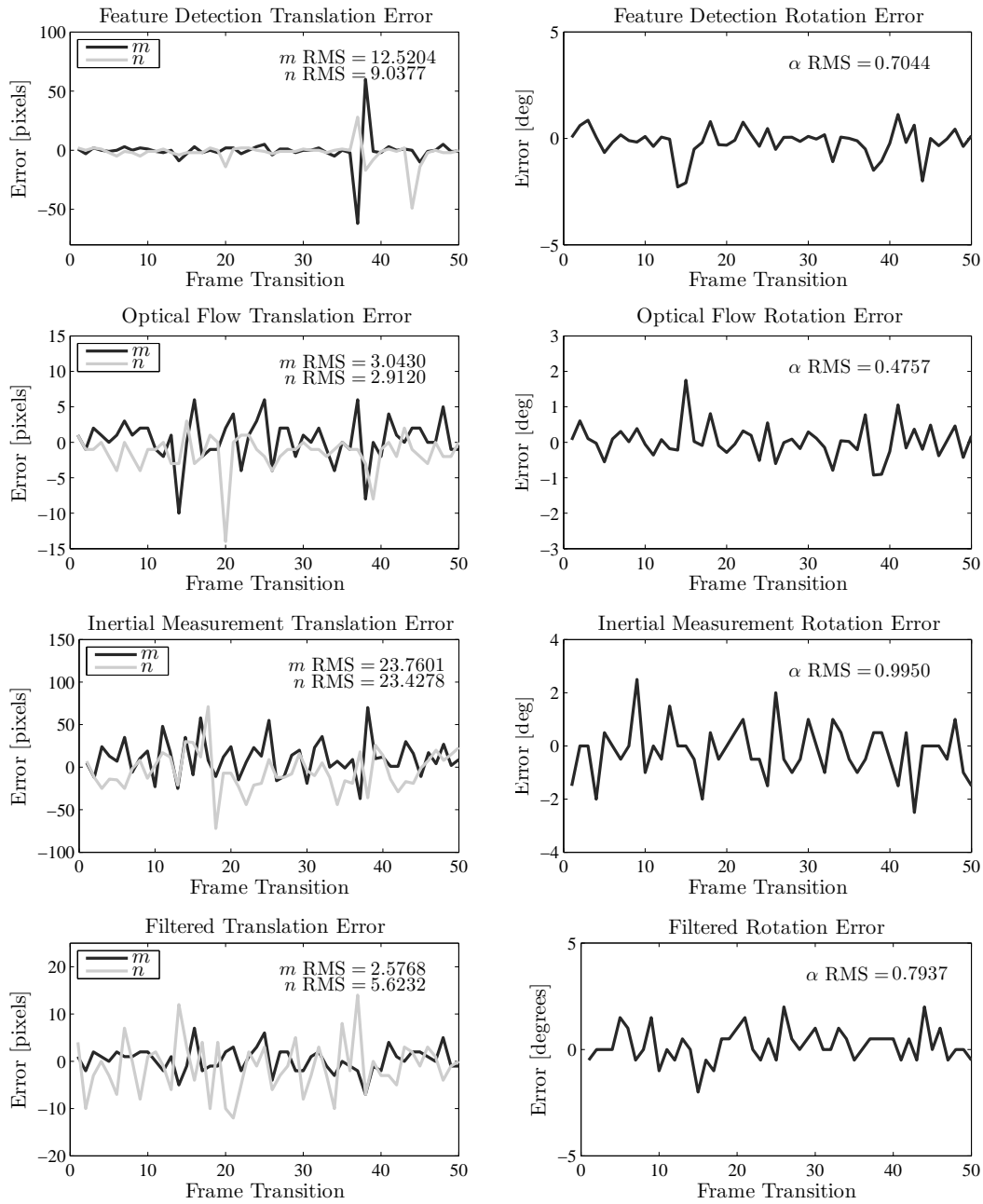
### 4.3 *Evaluation of Inertial Algorithms*

To rate the effectiveness of EIS using inertial measurement, an IMU is placed on the DAGSI Whegs<sup>TM</sup> platform, and video and IMU data is collected as the robot navigates through the hallway. This allows for analysis of the IMU performance. Precise truth values for the motion parameters for the hallway tests are not available, however, a systematic process is developed to manually approximate the true motion parameter values. Though the approximations are exact only to a few pixels, they are accurate enough to compare the performance of the EIS algorithms in a conclusive manner.

