# Digital Video Stabilization with Inertial Fusion

William John Freeman

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Electrical Engineering

Alfred L. Wicks

Daniel J. Stilwell

Amos L. Abbott

April 26, 2013

Blacksburg, VA

Keywords: Image Stabilization, Kalman Filter, Inertial Fusion.

Digital Image Stabilization with Inertial Fusion

William John Freeman

ABSTRACT

As computing power becomes more and more available, robotic systems are moving away from active sensors for environmental awareness and transitioning into passive vision sensors. With the advent of teleoperation and real-time video tracking of dynamic environments, the need to stabilize video onboard mobile robots has become more prevalent.

This thesis presents a digital stabilization method that incorporates inertial fusion with a Kalman filter. The camera motion is derived visually by tracking SIFT features in the video feed and fitting them to an affine model. The digital motion is fused with a 3 axis rotational motion measured by an inertial measurement unit (IMU) rigidly attached to the camera. The video is stabilized by digitally manipulating the image plane opposite of the unwanted motion.

The result is the foundation of a robust video stabilizer comprised of both visual and inertial measurements. The stabilizer is immune to dynamic scenes and requires less computation than current digital video stabilization methods.

Dedication

A lot of the work on this thesis was performed by my father's side in the hospital. His fight against cancer has given me strength and perseverance, for which I am eternally grateful. The uncompromising love of my family and friends has been my shining light at the end of the tunnel, thank you all for everything. I would also like to thank my partner in life, Erika, without her support none of this would have been possible.

# List of Figures

# List of Tables

# 1 Motivation

Robotic perception is generally performed with active range finding sensors and/or passive vision based sensors. Active sensors work great to avoid obstacles but lack the ability to distinguish between objects of similar volume and shape. Passive vision based sensors are able to distinguish between different objects and materials much in the same humans do with their eyes but require significant post-processing in order to extract 3 dimensional information of the scene.

Stabilization comes into play when vision sensors are placed on ground vehicles, and due to the dynamic nature of the vehicle, vibrations and oscillations are induced in the camera which cause a shaky and unstable video feed. The feed must be stabilized for 2 factors, to improve clarity for the viewer and to improve accuracy of computer vision algorithms. The first factor is obvious; it is difficult to view shaky-unstable video and can cause the operator to oversight key information in the video feed. The second factor is not so obvious; algorithms performed on images from moving vehicles are affected by serious problems related to vibrations and oscillations. Broggi et al. describe the errors are usually caused by the tight dependence between camera motion and the calibration parameters used [1]. In most cases - such as obstacle, pedestrian or lane detection - some features are extracted from the image and, using calibration parameters, their position and/or size is computed in the real world. Any moving vehicle is affected by oscillations. This causes a variation of camera orientation angles (with respect to a fixed reference system) that are used for distance computations. The problem becomes even harder for off-road tracks, where the path's coarseness causes larger oscillation [1].

The purpose of this thesis is to describe the solution of the stabilization problem through the fusion of visual and inertial measurements. The visual measurement is determined by tracking SIFT features in a video feed and solving for an affine model that describes the motion of the image plane. The inertial measurement is found by measuring the angular velocity of the camera through the use of an inertial measurement unit (IMU) and relating it to the motion in the image plane. These two measurements have complementary strengths and weaknesses; the visual measurement is slow and more accurate while the inertial measurement is fast yet error prone. Combining these two measurements through the use of a Kalman filter allows the stabilization system to exploit the strengths of both measurements. The remainder of this thesis is sectioned into chapters that go through the background, method and results of the video stabilizer.

## 1.1 Chapter 2 Summary—Literature Review

The literature review covers the background and techniques used to perform video stabilization. The chapter begins by describing the method of mechanical stabilization and its advantages and disadvantages. The chapter then describes the background and theory behind optical stabilization. Finally the method of digital stabilization is discussed along with a background on feature detection.

## 1.2 Chapter 3 Summary—Digital Video Stabilization with Inertial Fusion

This chapter explains the stabilization procedure developed in this thesis. It begins by describing the data acquisition rig developed to test the procedure; this includes a description of the camera, and a walkthrough of the design and build of the IMU. Then the visual measurement is discussed; this includes the derivation of SIFT features and descriptors, an explanation of the random sample and consensus (RANSAC) process that removes outliers and solves for the affine motion model, and finally the visual measurement strength coefficient is derived.

Section 3.3 explains the inertial measurement. It begins by describing the method of calibrating the accelerometers using the gravity vector and the method of calibrating the gyroscopes with a rotating platform. The section closes with a full description of the IMU design including; the PCB design (microcontroller, IMU chip, communication, and power supply), and the embedded software architecture.

The next part of chapter 3, section 3.4, explains the fusion method that combines the visual and inertial measurements. First the 9 state vector is shown and the dynamic model is derived, then the process noise matrix is evaluated. The Kalman gains are then explained as well as the Ricatti equations. The measurement matrix and measurement noise matrix are then derived. Finally a flowchart is presented that graphically defines the full method of the data fusion process.

Motion filtering is defined, which is the separation of wanted and unwanted motion. A frequency analysis of handheld motion induced in the camera is first performed. It is then shown that the wanted and unwanted motion can be separated by frequency content and a low pass filter is developed to remove the wanted motion.

The chapter is completed with a description of the frame compensation method. This begins by examining the different techniques available to provide frame compensation. The method used for frame compensation in this thesis is then defined, explained and justified.

## 1.3 Chapter 4 Summary—Experimental Results

The experimental results chapter goes through all the steps in the stabilization process and evaluates the error. It begins by examining the errors involved in the visual and inertial measurements. Then the full stabilization process is evaluated to determine its performance. Lastly the timing and computational load of the stabilization process are evaluated.

# 2 Literature Review

Video stabilization removes unwanted motion in the video feed. This is performed by first detecting the induced motion on the camera and then compensating for it, so the output video is void of unwanted shakes and jitters. The literature describes three different types of stabilization; mechanical, optical and digital stabilization. Each method derives the camera motion and performs compensation differently, resulting in them having different performance characteristics.

This literature review first examines the technique behind mechanical stabilization. This is done by describing the physics behind the motion sensors and relating it to the acceleration and angular velocity of the system. Then the motion compensation of mechanical stabilization is discussed, presenting a detailed description of each method.

Optical stabilization is discussed in section 2.2. This presents the theory, discusses the different methods of applications and describes the areas of application.

Digital stabilization is examined in the final section of the literature review. The overall process of digital stabilization is defined and a background of feature extraction is given. The method of global motion detection is presented and several methods of motion filtering are discussed. This section ends with a review of the performance evaluation methods for digital stabilization, describing several techniques used to quantify the performance of a digital stabilizer.

## 2.1 Mechanical Stabilization

Mechanical stabilization is a type of stabilization that mechanically determines the vehicle's motion and mechanically compensates for that motion. The motion is sensed through inertial measurements and compensated with an actuated platform. The motion detection and compensation can have as many as six degrees of freedom or as little as one; this is dependent upon the application.

The sensing is performed with accelerometers and gyroscopes, these transducers convert the physical world into electrical signals which can be interpreted using a microcontroller. The most commonly used accelerometers and gyroscopes are microelectromechanical systems (MEMS) and that will be the type discussed in this literature review as well as implemented throughout this thesis.

### 2.1.1 Inertial Sensors

MEMS accelerometers come in two different flavors, capacitive and piezoelectric sensed. Both sensing methods use the same basic physical principle of a mass suspended by a beam that is anchored to an inertial frame. The accelerometer is modeled as a second order mass spring system where the mass is fixed ($M$) the beam has a spring constant and damping factor ($K$ and $D$ respectively). An external acceleration displaces the frame relative to the mass which in turn creates a strain on the suspension-beam. The acceleration is transformed to an electrical signal by either measuring the displacement (capacitive sense) or the stress on the beam (piezoelectric sense) [2]. The transfer function of the accelerometer is shown in equation 2.1; where $a$ is the external acceleration, $x$ is the mass displacement, $w_r$ is the natural resonance frequency, and $Q$ is the quality factor.

$$H(s) = \frac{x(s)}{a(s)} = \frac{1}{s^2 + \frac{D}{M}s + \frac{K}{M}} = \frac{1}{s^2 + \frac{w_r}{Q}s + w_r^2} \qquad 2.1$$

Piezoelectric sense is performed by incorporating silicon piezoresistive material within the support beam, as the frame moves relative to the mass the beam will elongate or shorten thus changing

the stress profile and resistivity of the embedded piezoresistor [2]. The main advantage of piezoelectric sense is their simple structure and fabrication process. Another advantage is their readout circuitry, a bridge circuit, which offers a low output impedance voltage. Piezoelectric sense has higher temperature sensitivity and lower overall sensitivity as compared to their capacitive counterparts. These transducers typically have a sensitivity of 1-2mV/g in a 20-50g range and an uncompensated temperature coefficient of sensitivity less than 0.2%/◦C [2].

Capacitive sense is utilized from the principle that In the presence of acceleration the mass moves from its rest position, thus changing the capacitance between the mass and a fixed conductive electrode separated by air in a narrow gap [2]. The most widely used capacitive sense accelerometers are vertical (z axis) and lateral (x and y axes). The vertical sense has a mass suspended above the fixed plate electrode separated by a narrow gap; this forms a parallel plate sense capacitance [3]. Lateral accelerometers have a number of capacitive combs attached to the mass that are sensed by fixed combs parallel to the combs on the mass [4]. These transducers have high sensitivity, good dc response and noise performance, low drift, low temperature sensitivity, and low power dissipation [2]. The structure of the vertical and lateral capacitive sense accelerometers are shown in figure 1.

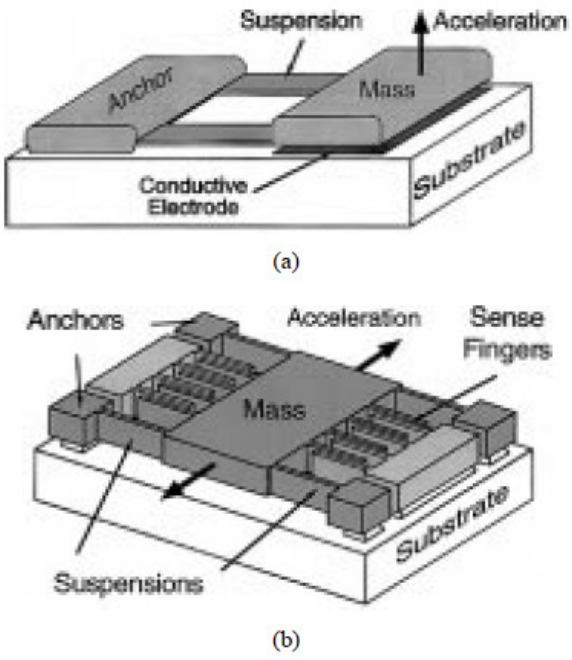


Figure 1: Vertical capacitive sense (a), lateral capacitive sense (b) [2]

MEMS gyroscopes measure the rate of rotation about a given axis. All of these transducers take advantage of the Coriolis force to convert this physical property into an electrically readable signal. They are fabricated as a proof mass that has 2 perpendicular degrees of freedom, of which are orthogonal to the desired axis of rotation to be measured. The mass is held by springs that have damping constants $c_x$ and $c_y$ and stiffness constants $k_x$ and $k_y$, as shown in figure 2.

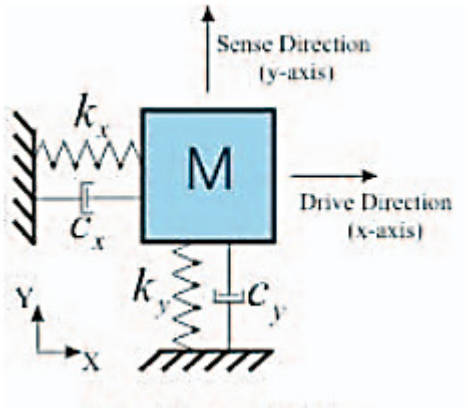


Figure 2: Mass spring model of MEMS gyroscope [5]

The mass is electrostatically driven along one axis and capacitive sensed along the other, the driven force is oscillatory and described by equation 2.2

$$x(t) = X_D \sin(\omega_n t) \tag{2.2}$$

The equations of motion for the mass are described in equation 2.3.

$$\begin{aligned} m\ddot{x} &= -c_x\dot{x} - k_x x + m\left(\Omega_z^2 x + 2\Omega_z\dot{y} + \dot{\Omega}_z y\right) + F_x \\ m\ddot{y} &= -c_y\dot{y} - k_y y + m\left(\Omega_z^2 y - 2\Omega_z\dot{y} + \dot{\Omega}_z x\right) + F_y \end{aligned} \tag{2.3}$$

These gyroscopes are designed so that the two natural frequencies and damping ratios are ideally identical ($\omega_x = \omega_y = \omega_n,\ and\ \zeta_x = \zeta_y = \zeta$) [5], exploiting this principle and the fact that the natural frequencies are much larger than the rotation rates measured from the gyroscope ($\omega_n \gg \Omega_z$) the equations of motion can be simplified to.

$$\begin{aligned} \ddot{x} + \frac{\omega_n}{Q} + \omega_n^2 x - 2\Omega_z\dot{y} &= \frac{F_x}{m} \\ \ddot{y} + \frac{\omega_n}{Q} + \omega_n^2 y + 2\Omega_z\dot{x} &= \frac{F_y}{m} \end{aligned} \tag{2.4}$$

Where Q is the quality factor, which is the ratio of energy stored per oscillation to energy lost ($Q = \frac{1}{2\zeta}$) [5].

The rotational rate can be derived by measuring the Coriolis force induced upon the sensing axis, this is shown by substituting the constant drive force, equation 2.2, into the y axis motion dynamics described in equation 2.4.

$$|y(t)| = \frac{2X_D\Omega Q}{\omega_n} \tag{2.5}$$

This result shows the output sense deflection is proportional to the input angular rate. This deflection is capacitive sensed by a sense electrode that converts the observed capacitance into a distance between the electrode and the proof mass, as shown in figure 3.

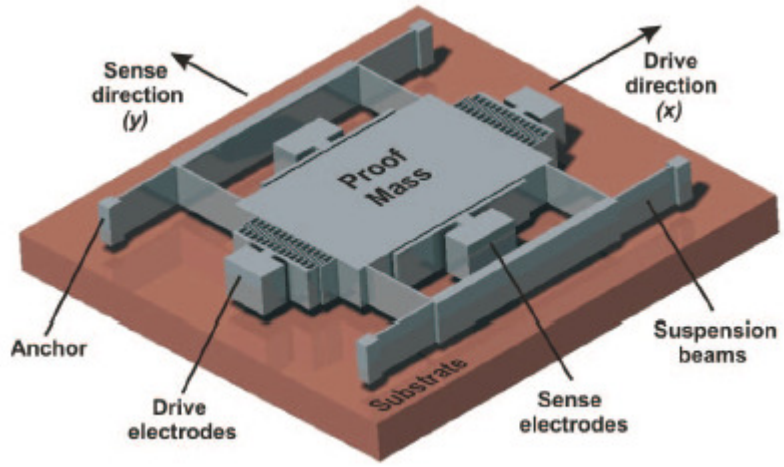


Figure 3: Physical model of a MEMS gyroscope [5]

For both the MEMS accelerometer and gyroscope the theory of operation has been described. It is important to note that the description has been how to convert the physical phenomena of linear acceleration and angular rate into readable electrical signals. In order to retrieve those signals and change them into meaningful data, further steps must be taken. These steps include analog to digital conversion and signal processing. Analog to digital conversion is generally performed at the chip level. Although some high end inertial sensors do provide analog outputs, the small and inexpensive MEMS devices generally only support a digital output. Extensive signal processing must also be performed to filter the data into something meaningful, this process is discussed in sections 3.3.1 and 3.3.2.

### 2.1.2 Mechanical Motion Compensation

The camera motion is determined by double integrating the vehicle's acceleration ($a$) to provide displacement ($x$) and integrating the vehicle's angular rate ($w$) from the gyroscopes to determine angular position ($\theta$).

$$x(t) = \iint a(t)dt$$
$$\theta(t) = \int w(t)\,dt \qquad 2.6$$

With mechanical stabilization the desire is to keep the camera stable with respect to the reference frame. This means that all measured motion is viewed as unwanted motion (shake and jitter) on the camera. The stabilization goal is to remove the unwanted motion and leave the wanted motion (pan, zoom…) present.

The motion is compensated through mechanical actuation of a platform. The platform is controlled as a closed loop, negative feedback system whose control inputs are from the inertial sensors and the camera can either be mounted on or off the actuated platform. When the camera is mounted

to the platform the complete system is stabilized (camera, inertial sensors and platform). If the camera is mounted off the actuated platform then its position is fixed and only the inertial sensors and platform are stabilized, this is either done by actuating a mirror to reflect the scene onto the camera's CCD or by actuating the CCD itself.

Mechanical stabilization is a specialized and expensive method for video stabilization. It requires accurate and robust control of the actuators and advanced signal processing on the sensors. The image is stabilized before it reaches the CCD which makes this method able to perform stabilization without loss of any information, which contrasts against digital stabilization. Furthermore, mechanical stabilization has a wider dynamic range than the other stabilization methods. The dynamic range describes the amount of jitter than can be compensated by the system. Current techniques for mechanical stabilization can compensate up to 50 pixels jitter, while the other stabilization methods can compensate up to 20 pixels jitter with the same performance criterion [6].

## 2.2 Optical Stabilization

Optical stabilization is performed similarly to mechanical stabilization in which the motion of the camera is detected through inertial sensors. The method varies from mechanical stabilization because the motion is compensated by altering the optical path that the light travels to reach the imaging sensor. This method is usually seen implemented in handheld cameras, Panasonic offers optical stabilization in their Lumix digital cameras [7] and Canon Incorporated installs optical stabilization in many of their lenses [8].

The unwanted motion is determined using inertial sensors, mainly pitch, roll and yaw gyroscopes due to the fact that rotational motion causes a larger disturbance in the image plane in comparison to linear motion when viewing objects that are reasonably far away. Rotational disturbances have more effect based upon the principle of parallax. This principle is graphically shown in figure 4, where the effect of linear and rotational motion in a 2 dimensional plane is described, the $(x, y)$ axes describe the reference before the motion and the $(x', y')$ axes are the reference after the motion.