*4.3.1 Manual Determination of Motion Parameter Estimates.* Because of the lack of a truth model for the IMU values, a manual determination of frame movement is accomplished. This is done systematically using the program shown in Figure 4.45. The program displays the difference between two images, and provides controls to alter the image translation and rotation, invert the image intensities, and zoom in on the images. When a good match between two images is visually determined, the displacements are saved.

After this process is conducted on the video sequence, the resulting motion estimates are used to create a new stabilized video. The parameters are then fine tuned visually, correcting for any movement the video appeared to undergo. After several iterations of this process, a stable video is achieved, and the true frame to frame motion parameter estimates well approximated.

The manual determination is very accurate, except in some cases of image blurring. Image blurring presents a difficulty to ascertain true locations of objects in an image. Thus errors are introduced into the determination. However, significant blurring is not frequent, and it does not degrade the accuracy by more than a few pixels when it is present.



**Figure 4.46. Hallway Test Performance.**

**Table 4.7. Hallway Test Performance.**

	$m$ Error		$n$ Error		$\alpha$ Error	
	RMS	$\sigma$	RMS	$\sigma$	RMS	$\sigma$
<b>Feature Detection</b>	12.52	12.65	9.03	8.90	0.71°	9.69°
<b>Optical Flow</b>	3.04	3.05	2.91	2.53	0.48°	0.48°
<b>Inertial Measurement</b>	23.76	21.53	23.43	23.23	1.00°	0.98°
<b>Optical Flow with Inertial Fusion</b>	2.58	2.59	5.62	5.55	0.79°	0.78°

*4.3.2 Hallway Testing.* The accuracy of the manual method to determine frame displacements is good enough to compare performance of one algorithm against the other. Figure 4.46 and Table 4.7 shows the performance of the inertial measurement EIS algorithm along with the feature detection, optical flow, and optical flow with inertial fusion algorithms. Feature detection has acceptable performance, except for an occasional spike. RMS errors are 13 pixels in  $m$  direction, 9 pixels in the  $n$  direction, and rotational RMS error of 0.7°. The optical flow algorithm works well, with RMS error approximately 3 pixels for both the  $m$  and  $n$  directions, and rotational RMS error is held to 0.4°. The inertial measurement algorithm does not perform well, with RMS error of 24 pixels in the  $m$  direction and 23 pixels in the  $n$  direction, and rotational RMS error of 1.0°. The optical flow with inertial fusion algorithm has RMS pixel errors of 3 in the  $m$  direction, 6 in the  $n$  direction, and 0.8° in rotational RMS error. Observe that the filtered estimate incurs errors slightly greater than the optical flow algorithm. This is because it integrates too much of the IMU estimate. Certain frames have a high detector value even though the EIS estimate is reliable. This is the unfortunate tradeoff for better performance in the presence of moving objects. However, errors are still acceptable even with the increase. The exact nature of this tradeoff is dependent upon the  $\mathbf{R}(\mathbf{D})$  equations used to determine the measurement uncertainty matrix for the Kalman filter.

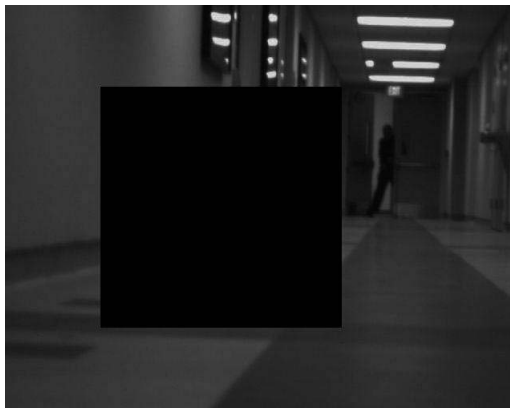
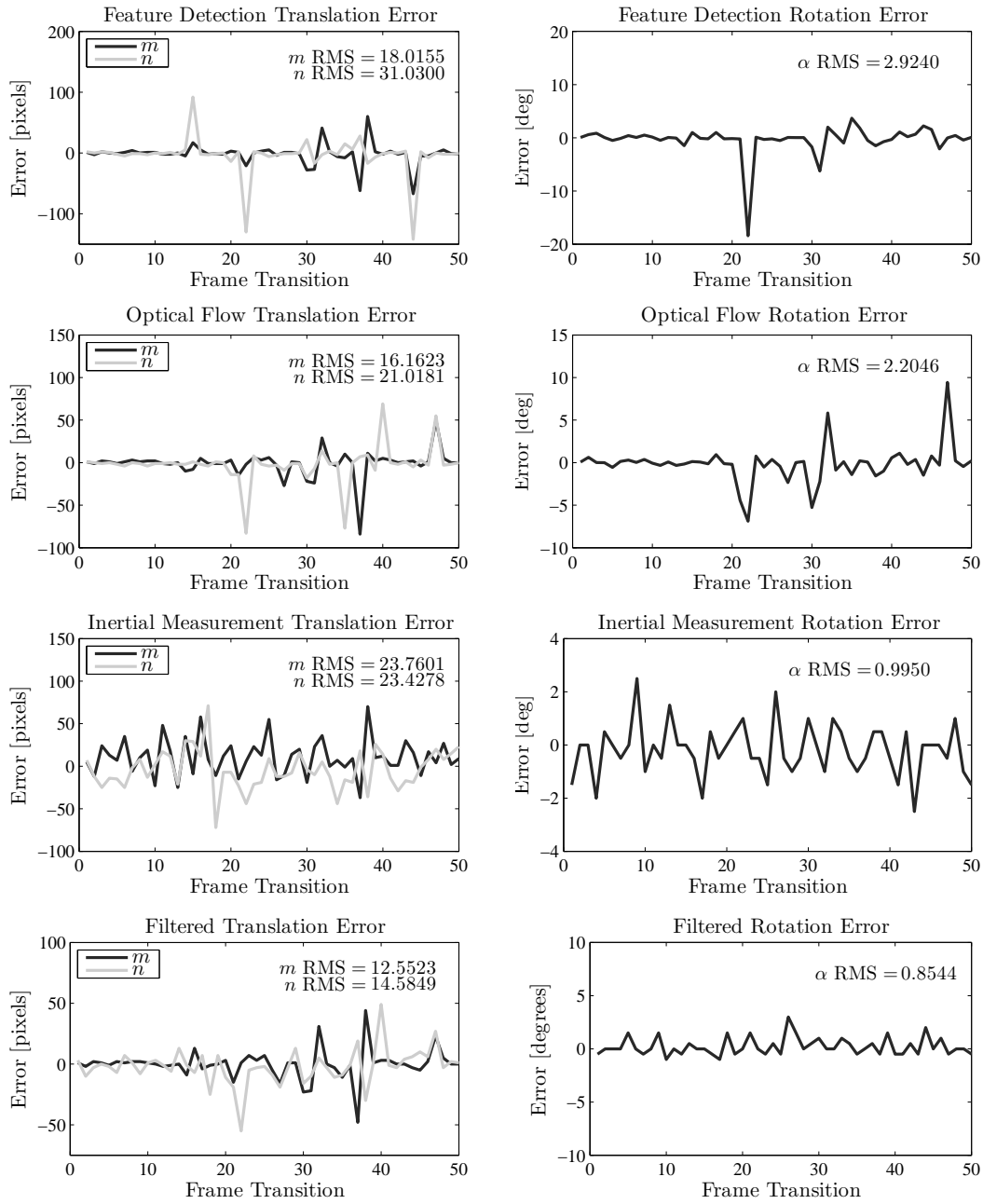


Figure 4.47. Hallway Test with Moving Object.

Table 4.8. Hallway with Moving Object Test Performance.