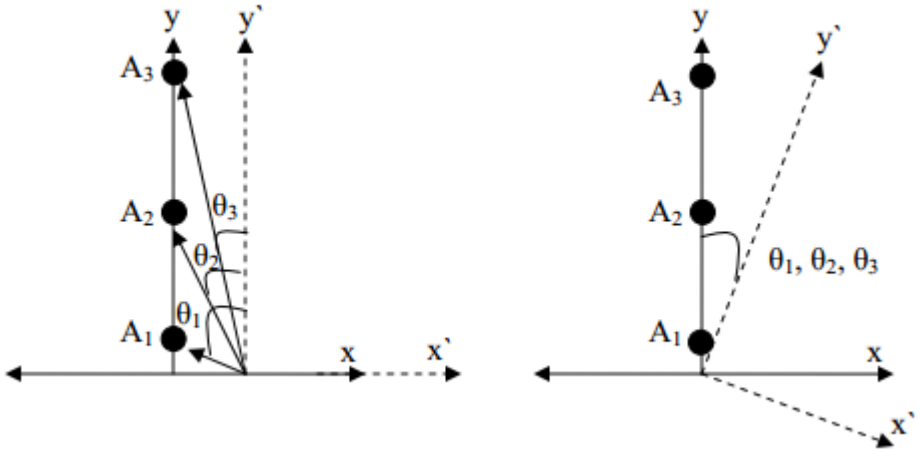


Figure 4: Impact of linear motion and rotational motion on the image plane, respectively [9]

Figure 4 shows that when a camera moves linearly the objects $A_1, A_2$, and $A_3$ result at different angles, but when the camera is rotated, the objects result at the same angle ($\theta_1, \theta_2$ and $\theta_3$ are all equal during the rotational motion). When the objects are farther away the impact of linear motion decreases, but the impact of rotational motion remains the same. For example, if the camera were rotated 1° then the objects in the image plane would be displaced 1° and for an object 10 meters away, it would require the camera to move 1.75cm in order to achieve the same displacement. This is an unlikely motion for someone taking a picture, while a 1° rotation fits well within reason.

Once camera motion is determined it can be filtered to separate wanted and unwanted motion. This is not always necessary and is dependent upon the application, for instance, when taking snapshots with a camera all motion can be perceived as unwanted, but when taking video, pan and zoom are considered to be wanted motion and should be left alone.

Optical stabilization compensates for unwanted shake and jitter by altering the path light travels to reach the imaging sensor. This is accomplished in one of two ways, either moving a set of lenses to alter the light path or actuating a fluid prism to change the refraction angle of the light passing through it. The most common method is actuating a set of lenses to alter the light path and is the technique implemented by Canon and Panasonic in [7] [8]. The fundamental theory behind using multiple lenses is shown in figure 5.

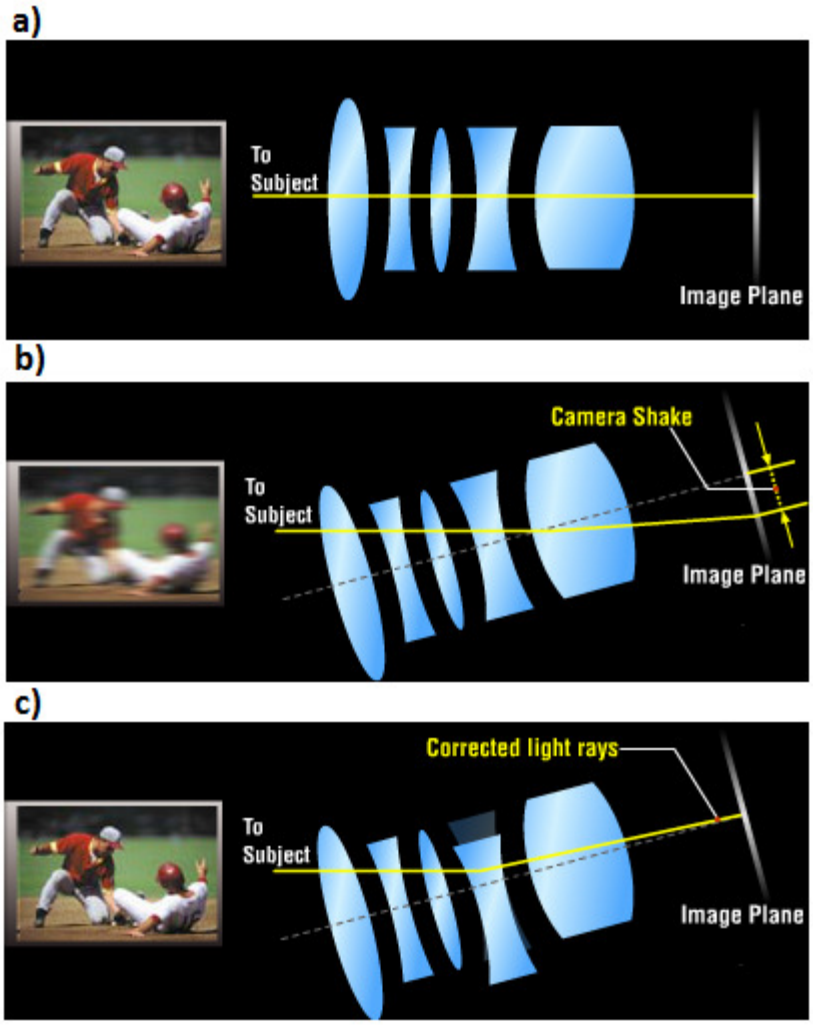


Figure 5: Optical stabilization using lens shift:

a) No jitter involved

b) Jitter involved with no compensation

c) Jitter involved with compensation. [8]

Each lens moves along a specific axis, when compensation is required, the lens is actuated so that the path traveled for the light is altered and it reaches the image plane at the correct location.

### 2.2.1 Fluid Prism Optical Stabilization

Sony Corporation implements a fluid prism in their video camcorders in order to achieve stabilization, the prism is made of a pair of glass plates that are connected with flexible bellows. The prism is filled with transparent liquid and both glass plate are independently rotated vertically and horizontally by actuators [10].

The motion detection is performed with a pair of pitch and yaw gyroscopes each sampled at a 1Khz rate and quantized with a 10 bit analog to digital converter [10]. Motion filtering is performed by

comparing the difference of the gyroscopes amplitude and frequency, then calculates the motion vector using nonlinear integration [10]. The result is then low pass filtered and sent to the control module for prism actuation. The optical stabilization system and principle of operation are shown in figure 6 and figure 7 respectively.

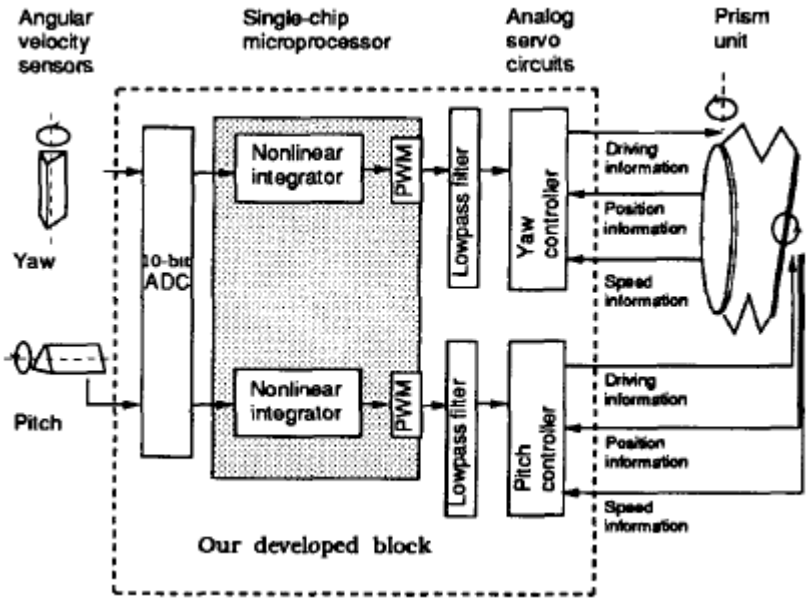


Figure 6: Sony corporation's optical image stabilizer [10]

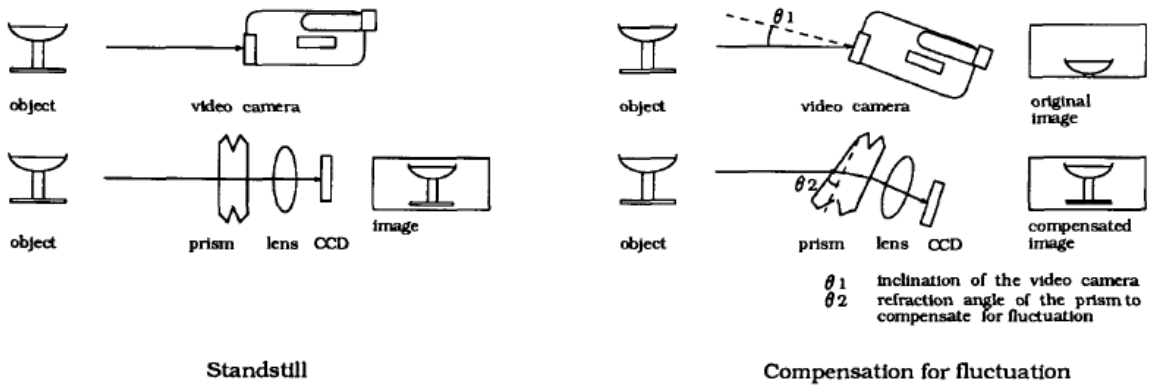


Figure 7: Principle of operation [10]

## 2.3 Digital Stabilization

Digital stabilization is the process of digitally removing unwanted motion in a video feed and creating a smooth video output. This technique post-processes the video to determine how the camera is moving and what part of that motion is unwanted. The image plane is then moved in the direction opposite of the unwanted motion to provide a video feed void of shake and jitter. The advantage digital stabilization holds over mechanical or optical stabilization is that it requires no additional hardware and can be performed after the video is taken, making it possible to stabilize video that has been archived.


Figure 8: Digital stabilization block diagram

Digital video stabilization (DVS) is generally discussed in three main sections: global motion detection, motion filtering and motion compensation. Global motion detection is the process of finding matching features between frames and forming a model that describes the motion in the image plane. Most DVS systems accomplish this using optical flow or feature tracking techniques, although some find global motion using phase correlation. Motion filtering separates the wanted motion (zoom, pan…) from unwanted motion (shake, jitter…) and outputs a model representing the unwanted motion. Most DVS systems separate the motion using recursive Kalman filters, while others separate using frequency analysis by exploiting the fact that wanted motion has smaller frequencies with high energy content and unwanted motion is at higher frequencies with smaller energy content. The process is completed with frame compensation, which moves the image plane opposite the direction of the unwanted motion model. This is commonly achieved using only translational compensation, but more advanced methods do apply an affine transformation which incorporates translation (2 dimensional), rotation and scaling.

### 2.3.1 Motion Estimation

The first step in digital video stabilization is to determine how the camera is moving, this is called motion estimation. Camera motion is estimated by viewing sequential frames in a video feed and deriving local motion vectors (LMV), these local motion vectors are then combined to form a global motion vector (GMV) that represents the motion of the camera.

### 2.3.2 Feature Extraction and Matching

Local motion can be derived by finding features in subsequent image frames and then matching them to determine their start and end coordinates $\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \end{bmatrix}$. There are many methods to find features in an image, the most common being the scale invariant feature transform (SIFT) [11, 12]. Other methods include corner detection [13, 14], Shi-Tomasi good features [15], and the speeded up robust

features (SURF)[16]. Scale invariant feature transform (SIFT) is widely regarded in the computer vision community as one of the most accurate and robust types of features [12] and thus the most common feature extraction technique seen in digital stabilization. This section continues by discussing the different methods of feature extraction.

The method of feature extraction began with the work performed by Harris and Stephens in [17]. Harris et al. proposed that image features must be discrete and not formed on a continuing texture or line; thus arises the need for choosing corners in the image to be the feature points. The theory behind the algorithm is to exploit the autocorrelation function of a sub-window in the image. If the window is moved slightly, the correlation with its previous position will either change slightly (when in a flat or textured region) or steeply (when viewing a corner or edge). The math behind the process is as follows and begins with the autocorrelation function:

$$c(x,y;\Delta x,\Delta y)=\sum_{(u,v)\in W(x,y)} w(u,v)\big(I(u,v)-I(u+\Delta x,v+\Delta y)\big)^2 \qquad 2.7$$

Where $W(x,y)$ is the window centered at point $(x,y)$ and $w(u,v)=\frac{e^{-(u-x)^2-(v-y)^2}}{2\sigma^2}$, a Gaussian weighted coefficient. The approximation of the shifted function is performed with Taylor expansion shown in equation 2.8

$$I(u+\Delta x,v+\Delta y)=I(u,v)+[I_x(u,v),I_v(u,v)]\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} \qquad 2.8$$

where $I_x$ and $I_y$ are partial derivatives of $I(x,y)$. Putting equation 2.7 into 2.8 yields:

$$c(x,y;\Delta x,\Delta y)=[\Delta x,\Delta y]Q(x,y)\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} \qquad 2.9$$

$$Q(x,y)=\begin{bmatrix}\sum_W I_x(x,y)^2 & \sum_W I_x(x,y)I_y(x,y)\\ \sum_W I_y(x,y)I_x(x,y) & \sum_W I_y(x,y)^2\end{bmatrix} \qquad 2.10$$

The two eigenvalues of Q are then computed, representing two perpendicular directions of greatest change. These eigenvalues ($\lambda_1$,$\lambda_2$) can then form one of three situations. If they are both small the point lies in a flat region, if one is large and one is small the point lies on an edge, and if both values are large the point lies on a corner. We select only corners as feature points, so when both eigenvalues are large we choose it to be a feature point. Harris et al. perform this check efficiently by thresholding the corner response ($H$) as shown in equation 2.11.

$$H(x,y)=\det(Q(x,y)-k\left(trace\big(Q(x,y)\big)\right)^2 \qquad 2.11$$

Corner detection proves to be the fastest method of feature extraction in regards to computational time but is disadvantaged against other, slower, methods because it does not create a distinct descriptor for each feature point. Having a descriptor is a valuable tool in feature tracking because it allows efficient matching of features in consecutive video frames. The method to match features without the use of a descriptor has traditionally been a correlation approach derived from the method proposed by Lucas and Kanade in [18] which describes a method for registering two images for stereo vision matching.

Lucas et al. begin the solution for tracking features in images by exploiting the fact that because a video feed has up to 30 images taken per second, it can be assumed that there is a slight difference in feature position from image taken at time $t$ and one taken at time $t + 1$. With that in mind, we will define two sets of features taken from sequential images $I_1(x, y)$ and $I_2(x, y)$ and allow the motion between matching features to be purely translational $(u, v)$. The translation can be solved by minimizing the error in equation 2.12.

$$error(u, v) = \sum_{(x,y)\in R} \big(I_2(x + u, y + v) - I_1(x, y)\big)^2 \qquad 2.12$$

The region, $R$, is the window size around the feature that the summation is performed, once convergence is achieved and translation is solved, this window becomes a local motion vector (LMV) that is fed into a model whose solution defines the global motion vector (GMV). The error is minimized through an iterative process of solving a system of equations $A[u, v]^T = b$ where the matrix $A$ and $b$ are defined as:

$$A = \sum_{(x,y)\in R} \nabla I_2(x, y) \nabla I_2(x, y)^T \qquad 2.13$$

$$b = -\sum_{(x,y)\in R} \nabla I_2(x, y)\big(I_2(x, y) - I_1(x, y)\big) \qquad 2.14$$

Beginning with an estimate of translation $(u, v)$ the features in $I_1$ are translated towards $I_2$ then the translated features are used to compute $b$ in the next iteration until convergence is reached. It is only necessary to calculate matrix $A$ once, however matrix $b$ varies with each iteration according to equation 2.15.

$$b^{i+1} = -\sum_{(x,y)\in R} \nabla I_2(x, y) I_2(x, y) + \sum_{(x,y)\in R} \nabla I_2(x, y) I_1^i(x, y) \qquad 2.15$$

This method is improved upon and used for image stabilization by Keng-Yen et al. in [13] where they remove the need for iteration by following a technique proposed by Rav-Acha and Peleg in [19] which convolves the image with an interpolation kernel to provide the translation.

$$I_1^i = I_1 * m^i \tag{2.16}$$

Following this relationship, the right half of equation 2.15 becomes:

$$\begin{aligned}\sum_{(x,y)\epsilon R} \nabla I_2(x,y) I_1^i(x,y) &= \sum_{(x,y)\epsilon R} \nabla I_2(x,y)\big(I_1(x,y) * m^i\big) \\ &= \sum_{(k,l)} m^i(k,l) \sum_{(x,y)\epsilon R} \nabla I_2(x,y)\, I_1(x-l, y-k)\end{aligned} \tag{2.17}$$

Now equation 2.15 can be rewritten as:

$$b^{i+1} = r + \sum_{k,l} m^i(k,l) s(k.l) \tag{2.18}$$

$$r = -\sum_{(x,y)\epsilon R} \nabla I_2(x,y) I_2(x,y) \tag{2.19}$$

$$s(k,l) = \sum_{(x,y)\epsilon R} \nabla I_2(x,y) I_1(x-k, y-l) \tag{2.20}$$

The sum runs over all locations $(k,l)$ of the interpolation kernel and the functions $r$ and $s(k,l)$ are 2x1 vectors that remain constant through the iterations, thus only needing a single calculation. In conclusion, this method converges faster and only requires about the same number of operations as a single iteration of the Lucas Kanade method [18] but still provides equivalent accuracy [19].

Another method of feature tracking proposed by Censi in [14] uses Shi-Tomasi good features [20] and a Kalman filter to predict the features trajectory in sequential frames. Shi-Tomasi defines good features as two large eigenvalues of the matrix (Q) in equation 2.10 that satisfy:

$$\min(\lambda_1, \lambda_2) > \lambda \tag{2.21}$$

The threshold $\lambda$ is predefined and is specific to the application. First a lower bound for the threshold is determined by measuring the eigenvalues for images of approximately uniform brightness, and the upper bound is found by measuring the eigenvalues of detected features (corners). The threshold is set as the middle point between the two bounds. Feature tracking is performed by searching in a region where the feature should be as predicted from a Kalman filter. This is done by

considering the frame sequence $f_0, f_1 \dots f_k$ acquired by a camera with frame rate of $\frac{1}{\Delta t}$, the state vector is defined as:

$$X_k = [x_k \; y_k \; u_k \; v_k]^T \tag{2.22}$$

Where $(x_k, y_k)$ are the feature's location and $(u_k, v_k)$ are the feature's velocity in frame $k$. The state transition matrix and measurement matrix are defined as:

$$\Phi = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.23}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.24}$$

The noise covariance matrix $(Q)$ models the prediction uncertainty and the matrix $(R)$ models the measurement uncertainty. The matrix $(Q)$ depends on the local image derivatives and the higher a derivative in one direction means the feature's motion in that direction is more reliably predicted. The matrix $(R)$ depends on the correlation of matching feature windows, then a higher correlation leads to higher confidence in displacement measurement. Censi et al. choose the coefficients of $(Q)$ and $(R)$ specific to the camera through a trial and test process [14].

The state covariance matrix $(P_k)$ is updated dynamically and encodes the uncertainty of the current state; the region around the estimated state $\hat{X}$ that contains the true state with given probability $c^2$ is defined by the ellipsoid in equation 2.25.

$$\left(X - \hat{X}\right)(P_k)^{-1}\left(X - \hat{X}\right)^T \leq c^2 \tag{2.25}$$

The matching is performed by looking for the minimum of SSD in a small search window centered at the predicted state of the Kalman filter where the search window is an ellipse defined in equation 2.25. If the match fails then the search is again performed in the position of the previous frame, if that match fails then the feature goes into a temporary state where it will be searched for again in the next 3 frames and finally discarded if no match is found [14].