	<i>m</i> Error		<i>n</i> Error		$\alpha$ Error	
	RMS	$\sigma$	RMS	$\sigma$	RMS	$\sigma$
Feature Detection	18.01	18.09	31.03	31.11	2.92°	2.93°
Optical Flow	16.16	16.29	21.08	21.11	2.20°	2.22°
Inertial Measurement	23.76	21.53	23.43	23.23	1.00°	0.98°
Optical Flow with Inertial Fusion	11.82	11.93	14.47	14.57	0.78°	0.83°



**Figure 4.48. Hallway with Moving Object Test Performance.**

The inertial measurement algorithm is ineffective because of the inherent errors of IMU gyro sensors. For inexpensive IMUs, these errors are large enough to significantly degrade EIS performance. However, if a precision IMU is used, the inertial measurement algorithm will perform much better. This would improve the accuracy of the optical flow with inertial fusion algorithm as well.

*4.3.3 Moving Object Simulation.* Moving objects are now injecting into the hallway video sequence. Figure 4.47 depicts the moving object. This allows for the operation of inertial measurement in the presence of moving objects. A large  $300 \times 300$  pixel black box is injected into the video sequence and sent across the video starting at frame 30.

Observe in Figure 4.48 that large errors are introduced into the feature detection and optical flow algorithm from frame 30 to the end of the video. Feature detection RMS pixel errors are 18 in the  $m$  direction, 31 in the  $n$  direction, and  $2.9^\circ$  in rotational RMS error. Optical flow performs slightly better, with RMS pixel errors of 16 in the  $m$  direction, 21 in the  $n$  direction, and  $2.2^\circ$  in rotational RMS error.

When inertial data is fused with the optical flow estimate, much better results are achieved. Using the novel optical flow with inertial fusion algorithm, RMS error is reduced to 12.5 in the  $m$  direction, 14.6 in the  $n$  direction, and  $0.85^\circ$  in rotational RMS error. Note that the final result yields a lower RMS error than either the IMU or the optical flow algorithms do on their own.

*4.3.4 Inertial Algorithm Summary.* In the presence of moving objects, using optical flow with inertial fusion results in a 27% reduction in RMS error in the  $m$  direction, a 31% reduction in RMS error in the  $n$  direction, and a 63% reduction in rotation RMS error, compared to the non-inertial best alternative optical flow algorithm. Averaging these values, optical flow with Inertial Fusion is capable of 40% lower RMS error than its non-inertial optical flow counterpart in the presence of moving objects.

Optimally fusing the inertial data with the optical flow motion estimates is an effective way to minimize errors introduced by moving objects. Optical flow estimates are incorporated when the image scene provides good features to track. Inertial measurement

estimates are incorporated when the image scene does not provide good features, and the optical flow estimates are inaccurate. The result is a robust EIS algorithm that operates effectively even in the presence of moving objects.

#### 4.4 *T-Significance Testing*

To obtain more insight into the fusion of inertial data, a T-significance test is conducted between the error plots of the optical flow algorithm and the optical flow with inertial fusion algorithm. The resulting p-values are

$$\begin{aligned} m : p - \text{value} &= 0.801 \\ n : p - \text{value} &= 0.762 \\ \alpha : p - \text{value} &= 0.303 \end{aligned} \tag{4.135}$$

We see that for the  $m$  and  $n$  directions, there is not much statistical difference between the performance of the optical flow algorithm and the optical flow with inertial fusion algorithm. This can be directly interpreted from the plots in Figure 4.48. Large spikes occur at the same frame transitions for both algorithms. This occurs because for these specific frame transitions, both optical flow and inertial measurement incur large estimation errors. The substitution of a bad IMU measurement is made for a bad optical flow measurement. Use of a higher grade IMU will improve the IMU measurements, resulting in greater statistical difference between the optical flow algorithm and the optical flow with inertial fusion algorithm.

However, even though both algorithms experience peak errors at the same frame transitions, these errors are significantly reduced.

#### 4.5 *Summary*

The four major classes of EIS algorithms can be used for effective motion parameter estimation. The performance of these algorithms in this particular use of EIS on highly dynamic mobile robotic platforms can be determined by developing a standard video truth model capable of simulating the video feed of a camera affixed to such a platform.

A video truth model is developed capable of such simulation, accurately depicting the effects of image translation, rotation, and blurring, and moving objects for a mobile robotic platform traveling down a hallway. Template matching, feature detection, and optical flow algorithms were all evaluated by this truth model. Optical flow is determined to be the best performing algorithm in the presence of image translation, rotation, and blurring. Feature detection and optical flow both have the same performance when blurring is not present, however feature detection requires a significant computational cost. Template matching is only accurate when frame displacements are small, and thus is not a viable option for EIS on highly dynamic mobile robotic platforms.

Using live data collected from a camera and IMU affixed to the DAGSI Whegs<sup>TM</sup> robotic platform, the effectiveness of the inertial measurement algorithm and the novel optical flow with inertial fusion algorithm is rated. The inertial measurement algorithm is shown to be an ineffective stabilizer on its own. This is because the IMU used in the experiment is corrupted by significant gyro sensor errors. However, inertial measurement does have the beneficial property of invariance to moving objects.

Optimally combining these two methods in the novel optical flow with inertial fusion algorithm is the most effective way to integrate the benefits of both. This is shown by injecting the video sequence with a large moving object and comparing the performance of the optical flow algorithm to the performance of the optical flow with inertial fusion algorithm. The result is a 40% drop in RMS error for the duration of the run. Further, the result is a lower RMS error than either the optical flow algorithm or the inertial measurement algorithm on their own.

## V. Conclusions

Electronic image stabilization is an important concern for work with video systems and video motion. Accurate and robust stabilization is necessary for complex video analysis, especially on highly dynamic mobile robotic platforms such as the DAGSI Whegs<sup>TM</sup>. EIS is a complex problem, and effective solutions require knowledge from many disciplines, including computer vision, Kalman filtering, and inertial navigation. In this thesis, the four main classes of EIS algorithms are presented; namely template matching, feature detection, optical flow, and inertial measurement. Each algorithm has unique benefits and drawbacks.

Template matching is an intuitive and straightforward method of EIS that is capable of effective frame to frame motion estimation in the presence of small image displacements. It requires average computational speed. However, in the presence of large image displacements template matching is ineffective. Because large image displacements are a frequent occurrence in EIS for highly dynamic mobile robotic platforms, template matching should not be used.

Feature detection is an effective EIS algorithm capable of precise motion parameter estimation in the presence of large image translation and rotation. However, it is computationally expensive. Further, image blurring significantly degrades performance.

Optical flow is the most accurate EIS algorithm in the presence of image blurring. It is effective for accurate estimation of large image translation and rotation as well. It is also a computationally fast algorithm. However, in the presence of moving objects, optical flow algorithms result in large estimation errors.

Inertial measurement is the only EIS algorithm invariant to moving objects. It is also the fastest algorithm. However, the accuracy of translation and rotation estimation depends upon the specific IMU used. This is due to the inherent drifting bias errors present within gyro sensors.

For robust image stabilization, it is necessary to combine the effectiveness of optical flow and inertial measurement. This is accomplished in the novel optical flow with inertial

fusion EIS algorithm contributed in this thesis. It utilizes the positive aspects of both optical flow and inertial measurement, and at the same time minimizes their drawbacks. Reliable translation and rotation estimation as provided by optical flow is incorporated when it is detected as being trustworthy. When it is detected that optical flow is no longer reliable, inertial measurement estimation of translation and rotation is incorporated. The result is a robust EIS algorithm capable of 40% reduction in RMS error compared to optical flow alone.

The numerical analysis of these algorithms is conducted using a video feed truth model simulating a robotic platform traveling down a hallway. The truth model is capable of portraying image translation, rotation, blurring and moving objects. Live data collected from a video camera and IMU affixed to the DAGSI Whegs<sup>TM</sup> platform is also used to rate algorithm performance.

The main contribution of this thesis is the novel electronic image stabilization algorithm, optical flow with inertial fusion. It uses Shi-Tomasi good features and pyramidal Lucas-Kanade optical flow fused with inertial data by way of a nine state discrete Kalman filter. No EIS algorithm to date uses optical flow and inertial data provided from an IMU by way of discrete Kalman filter.