### 2.3.3 Motion Filtering

Digital stabilization is usually performed on video; because of this it is important to segregate the unwanted and wanted motion. This is commonly performed on the principle that the unwanted shake and jitter induced in video sequences have a larger frequency and smaller energy content than intentional panning and zoom.

Ioannidis et al. solve this problem using the Hilbert-Huang transform [21] to sort the LMVs into ascending order of frequency and energy content, then threshold them to separate intentional from unintentional motion. This is effective because it is implemented without predetermined information making the segregation variable and only dependent upon the information in the scene [22]. Jin et al. separate wanted and unwanted motion by building LMVs in a 2.5 dimensional state where the half dimension is defined as a decision on what type of motion is being induced, depending on that decision, a specific inertial filter is implemented on the LMVs to remove the wanted motion and leave only unwanted motion available for frame compensation [23].

SRI International manufacture a standalone digital image stabilizer that implements a model based low pass filtering approach that performs well when a precise model of the camera carrier can be constructed [24]. They use a six parameter affine motion model built by the LMVs and is defined in equation 2.26,

$$M_{t_{-1}\to t} = \begin{bmatrix} \alpha\cos(\theta) & -\alpha\sin(\theta) & x \\ \alpha\sin(\theta) & \alpha\cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \quad 2.26$$

where $\alpha, \theta, x$, and $y$ are scale, roll angle, x translation and y translation respectively. The model is derived from the LMVs through linear regression and follows suit to equation 2.27,

$$p(x', y') = M_{t_{-1}\to t} p(x, y) \quad 2.27$$

where $p(x, y)$ and $p(x', y')$ are matching features from consecutive images.

After the initial model is formed it is smoothed using an IIR filter and then low pass filtered with an eight state Kalman filter that contains the model parameters and their derivatives. The Kalman filter employs the constant velocity model to estimate the state vector [24]. This technique was used to minimize the black border regions that occur when frame compensation creates regions in the output image that contain no real data.

Another type of motion that needs to be filtered from the LMVs is moving objects in the scene. LMVs are determined by tracking features in the image plane and when there are non-stationary objects in the scene their motion is considered a LMV, this does not represent the motion of the camera and needs to be accounted for. This is generally performed by viewing the background of the scene under the assumption that most moving objects are present in the foreground. Other researchers have solved this problem by fusing data from inertial sensors located on the camera to filter out motion in the scene that is not specific to the motion of the camera [15].

### 2.3.4 Performance Evaluation

Digital stabilization can easily be evaluated qualitatively, by viewing the un-stabilized and stabilized videos, an apparent reduction in shake and jitter can be seen. This qualitative approach allows the designer to know the algorithm is working properly but does not give quantitate information

of the stabilizer's performance. When two stabilization algorithms provide near similar stabilized feeds, a qualitative examination cannot accurately discriminate the performance of the two algorithms. This leads to the need for a quantitative examination of a stabilizers performance.

There are many ways to evaluate the performance of a digital stabilizer. Balakirsky et al. in [25] compare the performance of multiple stabilization algorithms based on deviations of tracked objects in the scene. The Army Research Laboratory (ARL) tracking algorithm is a pipelined algorithm that uses the difference of images to classify targets [26]. The stabilization performance is defined as the mean squared error (MSE) of the tracked object position between stabilized frames.

Morimoto et al. use fidelity and the displacement range to provide a performance measure [27]. Fidelity has two components, the inter-frame fidelity (ITF) and the global frame fidelity (GTF). ITF is defined as the peak signal to noise ratio (PSNR) between consecutive frames in the video feed and GTF is defined as the PSNR of the current and reference frame in the video feed. For both methods the PSNR is calculated according to equation 2.28, where MSE is the mean squared error. A successful stabilization algorithm will provide stabilized video that has a lower ITF and GTF than the original un-stabilized video.

$$PSNR = 10\log\left(\frac{255^2}{MSE(I_1, I_0)}\right) \tag{2.28}$$

Morimoto et al. define displacement range as the maximum amount of image displacement the stabilizer can handle. First, stabilization is performed on synthetic video feeds that contain different types of known displacements. Then the stabilized and un-stabilized feed is evaluated for their ITF and GTF components. As displacement increases the $ITF_{unstab}$ decreases and the $ITF_{stab}$ increases, their first crossing point is considered the minimum image displacement. The maximum displacement boundary is determined through synthetic video of large displacements. As the displacements become large the system provides invalid motion estimates causing the $ITF_{stab}$ to be less than $ITF_{unstab}$, this crossing point is defined as the maximum image displacement. Performance is defined as the maximum displacement velocity supported by the stabilizer which is the product of the maximum displacement and the frame rate, in pixels per second.

Zhang et al. evaluate performance by introducing synthetic video of known motion into the stabilizer [28]. The derived motion from the stabilizer is then evaluated with the known motion in the synthetic video through RMS analysis to declare a performance measure. The MSE derivation and motion model are shown in equation 2.29, where $(x_c, y_c)$ is the image center.

$$\begin{aligned} MSE(t) = {} & \left(d_{13} - \hat{d}_{13}\right)^2 + \left(d_{23} - \hat{d}_{23}\right)^2 + \left(d_{11} - \hat{d}_{11}\right)^2 x_c^2 + \left(d_{12} - \hat{d}_{12}\right)^2 y_c^2 \\ & + \left(d_{21} - \hat{d}_{21}\right)^2 x_c^2 + \left(d_{22} - \hat{d}_{22}\right)^2 y_c^2 \end{aligned}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.29}$$

The MSE is a single frame-to-frame alignment measure. RMS error is the average of the square root of MSE over the entire video sequence, if multiple sequences are used then RMS incorporates them as well. This is shown in equation 2.30, where $(N_r, N_t)$ is the total number of sequences and frames count for sequence respectively.

$$RMS = \frac{1}{N_r N_t} \sum_{r,t} \sqrt{MSE(r,t)} \qquad 2.30$$

Smith et al. also use the RMS value derived by equations 2.29 and 2.30, except the evaluation is not performed on synthetic video. They use real video taken from a camera mounted on a quadruped robot. The ground truth motion is found by visually mapping the motion with a frame-to-frame comparison, although this is time consuming, it provides a ±3 pixel ground truth to a video sequence of unknown motion. The MSE is calculated for the image translation and rotation according to equation 2.31 and the RMS calculation is the same as equation 2.30 [15].

$$\begin{aligned} MSE_{dx} &= \left(d_{13} - \hat{d}_{13}\right)^2 \\ MSE_{dy} &= \left(d_{23} - \hat{d}_{23}\right)^2 \\ MSE_{\theta} &= \left(\arctan\left(\frac{d_{31}}{d_{11}}\right) - \arctan\left(\frac{\hat{d}_{31}}{\hat{d}_{11}}\right)\right)^2 \end{aligned} \qquad 2.31$$

# 3 Digital Image Stabilization with Inertial Fusion

Through analysis of the literature review I have determined that image stabilization onboard mobile ground robotic systems can best be solved with the digital stabilization method. This conclusion was met with respect to two main considerations, cost and flexibility.

A large limiting factor of vehicular robots is the cost they impose, in order for them to be commercially and economically viable; the sensor, actuation and control suite must cost relative to that of the base vehicle. With this in mind there is little money left for image stabilization after purchasing the costly and necessary components required for automation. These components include; high fidelity GPS fused inertial navigation systems, robust and ruggedized computer systems for control, large field of view range finder (whether it be LIDAR, RADAR or SONAR), long range wireless communication system, and any other bells and whistles the project specifications may require. This leads to the choice of digital stabilization because it has a lower cost than its mechanical or optical counterparts.

When designing vehicular robots, it is common to convert a pre-existing platform into an autonomous system. This is different from designing a robot from scratch where the platform is built with consideration to sensor, actuator and computer mounting. This caveat leads to a certain flexibility required in the sensor suite of vehicular robots; they must be able to conform to the platform, instead

of the platform conforming to them. Digital stabilization provides this flexibility in design because it can be used with any camera mounted in any position. Mechanical stabilization requires a specifically mounted platform for stabilization and optical stabilization needs a specific camera. Another added flexibility from using digital stabilization is the fact that any previously designed visual sensor suite can be digitally stabilized as long as there is the computing power available.

Digital stabilization's drawbacks are its requirement for computing power and its instability with dynamic scenes. This thesis aims to alleviate those problems with the fusion of visual and inertial measurements. These measurements complement each other in their strengths and weaknesses. The visual measurement is slow and susceptible to dynamic scenes but the inertial measurement is fast and immune to dynamic scenes; also the visual measurement is temporally stable while the inertial measurement accrues error over time. Incorporating the inertial measurement allows the digital stabilization to run at a slower rate, which decreases the demand for computing power.

As was discussed in chapter 1, this chapter provides the theory and execution of the digital image stabilizer using inertial fusion. The flow of this chapter is designed to define the full system from a bottom up approach. First the acquisition rig is discussed and an overall system block diagram is presented. Then the visual measurement is explained by deriving SIFT features and descriptors, RANSAC outlier removal, the affine motion model, and the measurement strength coefficient. The inertial measurement is explained by describing the design and build of the IMU and the calibration process. Finally the fusion process is explained by defining the states and dynamic model of the Kalman filter, deriving the process and measurement noise, and presenting a flowchart that describes the full process of the fusion method.

## 3.1 Data Acquisition Rig

The theory and algorithm of the visual/inertial fused stabilization is tested on a sensor rig consisting of a camera and IMU shown in figure 9.

Figure 9: Sensor rig consisting of IMU and camera

The camera used is the Firefly MV provided by Point Grey Research Incorporated [29]. It is a monochrome CMOS imaging system capable of being externally triggered, it has a global shutter and interfaces to a computer through USB. This camera was chosen because of its low cost ($200.00) and external triggering capabilities, which is necessary to temporally fuse the IMU and the camera. It has a

1/3” CCD CMOS sensor and is incorporated with a lens that has a 4mm focal length. This lens was chosen because it provides a common field of view, 62° horizontal and 48.5° vertical.

Figure 10: Firefly MV camera [29]

I designed and built the IMU specifically for this thesis. Although off the shelf IMUs can be used for this purpose it was my intent to keep the system flexible and low cost. The IMU is designed to provide a variable rate external trigger for the camera and a variable sample rate so that multiple triggering and sampling rates could be evaluated. The IMU is shown in figure 21, and is fully discussed in section 3.3.3. The total cost for the IMU is $80.00 which is considerably lower than commercially available products that provide similar functionality.

The data is acquired and post processed to perform stabilization. The purpose of this thesis is not to have a functioning system in real time but to develop a robust foundation that can be ported to real time capabilities given adequate resources. The complete process is shown in figure 11.

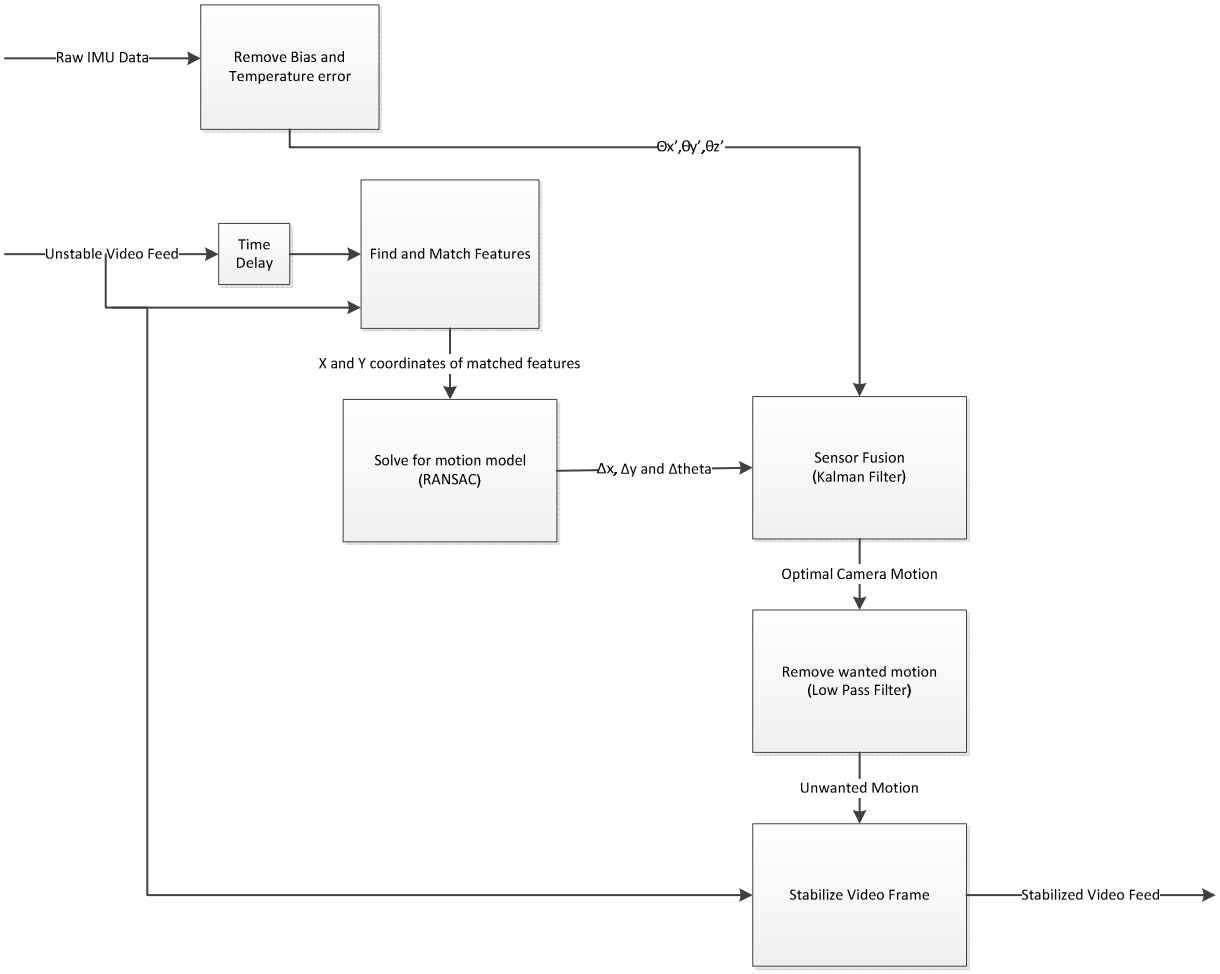


Figure 11: Complete process of stabilizer

## 3.2 Visual Measurement

The first step to visually determine the camera's motion is to find key features in subsequent video frames and match them. This is performed using the highly robust scale invariant feature transform (SIFT) implemented in Matlab using the VLFeat library provided by A. Vedaldi and B. Fulkerson [30]. SIFT is regarded as the most accurate and robust type of feature detection because it is invariant to image scaling and rotation. The features are well localized in both the spatial and frequency domains, thus reducing the probability of disruption by occlusion, clutter, or noise. The algorithm is efficiently able to extract a large numbers of features from typical images. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features [31]. The method for implementing SIFT is described in figure 12.

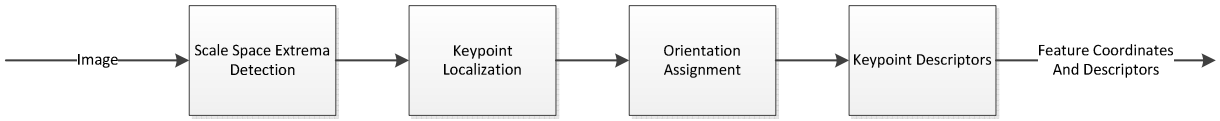


Figure 12: SIFT block diagram

### 3.2.1.1 Scale Space Extrema Detection

The first stage of SIFT is to search through all scales and image locations efficiently using a difference-of-Gaussian function to identify potential interest points. This helps minimize computational cost by efficiently finding regions of interest that can later be evaluated with more expensive operations. The method begins by first building octaves of the image and iteratively convolving each octave with a Gaussian kernel. The Gaussian kernel causes the image to blur which effectively removes detail in the image and is mathematically proven, under a variety of reasonable assumptions, to be the only possible scale space kernel[31] [32]. The amount of octaves and scales depend on the size of the image, the available computing power, and feature density output. However, research performed by Lowe in [31] shows that the algorithm is most efficient using 4 octaves with 5 scales.

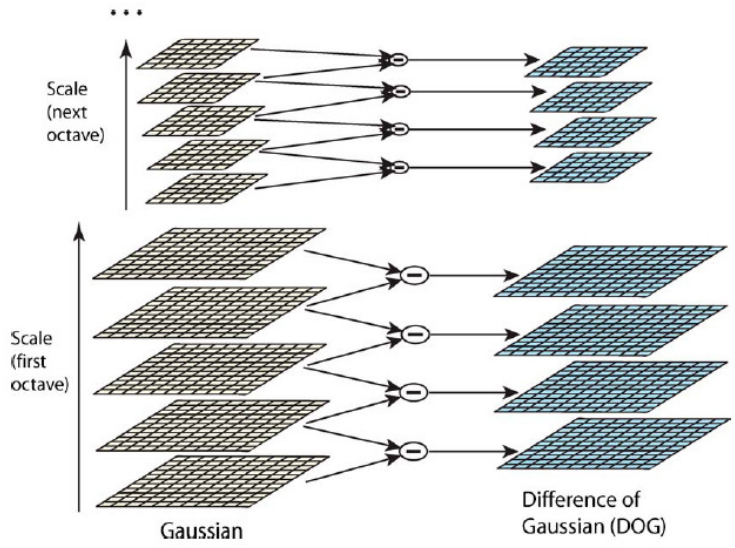


**Figure 13: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images [31]**

The difference of Gaussian images (DoG) are compared within their octave to locate local Extrema. This process is computationally intensive because one must sweep through all pixels in each DoG to find local Extrema. The computation can be reduced by only using the middle DoG images and using the pixels 8 neighbors in the current image and nine neighbors in the scale above and below, the current pixel is then chosen to be a local Extrema only if it is larger or smaller than all of those neighbors. This is particularly effective because most pixels are eliminated following the first few checks [31].

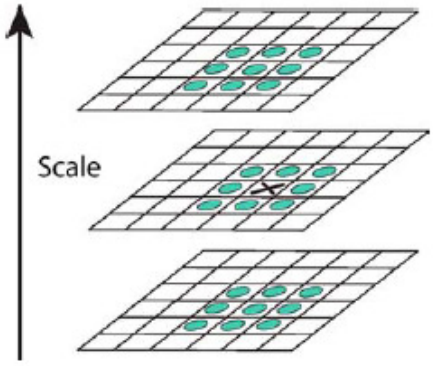


Figure 14: Pixel marked with X is compared with its 26 neighbors to determine local Extrema [31]

#### 3.2.1.2 Keypoint Localization

Now that a crude estimate of feature locations is found we can improve the accuracy by forming a detailed fit to the nearby data for location, scale and ratio of principle curvatures. This information allows points to be rejected that have low contrast or are poorly localized along an edge [31].