The secondary contribution of this thesis is an EIS algorithm capable of effective stabilization on the DAGSI Whegs<sup>TM</sup> robotic platform. This highly dynamic mobile robotic platform is capable of traversing difficult terrain such as stairs and large rocks. To date, no EIS algorithm has been developed for this specific platform.

The third contribution of this thesis is an algorithm that specifically reduces the errors associated with image blurring and moving objects. To date, these effects have never been addressed by specific EIS algorithms.

The fourth contribution of this work is a numerical analysis performed on the four main classes of EIS. Template matching, feature detection, optical flow, and inertial measurement algorithms are evaluated and compared using video truth models. To date, a numerical comparison of these methods has never been done.

## 5.1 Future Work

There are several aspects of this work which could be developed in the future to provide greater understanding into EIS.

*5.1.1 Speeding Up the Algorithm.* Speeding up the algorithm using novel techniques would allow for better integration into larger systems. One way this can be done is by converting the algorithm into C/C++ language.

*5.1.2 Determination of Global Motion Filter Coefficients.* The low pass filter used to detect global motion was selected as the most visually acceptable second-order filter. Though its performance is acceptable, a more rigorous determination is possible. Desired global motion must be characterized by some method. One method could be to maximize the desired motion of the robot. For example, the robot could walk in a figure eight loop for several minutes. The stabilized video should track precisely on the center of the camera during extreme right turn and left turn motions. A low pass filter whose cutoff frequency allows for this kind of track would have the necessary coefficients. Note however that this would only apply for  $n$  directional movement. A similar approach could be used to find coefficients for  $m$  and  $\theta$  movements. Create a course which will maximize the desired motion, and then find the filter cutoff frequency that keeps the stabilized video centered on the image frame.

*5.1.3 Inertial Truth Model Development.* The truth model used to simulate the video sequence of a robotic platform traveling down a hallway has one major shortfall; it does not provide simulated IMU values. Thus algorithms using inertial fusion cannot be evaluated with the truth model. The development of such a truth model would allow for evaluation and optimization of inertial EIS algorithms.

*5.1.4 Manual Determination Accuracy.* Hand determination of the pixel movement was acceptable for comparing the effectiveness of the algorithms amongst themselves, however a more accurate method is desirable for better analysis. This could be done through several different ways. One method could be using a high grade IMU also placed

on the platform, providing more accurate angular rate data. Another method would be to use a laser system capable of estimating the precise angular motion of the platform, such as the Vicon system, which estimates attitude and position using IR lasers providing 360° of coverage.

*5.1.5 Detector Value for the Feature Detection Algorithm.* In order to be implemented in a Kalman filter, a feature detection algorithm must have some way to determine the variance of its estimates. This could be done using the number of feature correspondences, or a track error number similar to the Lucas-Kanade optical flow function. Other unique metrics can be developed. Once a reliable detector value is determined frame to frame, feature detection can be incorporated with inertial data to provide an optimal motion parameter estimate.

*5.1.6 Numerical Analysis of Other Methods.* Only the four main classes of EIS algorithms were considered for truth model evaluation in this thesis. However, several other techniques exist. These can be implemented and rated according to the evaluation process used in chapter four.

*5.1.7 Console Bias Determination.* The source of the time bias between the camera clock and the IMU clock is unknown. For the purposes of this work, a one time determination was acceptable for testing. However, for realtime operation of the EIS algorithm, the precise nature of the error must be known. If this is not done, the fused results of the EIS estimate and the inertial data will be significantly inaccurate.

*5.1.8 Effects of Angular Rotation.* Rotation was set to a maximum of 6° in the truth model because it approximated the maximum rotation in the live data set. However, more analysis on the effects of rotation on the performance of the algorithms can be conducted. Further, ways to lessen the errors caused by rotation can be studied.

*5.1.9 Use of a Higher Grade IMU.* The IMU used had significant errors in its angular rate outputs. This resulted in reduced accuracy for the motion parameter

estimates. Using a higher grade IMU would allow for much better performance of the optical flow with inertial fusion algorithm.

## **5.2 Summary**

Stabilization is prevalent in many aspects of life. Of particular importance is image stabilization and visual systems. For vision systems affixed to highly dynamic mobile robotic platforms, EIS is a necessary preparatory step towards more complex analysis on the video feed. Because of large frame displacement and undesired image effects such as blurring and moving objects, this particular application of EIS is difficult. The novel optical flow with inertial fusion algorithm presented in this thesis is a viable solution to the problem. It integrates pyramidal Lucas-Kanade optical flow using Shi-Tomasi good features and inertial data provided by an IMU by optimally fusing the two by way of discrete Kalman filter. The algorithm is fast and effective, and is capable of robust stabilization in the presence of large image displacement, image blurring, and moving objects.

## Bibliography

1. M. A. Alharbi, *Fast Video Stabilization Algorithms*. M.S. Thesis. Air Force Institute of Technology, 2006.
2. P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*. Ganga-Jamuna Press, 2006.
3. A. S. Boxerbaum, J. Oro, G. Peterson, and R. Quinn, "The latest generation whegs robot features a passive-compliant body joint," *International Conference on Intelligent Robots and Systems*, September 2008.
4. M. J. Veth, *Fusion of Imaging and Inertial Sensors for Navigation*. Ph.D. Dissertation. Air Force Institute of Technology, 2006.
5. J. Shi and C. Tomasi, "Good features to track," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 593–600, 1994.
6. A. Amanatiadis, I. Andreadis, A. Gasteratos, and N. Kyriakoulis, "A rotational and translational image stabilization system for remotely operated robots," *IEEE International Workshop on Imaging Systems and Techniques*, pp. 1–5, 2007.
7. M. Ramachandran and R. Chellappa, "Stabilization and mosaicing of airborne videos," *IEEE International Conference on Image Processing*, pp. 345–348, 2006.
8. R. Kurazume and S. Hirose, "Development of image stabilization system for remote operation of walking robots," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1856–1861, 2000.
9. R. Szeliski, *Image Alignment and Stitching: A tutorial*. Microsoft Corporation, 2006.
10. S. L. Tanimoto, "Template matching in pyramids," *Computer Graphics and Image Processing*, vol. 16, no. 4, pp. 356–369, 08/01 1981.
11. G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA: O'Reilly Media, Inc., 2008.
12. I. Sobel and G. Feldman, "A 3x3 isotropic gradient operator for image processing," *In: Pattern Classification and Scene Analysis*, Duda, R. and Hart, P., John Wiley and Sons. (1973) 271-272.
13. C. Harris and M. Stephens, "A combined corner and edge detector," *Proceedings of The Fourth Alvey Vision Conference*, Manchester, vol. 15, pp. 147–151, 1988.
14. D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
15. R. Hartley and A. Zisserman, *Multiple View Geometry*. Cambridge University Press, 2003.

16. G. Golub and C. Van Loan, *Matrix Computations*. John Hopkins University Press, 1989.
17. B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of Imaging Understanding Workshop*, pp. 121–130, 1981.
18. B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.
19. D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology - 2nd edition*. The Institute of Electrical Engineers, 2004.
20. R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley and Sons, 1997.
21. R. Wolfson and J. Pasachoff, *Physics: With Modern Physics*. Addison-Wesley, 1999.
22. C. D. Kuglin and D. C. Hines, "The phase correlation image alignment method," *IEEE International Conference on Cybernetics and Society*, pp. 163–165, 1975.
23. P. S. Addison, *The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in Science, Engineering, Medicine, and Finance*. Bristol: Institute of Physics Publishing, 2002.
24. P. J. Rousseeuw, "Least median of squares regression," *Journal of the American Statistical Association*, vol. 79, no. 388, p. 871, Dec 1984.
25. M. A. Fischler, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, p. 381, June 1981.
26. M. Zuliani, *RANSAC for Dummies*. [www.mathworks.com](http://www.mathworks.com), 2008.
27. P. S. Maybeck, *Stochastic Models, Estimation, and Control*. Arlington, VA: Navtech Book and Software Store, 1994.
28. M. Hirooka, "Hierarchical distributed template matching," *Proceedings of SPIE, the international society for optical engineering*, vol. 3029, p. 176, 1997.
29. C. Guestrin, F. Cozman, and E. Krotkov, "Fast software image stabilization with color registration," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 19–24, 1998.
30. W. Zhu, K. Li, X. Gao, J. Gao, J. Li, and X. Zhang, "A real-time scheme of video stabilization for mine tunnel inspectional robot," *IEEE International Conference on Robotics and Biomimetics*, pp. 702–705, 2007.
31. B. S. Morse, D. Gerhardt, C. Engh, M. A. Goodrich, N. Rasmussen, D. Thornton, and D. Eggett, "Application and evaluation of spatiotemporal enhancement of live aerial video using temporally local mosaics," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