The keypoints found using DoG are approximate in that they are rarely the location of a pixel, usually their position lies between pixels. Thus arises the need to interpolate subpixel coordinates using a second order Taylor series expansion.

$$D(x) = D + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2} \quad 3.1$$

Where $D(x, y, \sigma)$ is the DoG image at scale $\sigma$ shifted so the origin is at the sample point and $x(x, y, \sigma)^T$ is the offset from this point. By evaluating the derivative of the above equation at zero we can solve for the subpixel location. This is achieved efficiently using the Hessian matrix.

$$\hat{x} = \frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x} \quad 3.2$$

Consequently by combining the above two equations we can determine the function value at the extrema:

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x} \quad 3.3$$

This Function describes the contrast of the pixel and areas of low contrast can represent unstable extrema. Therefore by thresholding this value unstable extrema can be rejected. Lowe suggests in [31] to use a threshold value of 0.03 when pixel values are normalized in the range [0 1].

The keypoints are then filtered to remove those located on edges or in flat regions. This is accomplished by computing the gradient in two perpendicular directions; if one is large and the other small, the keypoint is located on an edge, if both are small, the keypoint is located in a flat region and if both are large the keypoint is located on an edge. This is computed using a Hessian matrix:

$$Hessian(H) = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad 3.4$$

By analyzing the ratio of eigenvalues in the Hessian matrix a corner location can be identified efficiently. Let $\alpha$ be the eigenvalue with the largest magnitude and $\beta$ be the smaller one, and then the sum of the eigenvalues and their product can be computed by the trace of $H$ and its determinant respectively:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \quad 3.5$$

$$Det(H) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta \quad 3.6$$

Let $r$ be the ratio of eigenvalue magnitude so that $r = \frac{\alpha}{\beta}$ and $\alpha = r\beta$, plugging this into the above two equations we obtain:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \quad 3.7$$

This equation solely depends on the ratio of eigenvalues and is at a minimum when the eigenvalues are equal and it increases with $r$. Therefore by thresholding this value flat and edge regions can be discarded to leave only the corner regions which will yield stronger keypoints. Lowe describes in [31] that this is very efficient to compute, with less than 20 floating point operations per keypoint, he suggest thresholding with a value of $r = 10$ which eliminates keypoints that have a ratio of principle curvatures greater than 10.

#### *3.2.1.3 Orientation Assignment*

Now we have a set of robust scale invariant keypoints, but they are not rotationally invariant. Rotational invariance is achieved by finding the orientation of the keypoint related by the gradient magnitude and direction. The gradient is computed on the Gaussian smoothed image ($L$) of the scale closest to that of the keypoint. The window size of which the gradient is computed around the keypoint is also a function of the keypoint's scale, it is 1.5 times larger than the scale; so as the scale increases, so

does the window size. Equation 1.7 and 1.8 show the calculation for magnitude and direction respectively.

$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-1,y)\big)^2 + \big(L(x,y+1) - L(x,y-1)\big)^2} \quad 3.8$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1) - L(x-1,y)}\right) \quad 3.9$$

A direction histogram is formed by weighting gradient orientations with their magnitude around the keypoint and with a Gaussian-weighted circular window with $\sigma$ equal to 1.5 of the feature's scale. The histogram has 36 bins each holding a span of 10 degrees. The highest peak in the histogram corresponds to the dominant direction of the local gradient; this is chosen as the keypoint's orientation. If there exists a peak within 80 percent of the dominant peak that orientation is also chosen and a new keypoint is made at the same location and scale but with a different orientation, thus allowing for multiple keypoints at the same location.

#### *3.2.1.4 Keypoint Descriptors*

The descriptor is a vector with 128 values that gives a distinct fingerprint for each feature. This information can then be used to track features through a video feed or even detect objects that consist of multiple features. The descriptor is built from exploiting the image gradient magnitude and direction calculated during the orientation assignment, a 16x16 Gaussian weighted search window, σ equal to one half the width of the descriptor window, is centered around the keypoint and 16 orientation histograms (4x4) are formed, similar to the histograms made during the orientation assignment. These histograms represent the feature's orientation in the surrounding region and prove to be quite immune to spatial and frequency variances.

Each histogram has 8 bins to cover the range of a full circle, giving each bin a range of 45 degrees. This broad range allows for spatial invariance when matching features; it forces moderately sized spatial differences to be classified under the same descriptor, thus allowing for resistance to spatial variance. The orientation of the feature is subtracted from the gradient orientation before assigning it to a bin; this is how the descriptor achieves rotation invariance.

The histograms yield the 128 element descriptor (4 x 4 histograms x 8 bins= 128 elements) but further processing is performed to achieve resistance to changes in illumination. First the vector is normalized to unit length. A linear change in image contrast affects all pixels uniformly, thus vector normalization cancels this affect. Non-linear illumination changes caused by camera saturation or 3D objects with non-uniform surfaces are removed by reducing the influence of large gradient magnitudes. The author suggests in [31] to threshold values larger than 0.2 and re-normalize, this means that larger magnitudes are less important and the distribution of orientations has greater emphasis.

### 3.2.1.5 *Matching SIFT Keypoints*

Previously the technique for deriving SIFT features is discussed, in order to obtain LMVs the features in subsequent frames must be matched. This is done by matching a feature descriptor against the descriptor database from the previous frame by comparing Euclidean distance between descriptors; if the distance is within tolerance then the descriptors are defined as a match. In practice this is performed using k-d trees, however, since the search is exhaustive over 128 dimensions and k-d trees provide no speedup over about 10 dimensions [31] Lowe uses a Best-Bin-First (BBF) algorithm [33] to approximate a k-d tree search. Lowe also reduces the search exhaustiveness by removing features that do not have any good match to the database; this is done by comparing the distance of the closest neighbor to that of the second closest neighbor. If the second closest neighbor is near the closest neighbor the feature is said to have a bad match and is discarded, this lies on the principle that we want our feature to be severely distinct from other features. Through experimentation Lowe discovered that a ratio of 0.8 provides a good threshold to remove bad features as shown in figure 5.

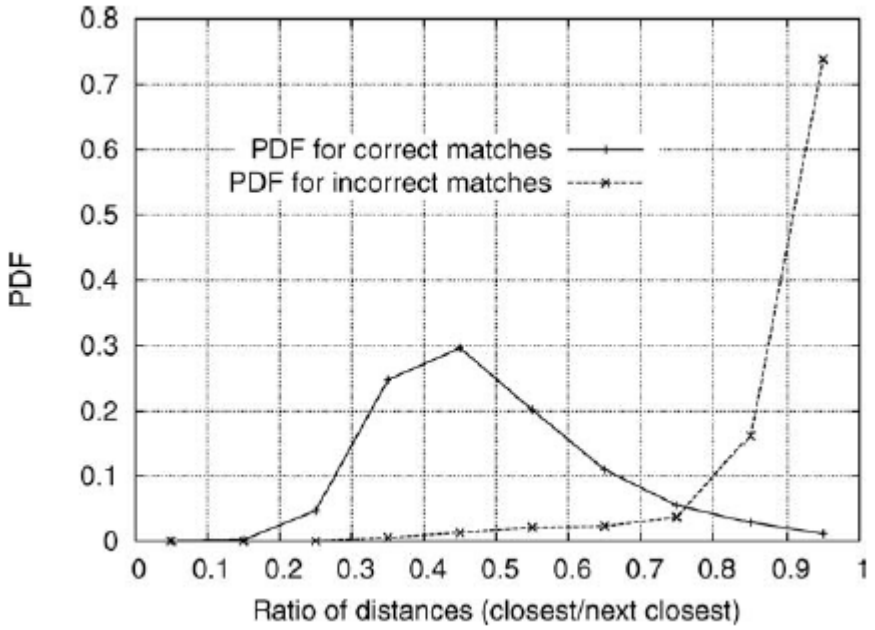


**Figure 15: Using a database of 40,000 keypoints that follow random scale and orientation change; the solid line shows the PDF of the ratio for correct matches, while the dotted line is for matches that are incorrect [31]**

The BBF algorithm modifies the k-d tree so that bins in feature space are searched in order of their closest distance from the query location. An approximate match can be returned with low cost by stopping once a specific number of nearest bins has been searched; Lowe cuts off the search after checking the first 200 nearest neighbor candidates. For a database of 100,000 keypoints, this provides a speedup over exact nearest neighbor search by about 2 orders of magnitude and results in less than 5% loss in the number of correct matches [31].

### 3.2.2 Determine Global Motion

After the keypoints have been identified and matched in consecutive frames their positions can be described as a set of local motion vectors that can be formed into a global motion model through the process of linear regression. The model implemented in this design is an affine model. An affine model is a 6 parameter transformation matrix that defines the image plane's translational and rotational motion. The model takes the form $A = MB$ where $A$ is the position of the feature point in the current frame and $B$ is its position in the previous frame, this is shown in equation 3.10.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & \Delta x \\ \sin(\theta) & \cos(\theta) & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad 3.10$$

In order to solve for the transformation matrix we must multiply $x$ by its transpose so that its inverse can be applied:

$$B^T A = M(B^T B) \quad 3.11$$

After multiplying by the inverse of $(B^T B)$ the motion model can be solved for:

$$(B^T B)^{-1} B^T A = M \quad 3.12$$

It is important to note that there are 2 different coordinate systems operating here, the location of the feature points is first found in raster coordinates and then converted to Cartesian coordinates to solve the motion model. The relationship between these systems is shown in figure 16.

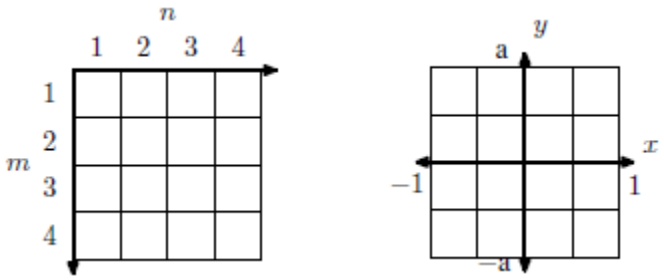


Figure 16: Raster and Cartesian coordinates respectively [15]

The Y axis limits of the Cartesian coordinate system are defined as the inverse of the aspect ratio, described in equation 3.13.

$$aspect\ ratio = \frac{H}{W} \quad 3.13$$

Where $H$ and $W$ are the image's pixel height and width respectively. The feature point, at location $(m, n)$, is converted to the normalized Cartesian system by equation 3.14 [34]:

$$x = \frac{2m - W}{S} \quad and \; y = \frac{(2n - H)}{S} \; where \; S = \max(H, W) \tag{3.14}$$

#### *3.2.2.1 Random Sample Consensus (RANSAC)*

Using all of the matched features to solve for the motion model leads to large global motion errors. These errors are caused by outlier matches that do not represent the global motion of the camera being fed into the machine that solves for the motion model. There needs to be a process that can determine which feature pairs are motion outliers and which are inliers. The process used to separate the matched features into these two categories is called, "random sample consensus" or RANSAC for short.

The method of RANSAC is to randomly choose a small subset of the matched features and solve for their motion model, this represents an initial guess and we will call it ($\widehat{M}$). $\widehat{M}$ is then applied to all other features and an estimate of where the features should be is made, this is shown in equation 3.15

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ 1 \end{bmatrix} = \widehat{M} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3.15}$$

The Euclidean distances from the estimated feature locations and their actual locations are then evaluated. This measures the error between the estimated and actual position and describes how well the estimated motion model ($\widehat{M}$) represents their motion.

$$error_i = \sqrt{\left(x_i' - \hat{x}_i\right)^2 + \left(y_i' - \hat{y}_i\right)^2} \tag{3.16}$$

If the error lies below a certain threshold $\left(\frac{10}{S}\right)$ the feature is said to fit with the model and is then added to the consensus.

Once all features have been tested, the consensus is evaluated, and if it contains at least one third of the total number of matched features it is said to be a good model. If it is determined to be a good model the full consensus error is evaluated by equation 3.17.

$$Consensus \; Error = \sum_{i=1}^{N} \sqrt{\left(x_i' - \hat{x}_i\right)^2 + \left(y_i' - \hat{y}_i\right)^2} \tag{3.17}$$

If the consensus error is less than all previous consensus errors, or it's the first consensus error calculated, then it is defined as the current best model. The RANSAC process is then repeated 10 times and the motion model with the least consensus error is determined to be the global motion model. Figure 17 is a flowchart showing the RANSAC process.

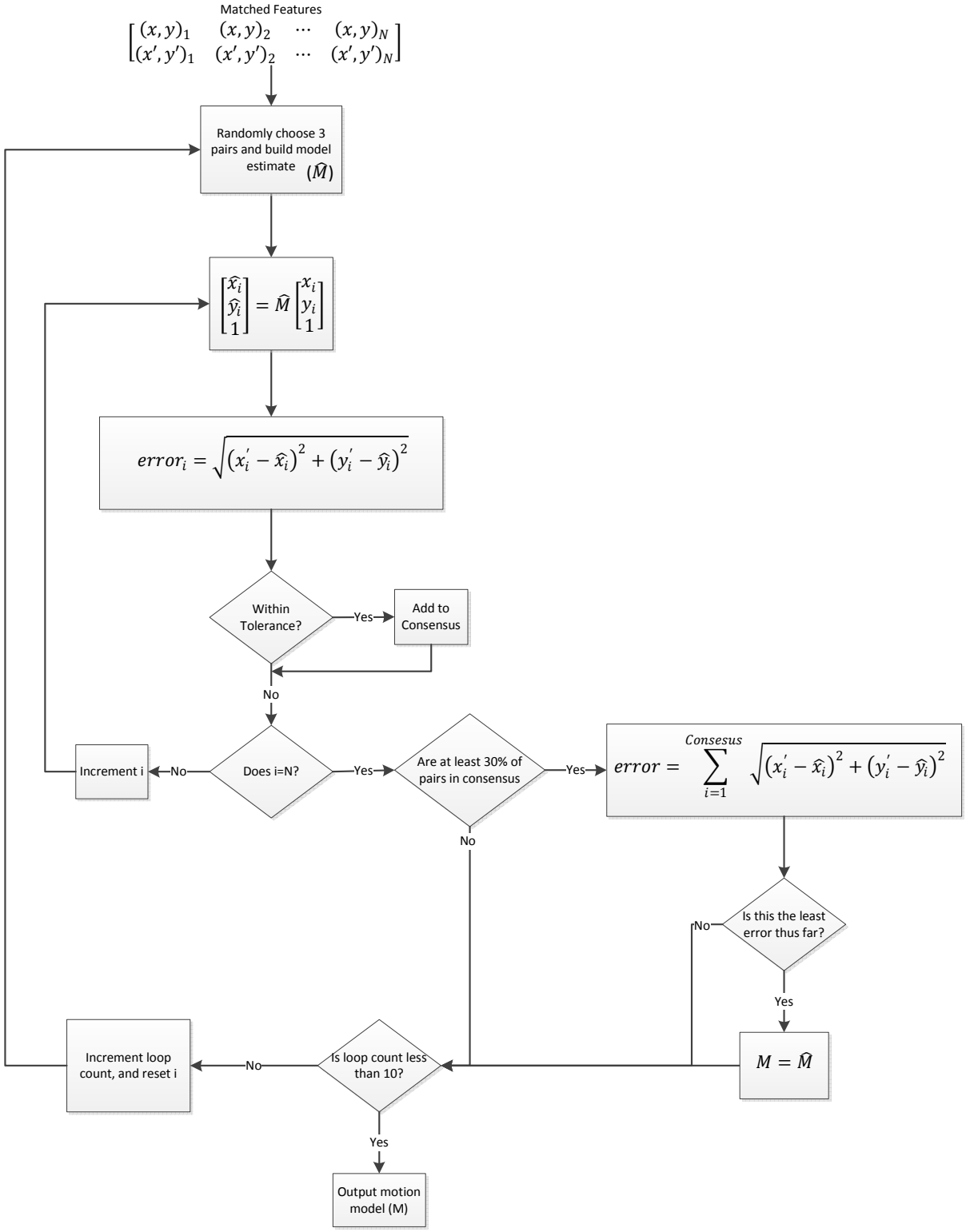


Figure 17: RANSAC flowchart

### 3.2.3 Calculate Measurement Strength

Even with removing outliers it is possible to have significant error in the global motion model. This is caused either from highly dynamic scenes where a small portion of the local motion is caused by the camera motion, or when there aren't enough features to form a good model.

Highly dynamic scenes can be determined from the tracking. Tracking error is defined as the total mean error of all feature pairs according to the model and shown in equation 3.18.

$$Tracking\ Error = \frac{1}{N} \sum_{i=1}^{feature\ pairs} \sqrt{\left(x_i' - \hat{x}_i\right)^2 + \left(y_i' - \hat{y}_i\right)^2} \qquad 3.18$$

Where $(\hat{x}, \hat{y})$ is the feature location in the previous frame transformed with the motion model $M$, and $(x', y')$ is the feature location in the current frame. Tracking error will be low when the scene represents camera motion and high when it does not. The range for tracking error was determined by calculating the error with scenes of only camera motion, then artificially introducing dynamic objects and solving again for tracking error. The comparison in figure 18 shows the result and proves it is a good measure of dynamic scenes because of the distinct separation.

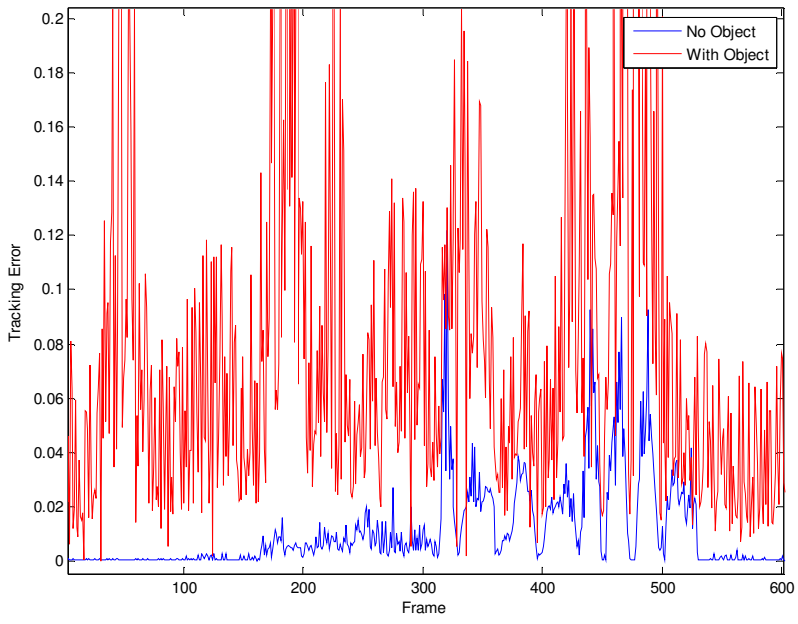


Figure 18: Tracking error of two types of scenes

Another measure of the visual measurement strength is how many features are matched in consecutive frames. When there are a few matches (less than 20) the global motion model will have

more error than if there are many matches. Using the tracking error and number of feature matches, a confidence coefficient can be determined that lies in the range of [0 1], this is shown in equation 3.19.

$$C = \frac{Tracking\ Error}{0.2}\left(1 - \frac{number\ of\ matches}{1864}\right) \quad 3.19$$

## 3.3 Inertial Measurement

The physical motion of the camera is measured using an inertial measurement unit (IMU). This device is made up of accelerometers and gyroscopes that provide information on the systems acceleration and angular rate.