32. C. Morimoto and R. Chellappa, "Fast electronic digital image stabilization," *Proceedings of the 13th International Conference on Pattern Recognition*, vol. 3, pp. 284–288, 1996.
33. H. R. Pourreza, M. Rahmati, and F. Behazin, "An electronic digital image stabilizer based on stationary wavelet transform (swt)," *International Conference on Image Processing*, vol. 2, pp. II–383–6, 2003.
34. Z. Yong-xiang, "A method of resolving gyro zero drift in electronic stabilization system," *International Conference on Computer and Automation Engineering*, March 2009.
35. E. Ifeachor and B. Jervis, *Digital Signal Processing: a Practical Approach*. Pearson Education Unlimited, 2002.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
1. REPORT DATE (DD-MM-YYYY) 10-09-2009		2. REPORT TYPE Master of Science Thesis		3. DATES COVERED (From – To) Sep 2008 – Sep 2009	
4. TITLE AND SUBTITLE  Electronic Image Stabilization for Mobile Robotic Vision Systems				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)  Smith, Michael, J. 2 <sup>nd</sup> Lieutenant, USAF				5d. PROJECT NUMBER 09-219	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) 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/09-53	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Labs/RYRN, Attn: Dr. Jacob Campbell 2241 Avionics Circle WPAFB, OH 45433 DSN: 785-6127 x4154 (jacob.campbell@wpafb.af.mil)				10. SPONSOR/MONITOR'S ACRONYM(S)  AFRL/RYRN	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT  APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT <p>When a camera is affixed on a dynamic mobile robot, image stabilization is the first step towards more complex analysis on the video feed. This thesis presents a novel electronic image stabilization (EIS) algorithm for small inexpensive highly dynamic mobile robotic platforms with onboard camera systems. The algorithm combines optical flow motion parameter estimation with angular rate data provided by a strapdown inertial measurement unit (IMU). A discrete Kalman filter in feedforward configuration is used for optimal fusion of the two data sources. Performance evaluations are conducted by a simulated video truth model (capturing the effects of image translation, rotation, blurring, and moving objects), and live test data. Live data was collected from a camera and IMU affixed to the DAGSI Whigs mobile robotic platform as it navigated through a hallway. Template matching, feature detection, optical flow, and inertial measurement techniques are compared and analyzed to determine the most suitable algorithm for this specific type of image stabilization. Pyramidal Lucas-Kanade optical flow using Shi-Tomasi good features in combination with inertial measurement is the EIS algorithm found to be superior. In the presence of moving objects, fusion of inertial measurement reduces optical flow root-mean-squared (RMS) error in motion parameter estimates by 40%. No previous image stabilization algorithm to date directly fuses optical flow estimation with inertial measurement by way of Kalman filtering.</p>					
15. SUBJECT TERMS EIS; electronic image stabilization; optical flow; IMU; inertial measurement unit; image processing; computer vision; navigation					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT  UU	18. NUMBER OF PAGES 121	19a. NAME OF RESPONSIBLE PERSON Dr. Gilbert L. Peterson
REPORT U	ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) DSN: (937)255 – 6565, x4281 (gilbert.peterson@afit.edu)