An accelerometer can detect linear acceleration in the X, Y and Z directions and the output of the accelerometer contains the linear acceleration with a few error sources as shown in equation 3.20.

$$a(t) = S * a(t) + b + b(T) + \delta(t) + n_{accel} \quad 3.20$$

This shows that the linear acceleration is equal to the actual acceleration multiplied by a scaling factor ($S$), a constant offset ($b$), a temperature dependence $\big(b(T)\big)$, a random walk ($\delta(t)$) and a zero mean Gaussian distributed noise ($n$).

Gyroscopes measure the rate of rotation about the X, Y and Z axes and their output is affected by the same type of error as the accelerometer.

$$w(t) = S * w(t) + b + b(T) + \delta(t) + n_{gyro} \quad 3.21$$

The random walk can be modeled as a first order Gauss-Marcov process which is a stochastic process where all the future values are scaled from past values plus a random input; this is described in the first order equation 3.22:

$$\dot{\delta} = -\frac{1}{\tau}\delta + w \quad 3.22$$

Where

$$w = \sqrt{\frac{2f_s\sigma_{bias}^2}{\tau}}v \quad 3.23$$

And whose statistical properties include:

$$E[\delta] = 0 \quad 3.24$$

$$E[\delta^2] = \sigma_{bias}^2 \quad 3.25$$

The noise driving the bias ($v$) is normally distributed with zero mean and a sampled covariance of one [35].

$$v \in N[0,1] \quad 3.26$$

The autocorrelation of a first order Markov process is described by equation 3.27. The values $\rho$ and $\beta$ can be estimated with an exponential curve fit of the IMU data during a period of no motion [15].

$$R(\tau) = \rho e^{-\beta|\tau|} \quad 3.27$$

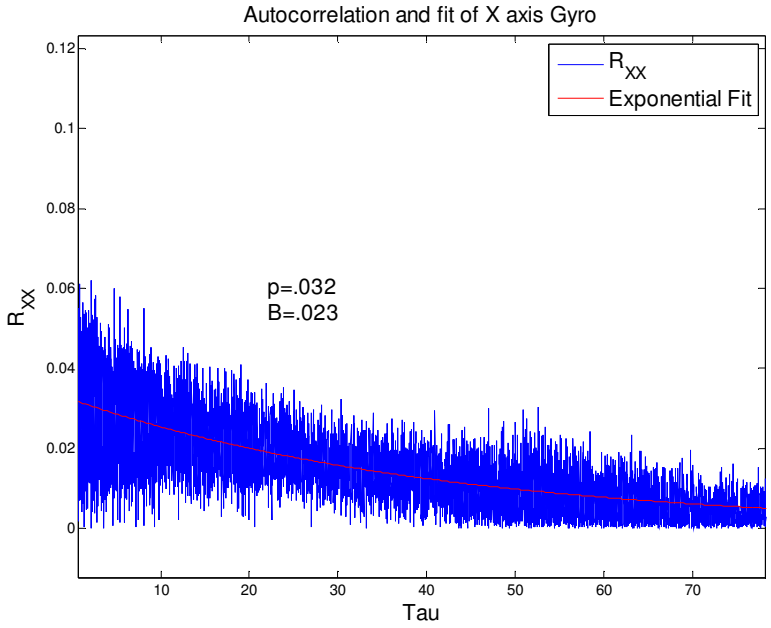


Figure 19: Autocorrelation of X gyro

Figure 19 shows the autocorrelation of the x axis gyroscope and it can be seen that it takes the form of an exponentially decaying function. The values for $\rho$ and $\beta$ were determined by averaging the results from 3 stationary tests of the IMU's gyroscopes. This is the same technique used by Smith et al. in [15] and the results are shown in table 1. These values will be used in the Kalman filter to model the variances of the bias states.

Table 1: First order Markov values

| | Run 1 | Run 2 | Run 3 | **Mean** |
|---|---|---|---|---|
| $\boldsymbol{\theta_x}$ | $\rho = .032$<br>$\beta = .023$ | $\rho = .008$<br>$\beta = .016$ | $\rho = .011$<br>$\beta = .024$ | $\rho = .017$<br>$\beta = .021$ |
| $\boldsymbol{\theta_y}$ | $\rho = .028$<br>$\beta = .017$ | $\rho = .005$<br>$\beta = .003$ | $\rho = .007$<br>$\beta = .010$ | $\rho = .0133$<br>$\beta = .01$ |
| $\boldsymbol{\theta_z}$ | $\rho = .054$<br>$\beta = .013$ | $\rho = .005$<br>$\beta = .001$ | $\rho = .006$<br>$\beta = .009$ | $\rho = .0217$<br>$\beta = .0077$ |

It is important to note these values can usually be found in the product datasheet or literature of the sensor and thus it is not always necessary to calculate them. The above calculations where performed because the literature specific to the IMU used in this thesis did not contain this information.

The random walk for MEMS inertial sensors grow at a constant time step, therefore it is also possible to simplify modeling the drift as a constant. This can still provide an accurate solution and has been implemented by Hol J. et al. in [36].

### 3.3.1 Accelerometer Calibration

Recall from equation 3.20 the acceleration is affected by a scale factor ($S$) and a constant offset ($b$). These are constants that represent axes misalignment and crosstalk effect between different channels caused by the sensor electronics [37].

$$S = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \tag{3.28}$$

$$b = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} \tag{3.29}$$

In a perfect accelerometer the matrix ($S$) would be the identity and ($b$) would be zero, but in reality they can have cross-axis factors of as much as 2 percent [38].

The matrices can be solved for by having the accelerometer undergo a series of highly accurate known accelerations and comparing the output. Through a regression technique the optimal parameters for matrices ($S$) and ($b$) can be calculated. The equipment required is expensive and must be able to deliver milli-G accuracy in a large range. When expensive calibration equipment is not available a reasonably precise calibration can be performed using the gravity vector.

MEMS accelerometers are affected by gravity and if no external force is applied the accelerometer output should yield:

$$g = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{3.30}$$

Using this principle the error of the accelerometer while in a static condition can be described as:

$$\begin{aligned} e_k &= a_x^2 + a_y^2 + a_z^2 - g^2 \\ &= \sum_{i=x,y,z} \left( \sum_{j=x,y,z} \left[ S_{ij}(V_j, k - b_j) \right]^2 \right) - g^2 \end{aligned} \tag{3.31}$$

A cumulative error can be found by summing the $e_k^2$ term for all k:

$$E(b_x, b_y, b_z, S_{xx}, S_{yy}, S_{zz}, S_{xy}, S_{xz}, S_{yz}) = \frac{\sum_{k=1}^{N} e_k^2}{N} \tag{3.32}$$

Notice there are only six terms that describe the nine element matrix $S$. This is possible because $S$ is symmetric and can be described with only six parameters. The cumulative error ($E$) is a non-linear function of the sensor's parameters, it best fits the observable data in the least squares sense and can be solved by minimizing the error with respect to the parameters ($b$ and $S$) [38].

A common method to minimize the error and solve for the calibration parameters is Newton's method, an iterative procedure that guarantees quadratic convergence [39]. Newton's method starts

from an initial guess and using the sensor's datasheet an educated guess can be made; the solution is then iterated according to equation 3.33:

$$x^{t+1} = x^t - \alpha[H^{-1}(x^t) * J(x^t)]$$
$$H(x^t) = \left\{ h_{ij} = \frac{\partial E}{\partial x_i \partial x_j} \right\} \quad 3.33$$
$$J(x^t) = \left[ \frac{\partial E}{\partial x_1} \quad \cdots \quad \frac{\partial E}{\partial x_9} \right]$$

Where $H(x^t)$ is a 9x9 hessian matrix and $J(x^t)$ is a 1x9 jacobian matrix, $x^t$ is the unknown vector at the current iteration ($t$) that contains the bias vector and the six independent elements of the scale factor matrix. $\alpha$ is a damping parameter less than one that is updated every iteration by means of a line search procedure [39].

The iterations are complete once the updated vector $x^{t+1}$ produces little change from the previous iteration, shown in equation 3.34; this means the error function ($E$) has reached a local minimum. The authors in [38] empirically found $1.5 * 10^{-6}$ to be a suitable threshold that generally converges within ten iterations.

$$max\left\{ \left| \frac{x^t - x^{t-1}}{x^t + x^{t-1}/2} \right| \right\} < \epsilon \quad 3.34$$

### 3.3.2 Gyroscope Calibration

Unfortunately the Earth does not give us a useful tool to calibrate the gyroscope, only gravity for the accelerometer. A highly accurate gyro could be calibrated with the Earth's rotation, but this is only $4.2 \times 10^{-3}\ ^\circ/s$ and a camera's motion is generally several orders of magnitude higher. Thus arises the need for calibration using a rotating platform of known velocity.

The platform used is a standard record player capable of rotating three speeds, 33.3RPM 45RPM and 78RPM. Data was collected at these speeds and fitted to a line in order to estimate non-orthogonality (diagonal elements of scale factor matrix (S) in equation 3.24) and the constant bias. The slope defines the non-orthogonality and the offset defines the constant bias. The results are shown in figure 20.

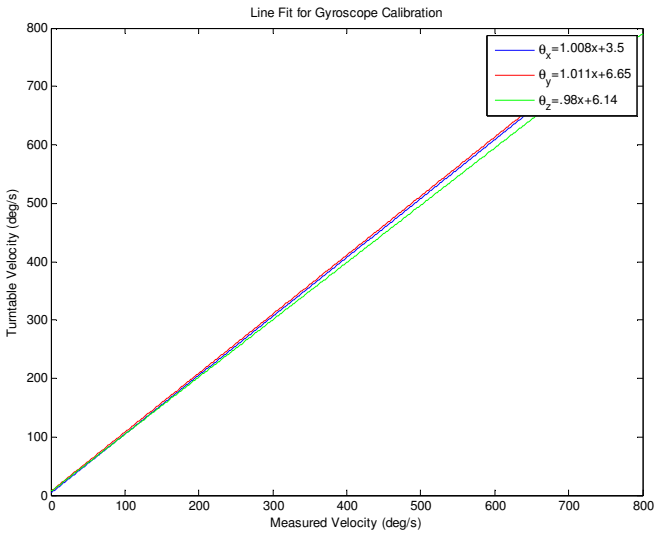


Figure 20: Gyroscope calibration

The gyroscope is also actively calibrated during run time. Whenever a one second period of no motion occurs the gyroscope bias is calculated and removed from the measurement. No motion is determined from the accelerometer measurements according to equation 3.36.

$$motion = \begin{cases} \sqrt{\ddot{x}^2 + \ddot{y}^2 + \ddot{z}^2} \neq 1 = true \\ \sqrt{\ddot{x}^2 + \ddot{y}^2 + \ddot{z}^2} = 1 = false \end{cases} \quad 3.35$$

### 3.3.3 IMU Design

The IMU designed to perform inertial fusion with digital stabilization is capable of measuring 3 axis acceleration and a 3 axis angular rate. The IMU communicates with the host computer through a USB bus and can deliver sensor measurements at a rate of 1 KHz. The complete unit is comprised of four main parts; the host processor, the inertial sensor, the voltage regulator and a USB front end. The PCB is shown in figure 21 where the main parts are highlighted.

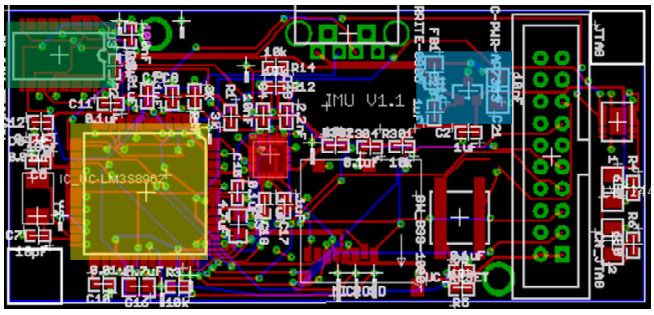


Figure 21: IMU unit with processor (yellow), power supply (blue), inertial sensor (red) and USB transceiver (green) highlighted

### 3.3.3.1 Host processor

The main processor onboard the IMU unit is an ARM Cortex-M3 and is highlighted as yellow in figure 21. The processor's function is to communicate with the inertial sensor to retrieve measurements, combine those measurements into a robust message packet and send the data to the host computer over the USB communication line.

The processor communicates with the inertial sensor through a standard $i^2c$ serial interface. The processor is always the master and the sensor is always the slave. The bus runs at 400 KHz with the sensor having a seven bit address size (0x68) and the logic high level is set at 3.3 volts. As is common with all $i^2c$ communication there are two communication lines on the bus, a clock (SCA) and a data line (SDA). The processor follows the protocol shown in figure 22 to communicate with the inertial sensor.

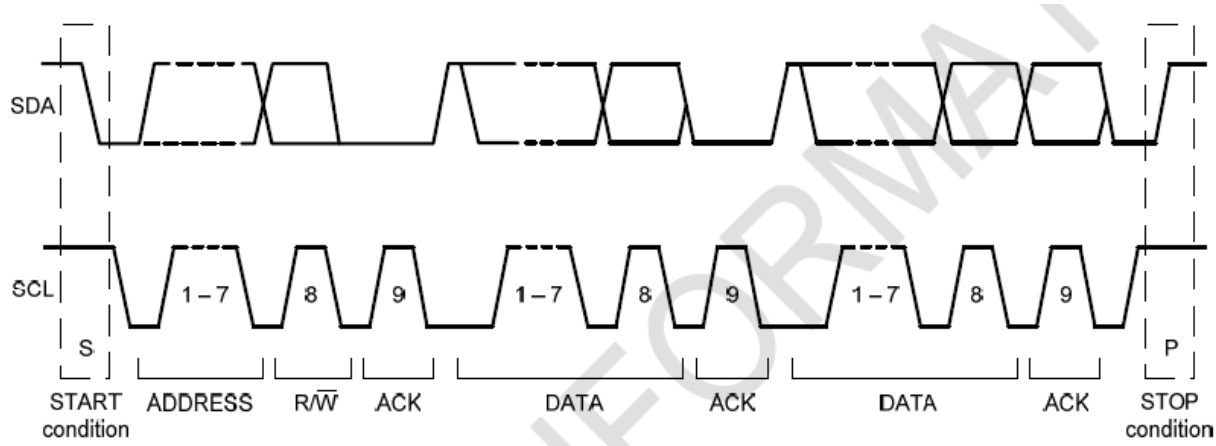


Figure 22: Complete I2C data transfer [40]

The communication begins with the start condition initiated by the master then the 7 bit address is sent followed by a single read/write bit, which indicates whether the master is writing to the slave or requesting data from the slave. The master then releases the SDA line and waits for an acknowledgment (ACK) from the slave. An ACK is performed by pulling the SDA line low for the full high period of the SCL line. After the ACK is received the master will either send or receive the data (depending on if it is a read or write operation), each data packet is of one byte length and is always preceded by an ACK. Once the operation is complete the master performs a stop command and the I2C bus is free until the next start command is received.

The processor sends out sensor measurements over the USB line to a host computer by communicating with a USB transceiver chip manufactured by Future Technology Devices International. The transceiver chip (FT232R) handles the USB protocol and communicates to the processor through a universal asynchronous receiver/transmitter (UART) connection at a rate of 3Mbps.

### 3.3.3.2 Power Supply

The power for all devices on the PCB is derived from the 5 volt USB power supplied by the host computer. The 5 volts is linearly regulated to provide an output of 3.3 volts with an available maximum current draw of 300mA. The location of the power supply system is highlighted in blue in figure 21. The

input power is decoupled from power supply glitches and noise with a 1uF capacitor shunted to ground and another 1uF capacitor is shunted on the output for stabilization and to help reduce transient response. The regulator requires no minimum load and provides current limit and short circuit protection. All components in the IMU unit run off of 3.3 volts and the complete unit has a maximum current draw of 100mA.

#### 3.3.3.3 Inertial Sensor

The inertial measurement device is a 3 axis accelerometer and 3 axis gyroscope both etched on the same silicon and highlighted in red on figure 21. It is the MPU-6050 model manufactured by Invensense Inc. The gyroscopes offer a programmable output rate up to 8Khz and their mechanical frequencies are driven at 33Khz, 30Khz and 27Khz (x, y and z axis respectively). The accelerometer has a programmable output rate up to 1 KHz. The gyroscopes have a programmable sensitivity range of 250 deg/s to 2000 deg/s and the accelerometer offers a range of 2g to 16g. The sensor employs a 16 bit analog to digital converter and provides its own internal clock at 8 KHz.

#### 3.3.3.4 Software Architecture

There are two parts to the software architecture of the IMU, one is the data acquisition and message transmission residing on the actual IMU and the other is the message reception and parsing performed by the host processor. This section describes the process residing on the IMU.

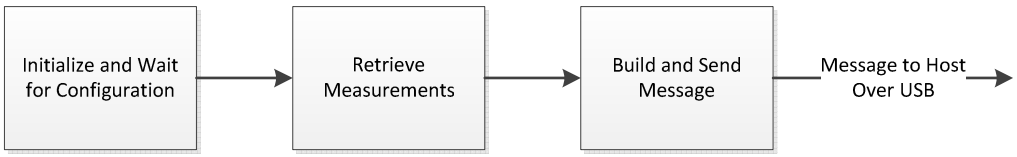


Figure 23: IMU software structure

Upon power up the microcontroller enables two timer interrupts, one for triggering the camera (30Hz) and another for controlling sample rate (variable, set during configuration). It then initializes the IMU; the initialization process is chronologically described in the preceding list:

1. Wake up accelerometers, gyroscopes and temperature sensor.
2. Set the IMU clock to be phase locked loop (PLL) at 8Khz
3. Set the gyroscope sample rate to 8Khz and the accelerometer sample rate to 1Khz
4. Disable the IMU's built in digital low pass filter (DLPF)

After initialization the microcontroller remains in a waiting stage where it polls the USB bus for configuration messages from the host computer. These messages are described in table 2 below.

Table 2: IMU configuration messages

| Configuration Message | Description |
|---|---|
| Accelerometer Sensitivity | Configures the range of the accelerometer measurements (±2g, ±4g, ±8g or ±16g) |
| Gyroscope Sensitivity | Configures the range of the gyroscope measurements (±250◦/s, ±500◦/s, ±1000◦/s or ±2000◦/s) |
| Sample Rate | Sets the rate at which measurements are sent over the USB bus (4Hz → 500Hz) |

The IMU does not begin sending messages until it receives the start command from the host processor. This allows the USB bus to be free from traffic until the host computer is ready to receive data. Implementing the start commands not only prevents unnecessary loss of bandwidth but also prevents the host computer's operating system from perceiving the IMU messages as another USB device (such as an external hard drive or a USB mouse) and causing operating system failures.

After the start command is received the IMU acts as a state machine with three possible states shown in figure 24 below. The control flag is set through the two timer interrupts, when the sample rate interrupt occurs the control flag is set to 1 and when the trigger camera interrupt occurs the flag is set to 2. After the measurement or trigger state is completed they set the control flag to zero and the program returns to the "do nothing" state.

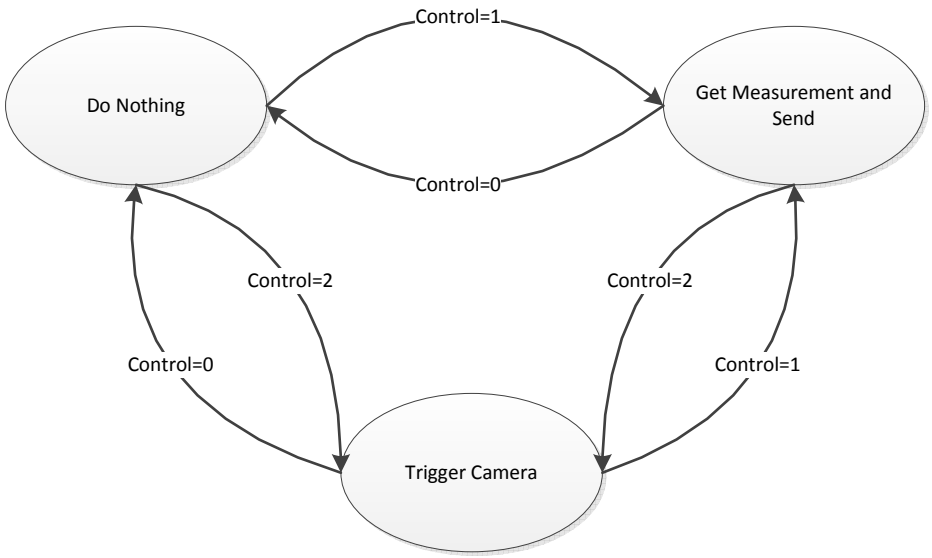


Figure 24: IMU state diagram

### 3.3.3.5 IMU State Description

Once the measure and send state is reached the microcontroller retrieves 14 bytes of data from the inertial sensor via the I2C serial bus. The 14 bytes comprise the inertial measurements, 2 bytes each of the three axis accelerometer, 2 bytes each of the three axis gyroscope and 2 bytes for the

temperature measurement. The microcontroller then builds the message, performs a standard exclusive OR checksum, converts it into a string and sends it to the host processor via the USB bus. The total length of the message is 25 bytes and the format is shown in table 3.

Table 3: IMU message structure

| Header | Data | | | | | | | | | Checksum | Footer |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0xFFFF | Xaccel | Yaccel | Zaccel | Xgyro | Ygyro | Zgyro | Temp | Time | Message number | 8 bits | 0xABCD |

The camera trigger timer interrupt is set to occur every 16ms. When in this state the output pin used to externally trigger the camera is toggled, effectively triggering the camera every 32ms or a rate of 31.25 frames per second (fps). This is graphically shown in figure 25.

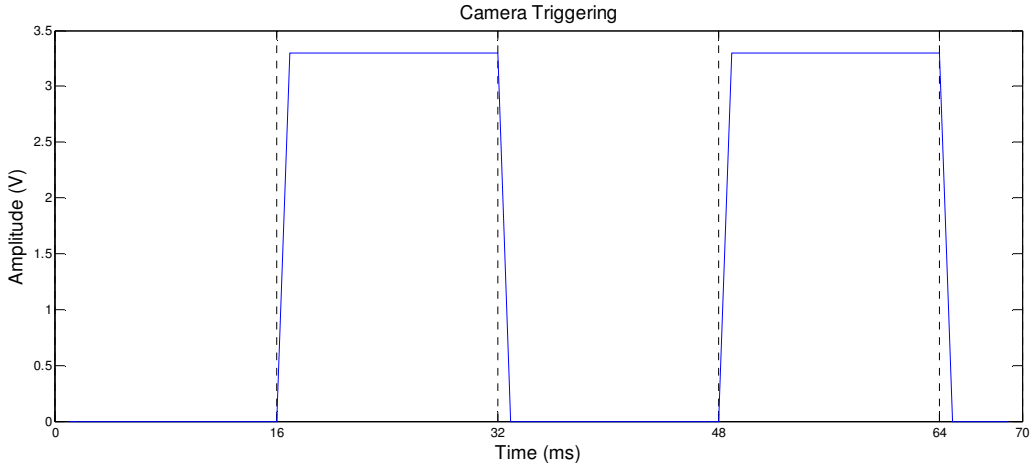


Figure 25: Camera triggering

## 3.4 Digital and Mechanical Fusion

The inertial and visual measurements are combined through the use of a Kalman filter. Combining the two types of measurements provides significant advantage over using either one alone. Visual measurements are highly accurate but susceptible to error from dynamic scenes, plus they are computationally expensive to solve. Inertial measurements are computationally inexpensive and immune to error from dynamic scenes, but they have bias and noise errors that decrease their accuracy over time. Using a Kalman filter, the strengths of both measurements can be combined to form a robust image stabilizer.

### 3.4.1 Dynamic Model

The Kalman filter is a data fusion process that recursively estimates the states of the system to provide a more accurate estimation than any one measurement alone. In the case of the image

stabilizer the system is the dynamics of the camera and the states are the camera's angular position, angular velocity and IMU bias shown in equation 3.36.

$$x = \left[\theta_x\ \theta_y\ \theta_z\ \dot{\theta}_x\ \dot{\theta}_y\ \dot{\theta}_z\ b_x\ b_y\ b_z\ \right]^T \tag{3.36}$$

The dynamic model of the system is described as:

$$\dot{x} = Fx + Q \tag{3.37}$$

The system dynamics matrix ($F$) is defined as:

$$F = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\beta_x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta_y & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta_z \end{bmatrix} \tag{3.38}$$

Where $(\beta_x, \beta_y, \beta_z)$ are derived from the autocorrelation analysis of the gyroscope's bias in section 3.3 and shown in figure 19.

### 3.4.2 Process Noise

The process noise matrix $(Q)$ defines the uncertainty of the states and is calculated by evaluating the covariance of the states. This is performed by acquiring data while the system is motionless and calculating the covariance matrix according to equation 3.39.

$$Q = E[(x - \bar{x})(x - \bar{x})^T] \tag{3.39}$$

The diagonal terms of the matrix Q define the variance of the states and the non-diagonal terms are their respective covariance. It is expected for the covariance terms to be at least an order of magnitude less than the variance terms because the process noise should be uncorrelated between states. This is the case for the matrix Q and thus the covariance terms are said to be zero, allowing the Kalman gain in equation 3.41 to be calculated faster.

Equation 3.39 is only performed on the angular position and velocity states, the variance of the bias states are found by following the method used by Smith et al. in [15] where their values are derived from the autocorrelation analysis and take the form:

$$\sigma^2 = 2\beta\rho \tag{3.40}$$

The final values for the state uncertainty matrix are:

$$Q = \begin{bmatrix} 1.8E-5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.5E-7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.1E-5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9.6E-3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 8.8E-3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.2E-2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7.1E-4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8.4E-4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.9E-4 \end{bmatrix}$$

### 3.4.3 Ricatti Equations

The ability for the Kalman filter to fuse together the two measurements and significantly reduce error caused by zero mean Gaussian noise lies with the use of Ricatti equations. These equations effectively solve for the optimal gain of each state while taking into account process and measurement noise. The Ricatti equations take the form described in equation 3.41.

$$\begin{aligned} M &= \Phi \mathrm{P} \Phi^{\mathrm{T}} + Q \\ K &= MH^T(HMH^T + R)^{-1} \\ P &= (I - KH)M \end{aligned} \qquad 3.41$$

A Kalman filter respectively loops through the Ricatti equations and applies the Kalman gain ($K$) to the measurement in order to update the state estimate ($\hat{x}$). The covariance matrix $M$ is a time projection of the errors in the state estimate, it is a function of the covariance matrix from the previous time step ($P$) that is projected forward using the fundamental matrix ($\Phi$). The fundamental matrix projects the states forward in time and is derived from the system dynamics matrix ($F$) by following equation 3.42

$$\Phi = \mathcal{L}^{-1}[(sI - F)^{-1}] = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad 3.42$$

### 3.4.4 Measurement Matrix

The measurement matrix ($H$) provides a linear relationship between the visual measurement update ($z$) and the mechanical measurement process model ($x$) as shown in equation 3.43.

$$z = Hx + v \qquad 3.43$$

The values for $H$ vary depending on the scene being viewed, for far field scenes a pitch or yaw motion of the camera will have less effect on the image plane than scenes of closer objects. In order to

create a robust stabilizer that is capable of functioning in different scenes, $H$ must be calculated for the specific scene. This requires the system to perform a pitch and yaw motion, then calculate the visual measurement and perform a gradient descent algorithm to match the angular position derived from the inertial measurement to the visual measurement. The gradient descent effectively finds $H$ that provides the minimum error between the visual and inertial measurements. The values for the $H$ matrix while viewing an indoor scene is shown in equation 3.44 and the programmatic flow is described in figure 26.

$$H_{indoor} = \begin{bmatrix} -14.25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 12.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad 3.44$$

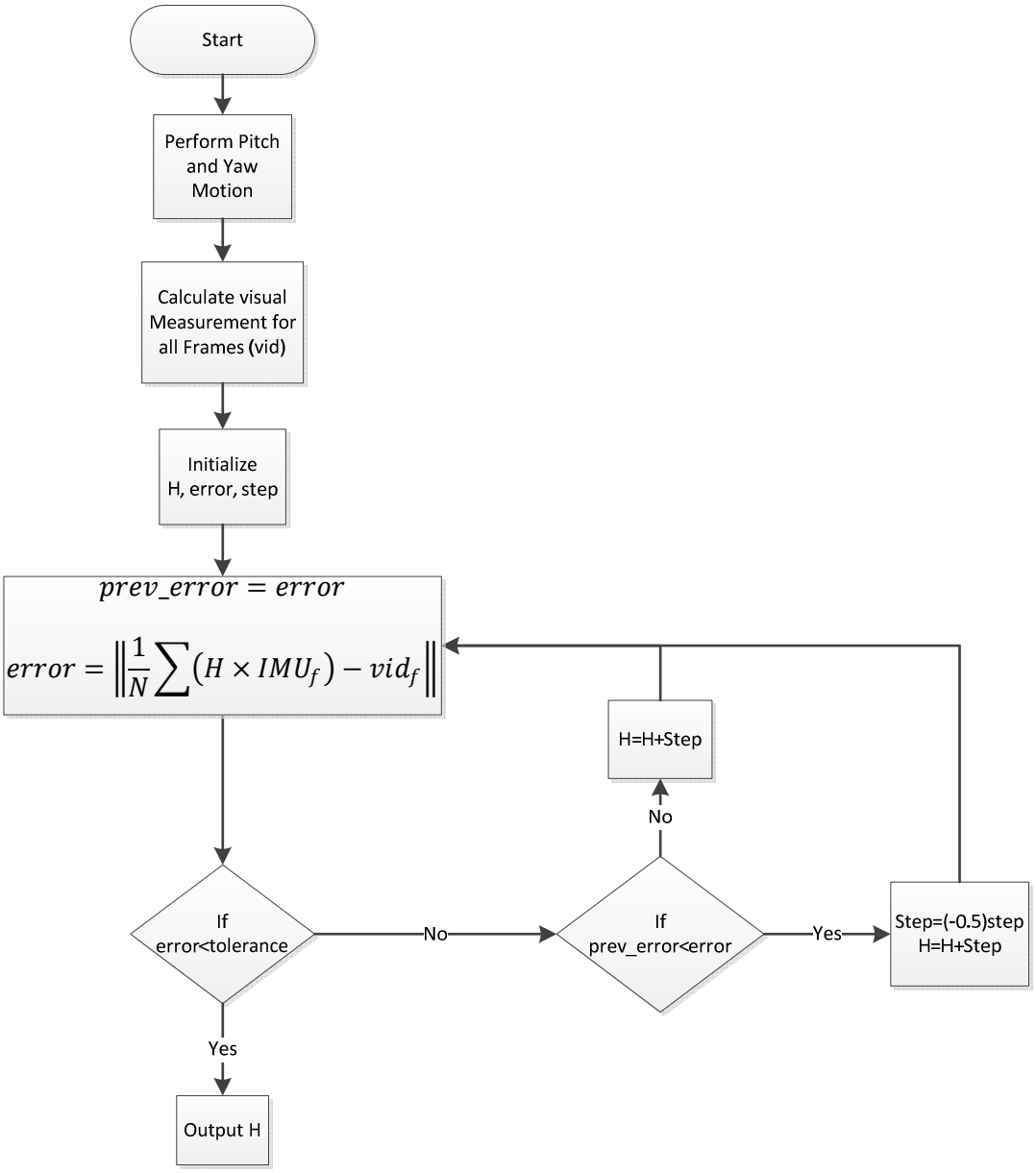


Figure 26: Gradient descent to solve for H parameters

### 3.4.5 Measurement Noise

The measurement noise matrix ($R$) defines how much of the visual measurement is incorporated in the fusion process. It is a function of the measurement strength coefficient derived in section 3.2.3, when the strength coefficient ($D$) is near zero, all of the visual measurement is desired, when it is close to one, the visual measurement is said to be flawed and none of it is desired.

The values for the $R$ matrix are found using the same method as Smith et al. in [15]. Data is acquired from the sensor rig as it is left motionless and thus the angular rate from the inertial

measurement is zero plus process noise. The visual measurement is simulated by a zero vector and the Kalman filter is performed through a wide range of R values as shown in figure 27.

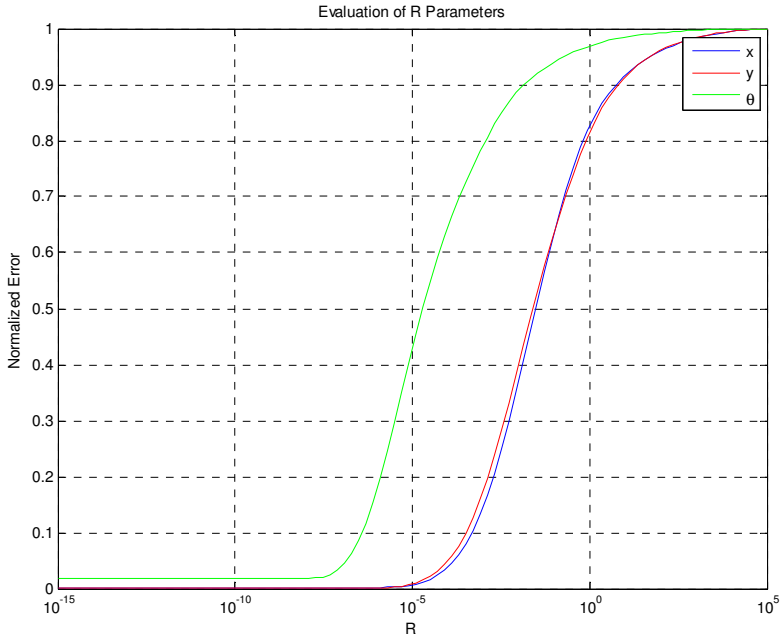


Figure 27: Normalized error versus R

Looking at figure 27 it can be seen that when the $R$ value is low the filter relies heavily on the visual measurement and thus results in a smaller error, as the $R$ value increases the error increases, this is because the filter is relying less on the visual measurement and more on the angular rate from the IMU. The relationship is obviously exponential, and we wish to find a function that fits the linear portion on the curve so that when the measurement strength coefficient is low the $R$ value is at the bottom of the curve and when the coefficient is near 1 the $R$ value is at the top. The values for the final measurement matrix $R$ are shown in equation 3.45.

$$R = \begin{bmatrix} 8.1 \times 10^{-5} e^{11.63D} & 0 & 0 \\ 0 & 8.1 \times 10^{-5} e^{11.63D} & 0 \\ 0 & 0 & 7.5 \times 10^{-8} e^{13.96D} \end{bmatrix} \quad 3.45$$

Figure 28 shows the complete process to fuse together inertial and visual measurements through the use of a Kalman filter. The preceding sections will describe removal of wanted motion from the state vector $\hat{x}$ and an analysis of the performance and capabilities of the full system.

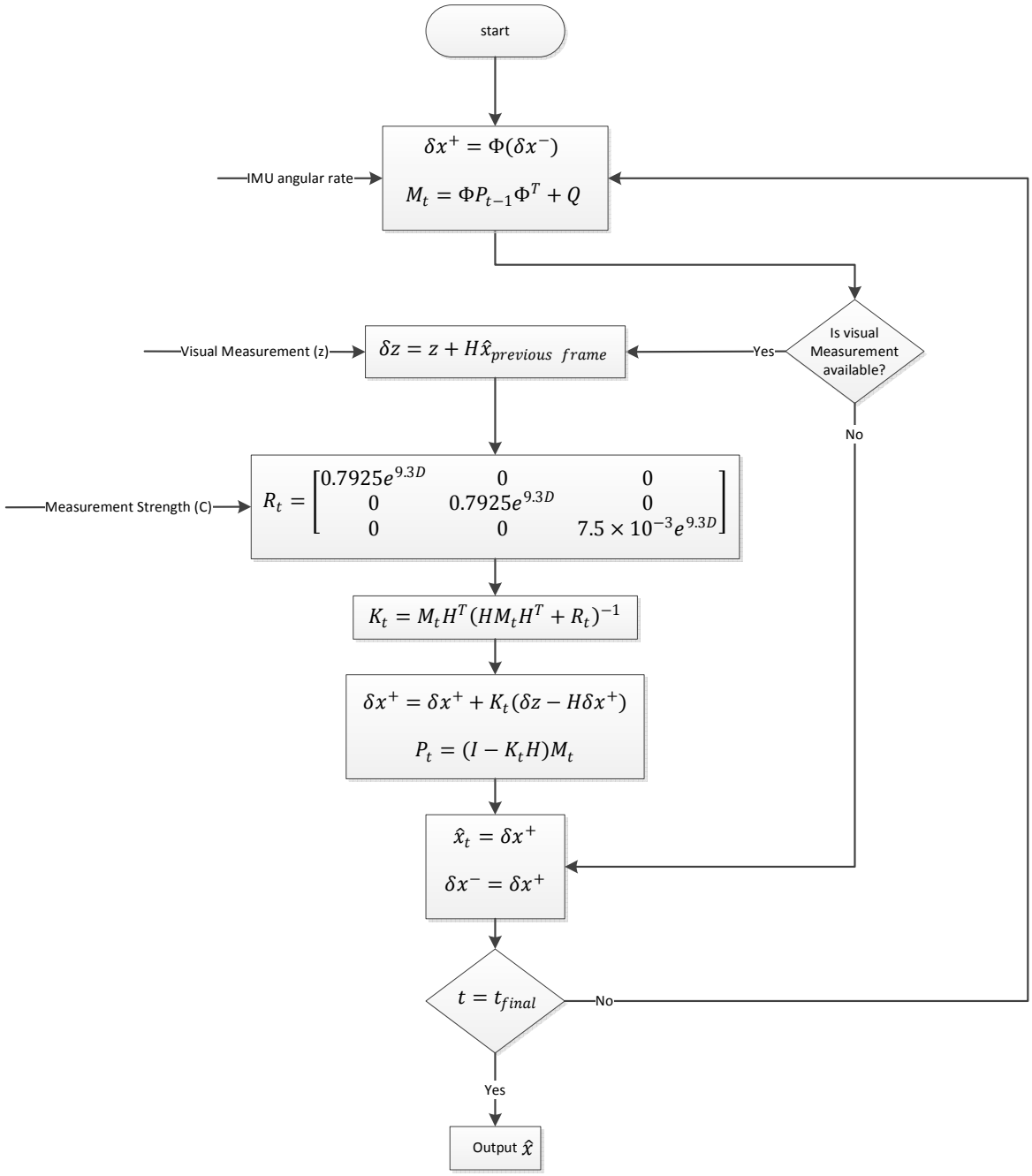


Figure 28: Kalman filter process

## 3.5 Motion Filtering

The state vector $\hat{x}$ calculated in the Kalman filter contains all the motion induced in the camera. This motion contains two parts; wanted and unwanted motion. Wanted motion is defined as the movement of the camera that should remain after the stabilization process; this includes panning, zooming and tilting of the camera. Unwanted motion is defined as the camera movement that should be removed through the stabilization process; this includes shake and jitter. In order to provide an effective image stabilizer these two types of motion must be separated from $\hat{x}$ and frame compensation performed solely on unwanted motion.

The main dividing factor between these two motion types is frequency content. The wanted motion is a slower and more purposeful type, and therefore consists of lower frequencies. The unwanted motion is faster with less energy content and consists of higher frequencies. The specific cutoff frequency to determine the separation of wanted and unwanted motion is dependent upon the situation the camera is in. If the camera is mounted on a vehicle, the engine and road noise is considered to be the unwanted motion; but if the camera was held by a person walking, their steps and hand shake is considered the unwanted motion. The former would have greater frequency content than the latter and thus require a cutoff frequency greater than the walking photographer.

The frequency analysis performed in this thesis is performed manually by shaking the sensor rig. The purpose of this thesis is to build a system that can be ported to a specific application, so an in depth frequency analysis of various types of motion (vehicle, human…) is not performed. However, the filter is designed to easily modify the cutoff frequency for different applications.

Video was taken that involves wanted and unwanted motion. The video performed pan and tilt motions while being vigorously shaken in the vertical direction making most of the unwanted motion lie on the y axis of the image plane. Figure 29 shows the motion perceived in the image plane, notice the rapid yet small motions seen on the vertical axis ($dy$). These faster small motions represent the unwanted motion that the stabilizer wishes to remove.

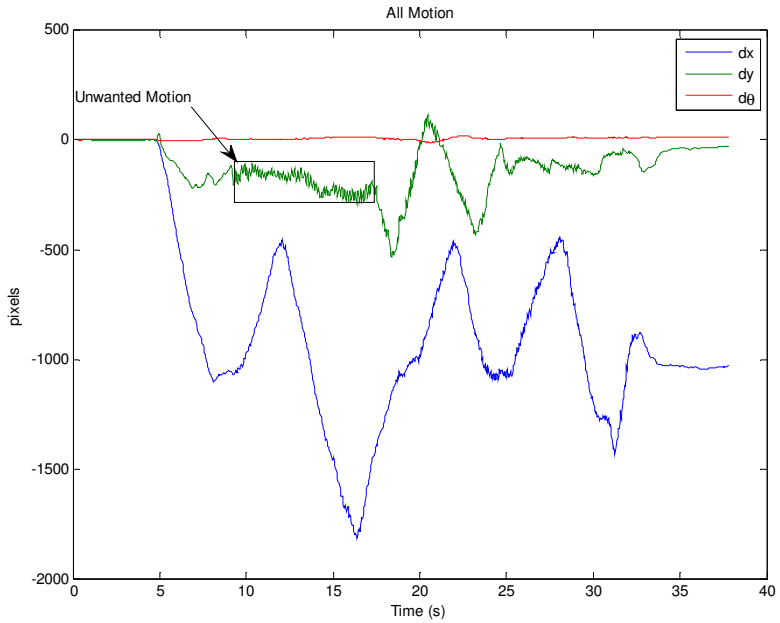


Figure 29: Wanted and unwanted motion from video feed

A frequency analysis of the vertical component ($dy$) shows the general range where the cutoff frequency should reside as shown in figure 30. It can be seen that after around 0.5 Hz there is little frequency content, this explains that the wanted and unwanted motion lie in the range of zero to one half Hz. Through experimentation the desired cutoff frequency can be found, this requires a bit of trial and error; choose a cutoff frequency, apply the filter and view the results. Performing this trial and error process leads to a cutoff frequency of 0.45Hz.

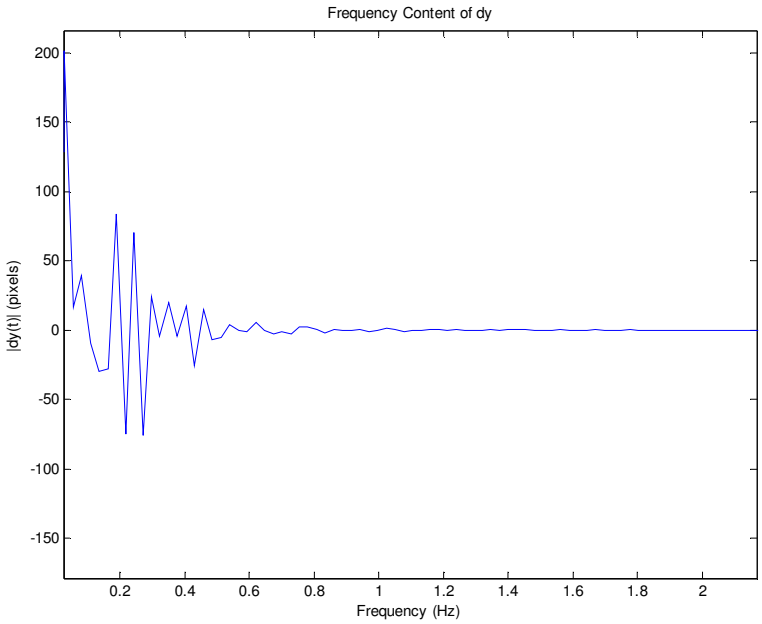


**Figure 30: Frequency analysis of hand shaken video**

Once the cutoff frequency is found the filter can be applied, this thesis incorporates a 2nd order Butterworth low pass filter to separate the two motion types. The transfer function of the low pass filter is described in equation 3.46, where $\omega_c$ represents the cutoff frequency. The frequency response of the filter is shown in figure 31.

$$H(s) = \frac{1}{1 + \left(-\frac{s^2}{\omega_c^2}\right)^2} \qquad 3.46$$

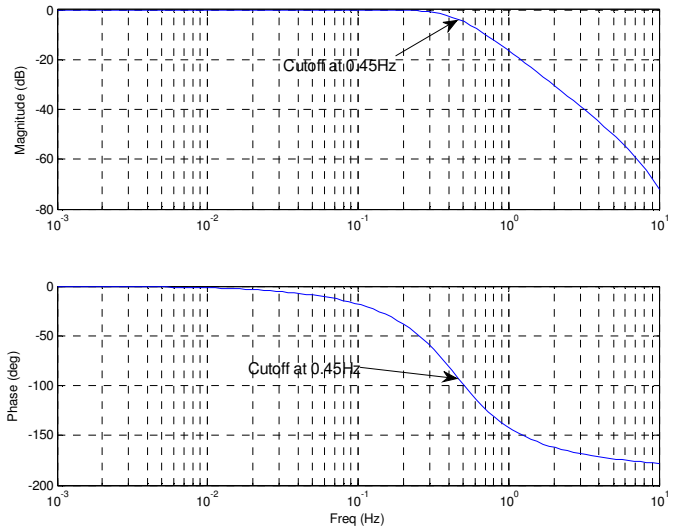


Figure 31: Frequency response of motion filter

The unwanted motion can now be separated from the full motion derived from the state vector $\hat{x}$ by following equation 3.47, the results are shown in figure 29.

$$Unwanted\ Motion = all\ motion - filtered\ motion \quad 3.47$$

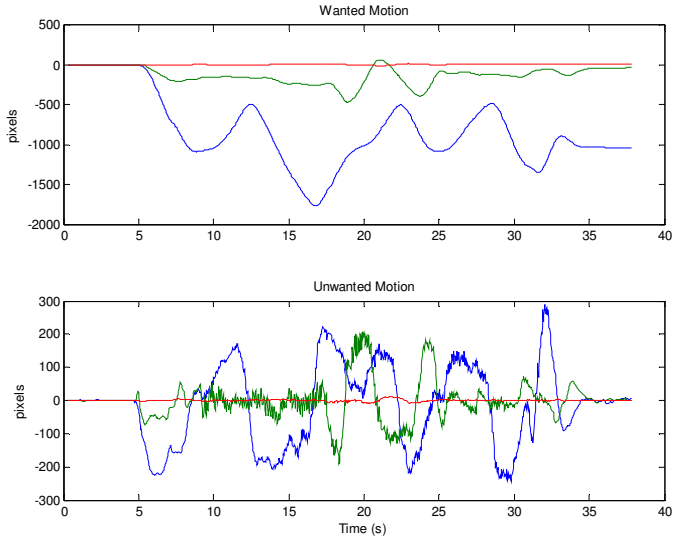


Figure 32: Wanted and unwanted motion

## 3.6 Frame Compensation

This section began by described how to derive motion of the image plane by tracking features in consecutive frames of a video feed. The preceding section explained the design, build and calibration of the inertial measurement unit (IMU). It was then shown how to combine the two measurements through Kalman filtering to provide a robust measurement of the camera's motion. After filtering the motion and removing the wanted motion the stabilizer is now ready to manipulate the image plane and provide a stable video feed.

The stabilization is realized by moving the frame in the direction opposite of the unwanted motion. This can be done in one of two ways; using the full image frame or only using only a portion of the image frame. When using the full image frame black border regions occur when the frame is manipulated as shown in figure 33, this is because there is no information available from the video feed about these regions. This can be alleviated by manipulating and outputting only a subset of the image frame, but comes at a cost of lower resolution and loss of information. The comparison of these two methods can be seen in figure 34.


Figure 33: Stabilized image with black border region

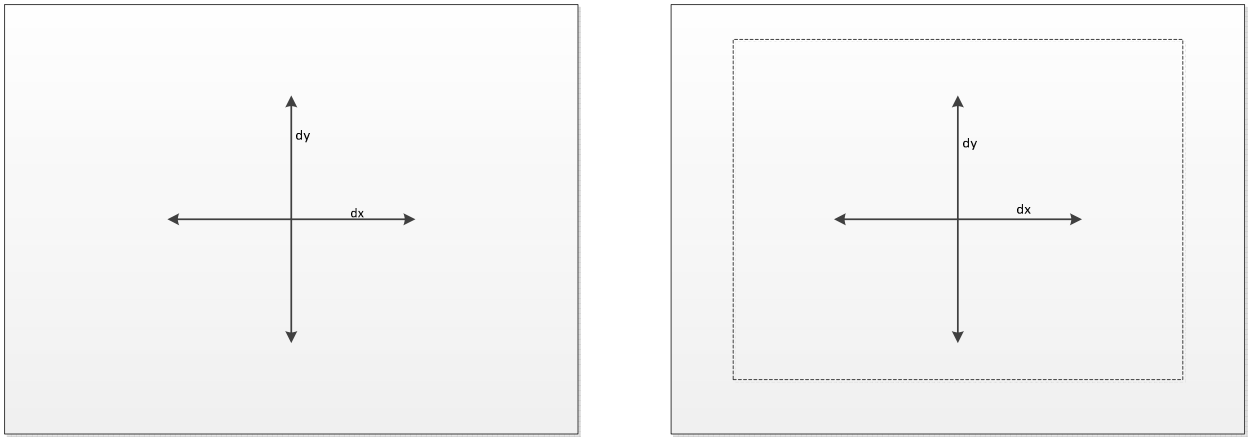


Figure 34: Two types of frame compensation

Another method that can be implemented is called mosaicking. This method continuously builds a larger image as motion occurs. This method retains no loss of information but can be deceiving when scene changes occur outside of the camera's field of view. These changes will not be reflected on the image mosaic until they become present in the camera's field of view, conversely causing the viewer to believe an object is not present when indeed it is. The mosaic technique can be seen by the aerial pictures in figure 35.

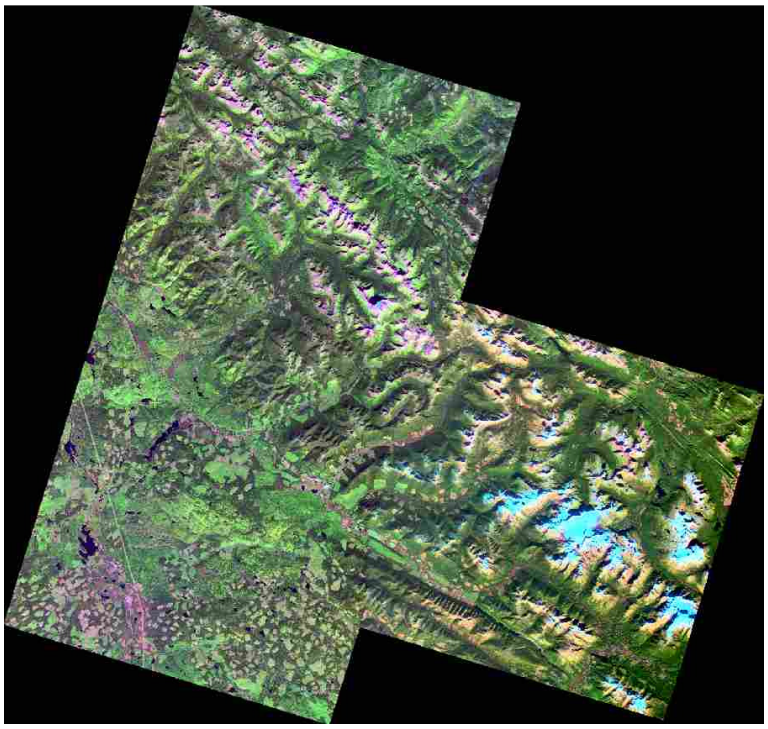

Figure 35: Mosaicking of aerial images [41]

This thesis implements the full frame compensation method because it provides a full resolution stabilized image and does not allow the misconceptions that mosaicking can provide.

# 4 Experimental Results

The performance of the image stabilizer is evaluated in three parts; performance of the visual measurement, performance of the inertial measurement and the performance of the full stabilization system. By evaluating these parts the overall error of the image stabilizer can be broken down into its individual error sources. The categorization of error can then be used in future work to improve performance of the image stabilizer.

## 4.1 Visual Measurement Evaluation

The visual measurement begins by finding and matching features in consecutive image frames. In order to evaluate this process's performance 35 seconds of video was taken while the camera remained motionless and there was no motion in the scene. The features were found and matched in consecutive image frames and their mean horizontal and vertical motion analyzed according to equation 4.1. Figure 36 shows the average motion versus time.

$$\bar{x}_{motion} = \frac{1}{N_{features}} \sum_{features} P'_x - P_x$$
$$\bar{y}_{motion} = \frac{1}{N_{features}} \sum_{features} P'_y - P_y \qquad 4.1$$

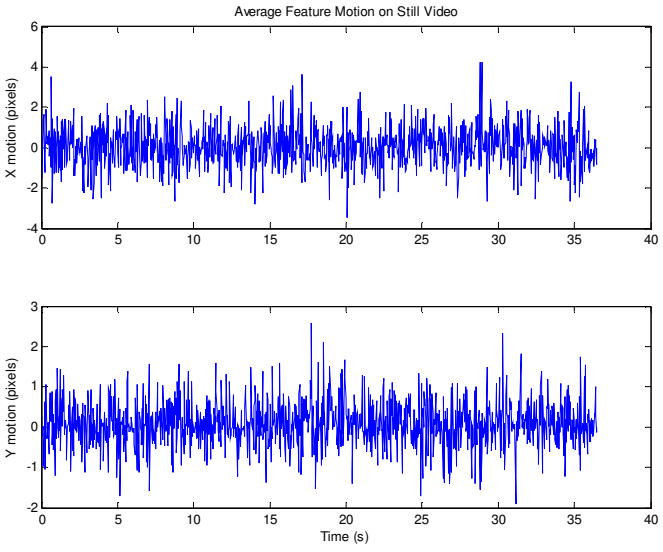


Figure 36: Average feature motion on still video

Looking at figure 36 it can be seen that there is some noise in the feature detection and matching stage of the visual measurement. This noise can be evaluated for its statistical properties as shown in figure 37.

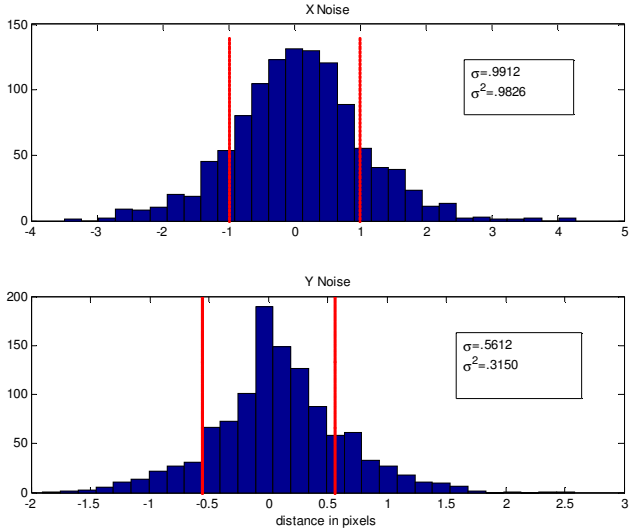


Figure 37: Histogram of feature detection noise

Looking at the histogram it can be seen that the noise follows a zero mean normal distribution. The standard deviation in the horizontal direction is twice that of the vertical counterpart and there is less variance in the vertical noise. These noise distributions are within an acceptable range but are not accounted for in this thesis, future work on the stabilizer can improve performance by accounting for this noise.

Once the visual measurement finds and matches the features it derives a global motion model that defines the total motion of the image plane. This process is evaluated by introducing synthetic video into the digital stabilizer and comparing the calculated global motion with the known motion of the synthetic video. This error measurement is described in equation 4.2, by evaluating the RMS of the error; insight can be gained as to the performance of the visual measurement.

$$error(t) = \hat{x}(t) - x(t)$$
$$where\ x(t) = \begin{bmatrix} x(t) - x(t-1) \\ y(t) - y(t-1) \\ \theta(t) - \theta(t-1) \end{bmatrix} \quad 4.2$$

The synthetic motion was made by manipulating the still video sequence from the feature matching and detection evaluation. First the middle 64% of the frame is selected; this results in a 384x576 image size. The selected portion is then randomly moved either in the horizontal, vertical or rotational direction. The ranges of motion are $(\pm 72 pixels, \pm 48 pixels, \pm 10°)$ in the horizontal, vertical and rotational directions respectively. The three motions are separated by using three synthetic videos

each containing only one motion, this provides insight into the global motion derivation as a function of the motion induced.

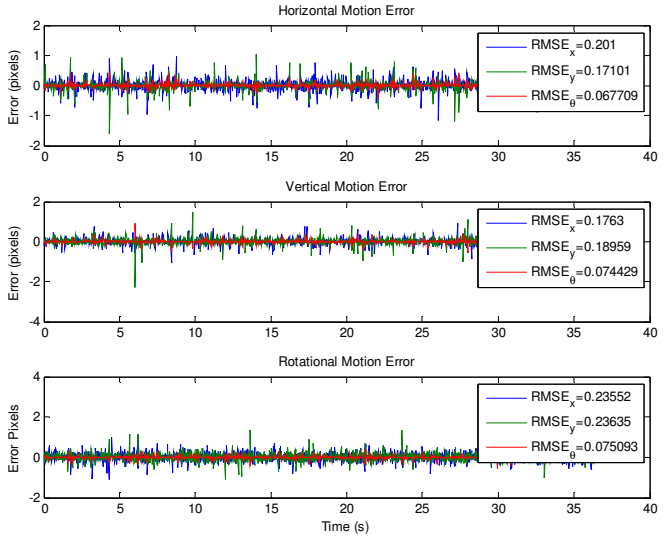


**Figure 38: Global motion error**

Figure 38 shows the error versus time and the calculated RMS values. The RMS values are well within tolerance; this shows that the RANSAC process is removing outlier feature pairs effectively. By looking at the histograms of the error in figure 39, further insight can be obtained.

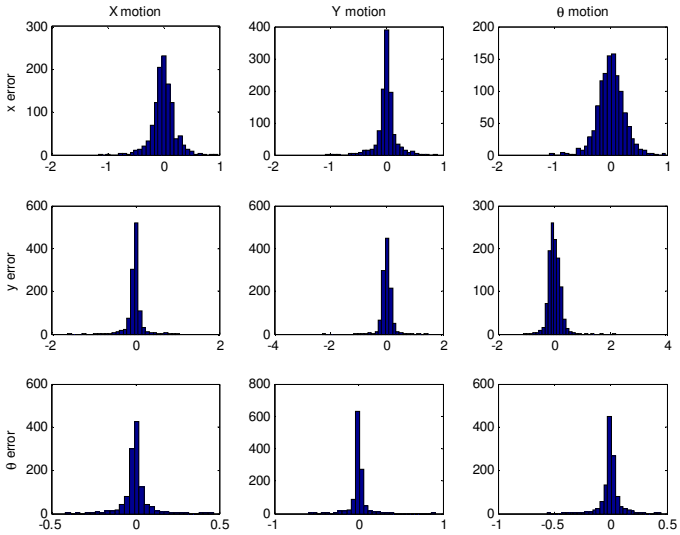


**Figure 39: Error histograms of global motion determination**

Figure 39 shows that the rotational component of the global motion contains the least amount of error and the translational errors are statistically equivalent. The translational errors are minimal when orthogonal translation occurs and rotation error is minimal when strictly rotation occurs.

## 4.2 Inertial Measurement Evaluation

There are two parts in evaluating the inertial measurement; first is to determine how well the angular position derived from the IMU relates to the motion in the image plane, and second is how much an incorrect $H$ matrix affects the inertial measurement error.

The visual measurement is used as the ground truth when evaluating the inertial measurement accuracy. The previous section showed that the visual measurement is a good approximation of the actual motion of the image plane, and thus making it a reasonable ground truth evaluation for the inertial measurement. The error is calculated according to equation 4.3 and like the evaluation of the visual measurement; the RMS of the error is used to quantify the accuracy.

$$error(t) = x_{imu}(t) - x_{dig}(t)$$

$$\begin{aligned} x_{imu}(t) &= [x_x(t) \quad x_y(t) \quad x_\theta(t)] \\ &= \begin{bmatrix} \theta_x(1) & \theta_y(1) & \theta_z(1) \\ \vdots & \vdots & \vdots \\ \theta_x(N) & \theta_y(N) & \theta_z(N) \end{bmatrix} H \end{aligned} \qquad 4.3$$

A motion platform was designed to provide the sensor rig with a controlled motion of various frequencies. The platform, shown in figure 40, uses a shaker table to move the sensor rig with a rotational motion in the vertical direction. The shaker table is driven with a sinusoidal signal and provides a stroke length of ±.25" allowing the sensor rig to have a 6° change in pitch.

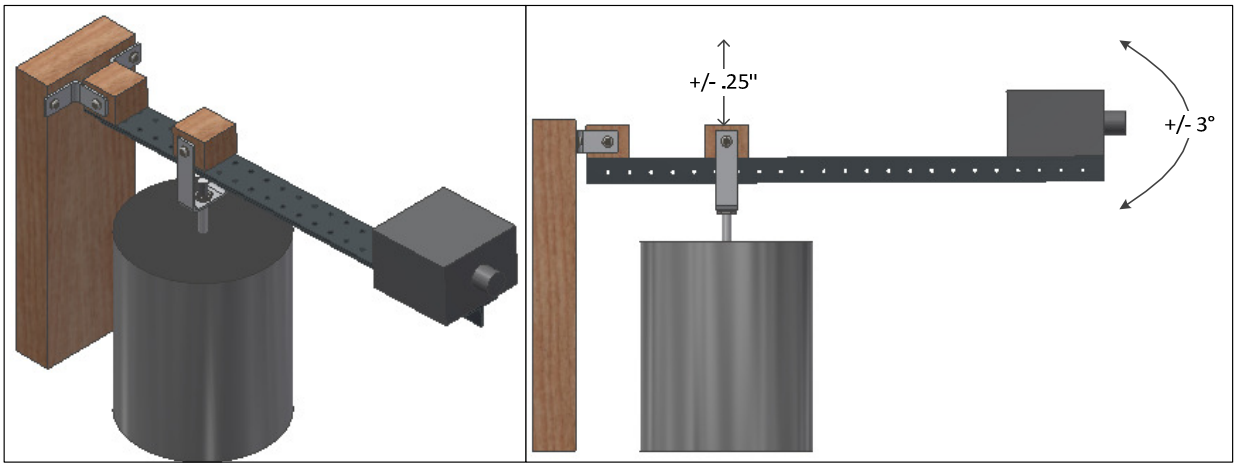


Figure 40: Motion platform

Figure 41 shows the inertial error versus time when the sensor rig is given a periodic motion of 1Hz. The RMSE is greater than the RMSE calculated for the visual measurement, which is to be expected. Table 4 shows the RMSE for all periodic motion frequencies; notice how the error increases as the periodic motion frequency increases.

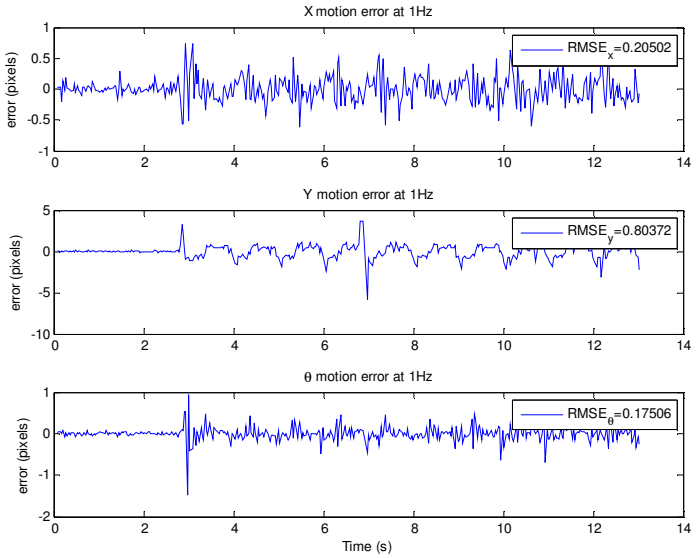


Figure 41: Mechanical measurement error at 1Hz

Table 4: Mechanical measurement RMSE for various motion frequencies

| Periodic Motion Frequency | $RMSE_x$ | $RMSE_y$ | $RMSE_\theta$ |
|---|---|---|---|
| **0.2Hz** | 0.125 | 0.446 | 0.052 |
| **0.5Hz** | 0.175 | 1.157 | 0.046 |
| **1Hz** | 0.253 | 1.595 | 0.056 |
| **2.5Hz** | 0.259 | 3.01 | 0.049 |
| **5Hz** | 1.062 | 10.626 | 0.368 |
| **10Hz** | 2.479 | 9.522 | 0.576 |

The measurement matrix $H$ is a transformation matrix that relates the rotational motion from the IMU into the image plane motion seen in the video feed. This relationship means that the accuracy of the inertial measurement relies strongly on a correctly chosen measurement matrix. The relationship can be evaluated to provide insight on how well the $H$ matrix must be calibrated for a given scene.

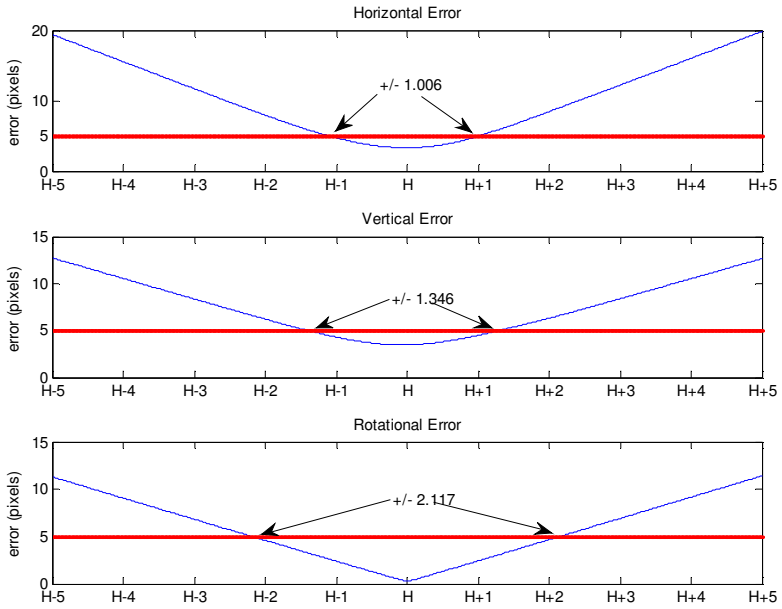


Figure 42: Mechanical measurement error as a function of H

Figure 42 shows that to be within a ±5 pixel error boundary the $H$ matrix must be correctly calibrated to (±1.006,±1.346, ±2.117) of the optimal $H_x$, $H_y$, and $H_\theta$ parameters respectively. The error boundary is application specific; the ±5 pixels error boundary was chosen because it is a reasonable error boundary for most applications that are using a sensor fusion method to provide stabilization.

## 4.3 Stabilization Evaluation

The full system performance is evaluated in two ways; the first is to measure the performance of the stabilizer by comparing how much motion is in the scene before and after stabilization. The second is to evaluate the systems capability to reduce error during dynamic scenes.

The motion in the video sequence can be quantized as the change in pixel values from one frame to another. If there is no change in the scene, either due to camera motion or moving objects, the pixel difference from frame to frame will be less than a scene where camera motion or moving objects occur. The RMS value of the pixel difference in equation 4.4 describes how much motion is in the scene and provides insight into the performance of the stabilizer.

$$RMSE(t) = \sqrt{\frac{1}{MN}\sum_{N}\sum_{M}\big(image(t-1) - image(t)\big)^2} \qquad 4.4$$

$$where\ (M,N)\ are\ the\ image's\ pixel\ height\ and\ width$$

The evaluation was performed on the video taken from the motion platform shown in figure 40. Figure 43 shows the RMSE through the video sequence when a 1Hz periodic motion is induced. Both the visual and fusion process reduce the error by a factor of half, the visual stabilization performs the best,

with an average RMSE of 1.385 pixels. The complete RMS values of the error are shown in table 5, notice that both the fusion and visual error increase as the motion increases. The visual measurement outperforms the fusion method for all motion frequencies and both stabilization methods have less error than the un-stabilized video feed. It is unfortunate that the visual stabilization method outperforms the fusion method under all frequencies; this is likely due to inaccuracies in the process model approximation. Although, you will see in the next section the fusion process delivers much better performance than visual stabilization when viewing a dynamic scene.

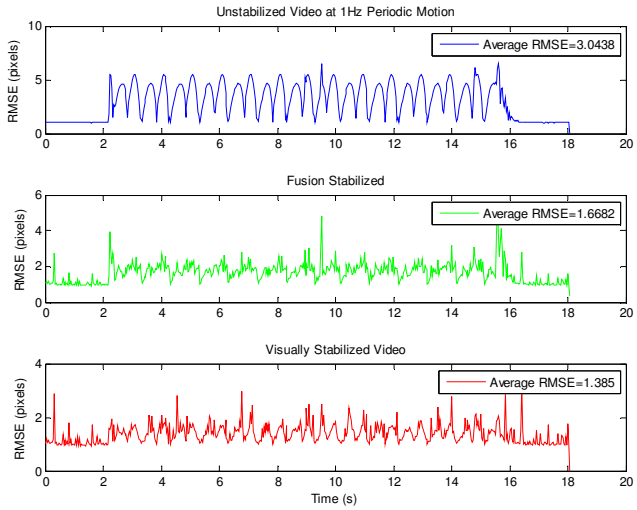


Figure 43: RMSE of un-stabilized, visually stabilized, and fusion stabilized video sequence

Table 5: Average error for all test cases

| Periodic Motion | Un-stabilized RMSE | Fusion Stabilized RMSE | Visually Stabilized RMSE |
|---|---|---|---|
| **0.2Hz** | 1.638 | 1.242 | 1.205 |
| **0.5Hz** | 2.508 | 1.3696 | 1.2929 |
| **1Hz** | 3.0438 | 1.6682 | 1.385 |
| **2.5Hz** | 4.1222 | 2.035 | 1.56 |
| **5Hz** | 5.0853 | 4.3266 | 2.1449 |
| **10Hz** | 5.3567 | 4.3605 | 2.5797 |

### 4.3.1 Dynamic Scenes

The ability of the fusion process to handle dynamic scenes is evaluated by collecting motionless video while viewing a dynamic scene. A 40 second video was taken in an office scene where people are moving about, the camera was left motionless and the error is calculated according to equation 4.5.

$$error = \sqrt{\frac{1}{N}\sum_{t}(\hat{x}(t) - 0)^2} \qquad 4.5$$

Figure 44 shows the time plot of the error for both the fusion and visual stabilization. The visual stabilization has a larger RMSE than the fusion stabilization for translational motion but the rotational motion is nearly equivalent and even a bit larger. The error can be further reduced with a better determination of a dynamic scene when calculating the measurement strength coefficient(C). Notice the strength coefficient in figure 44, because it is near zero it means the Kalman filter trusts the visual measurement more. If the detector value is forced to be unity, as shown in figure 45, the Kalman filter trusts the inertial measurement more and reduces the error an order of magnitude.

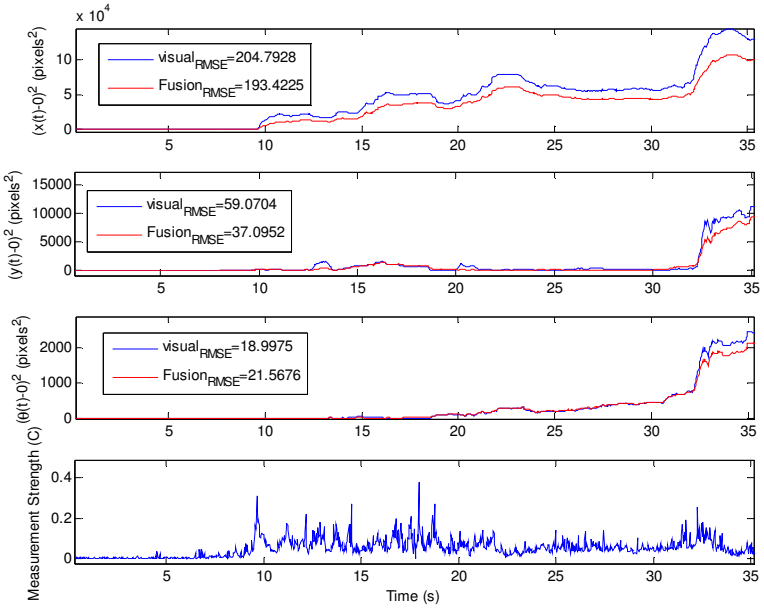


Figure 44: Stabilizer performance during a dynamic scene

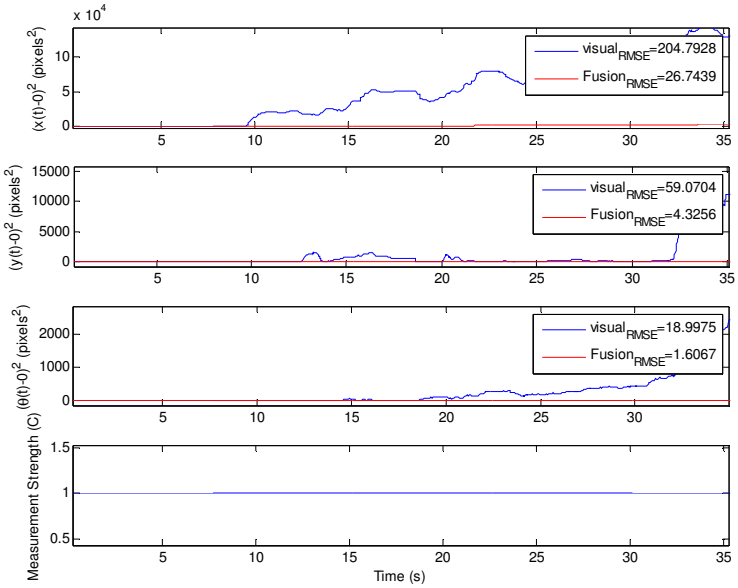


Figure 45: Stabilizer performance of a dynamic scene with unity measurement strength

## 4.4 Timing Evaluation

The computational load of the video stabilizer was evaluated to determine what aspects of the algorithm demand the most processor time. The results are shown in table 6.

Table 6: Timing chart

| Function | Percent of Time |
|---|---|
| Find Features | 46% |
| Match Features | 9.5% |
| RANSAC | 8% |
| Kalman Filter | 0.1% |
| Filter Wanted and Unwanted Motion | 0.1% |
| Stabilize Frame | 29% |

The stabilization algorithm is performed onboard a 2.2 GHz dual core processor. It can be seen that the algorithm spends most of its processing time either on finding features or stabilizing the frame. When finding features, the most computation time goes into finding local extrema in the DoG pyramid. This time can be reduced by appropriately choosing the contrast threshold that rejects unstable extrema, a lower threshold will provide more keypoints but require more computation and a high threshold will provide less keypoints but reduce the computational load. This is evaluated by finding

and matching features of the synthetic video from section 4.1 while varying the contrast threshold. The synthetic video was evaluated 50 times with a varying threshold in the range of [0 30]; the RMSE, time/frame and number of features matched are compared for each run and the results are shown in figure 46.

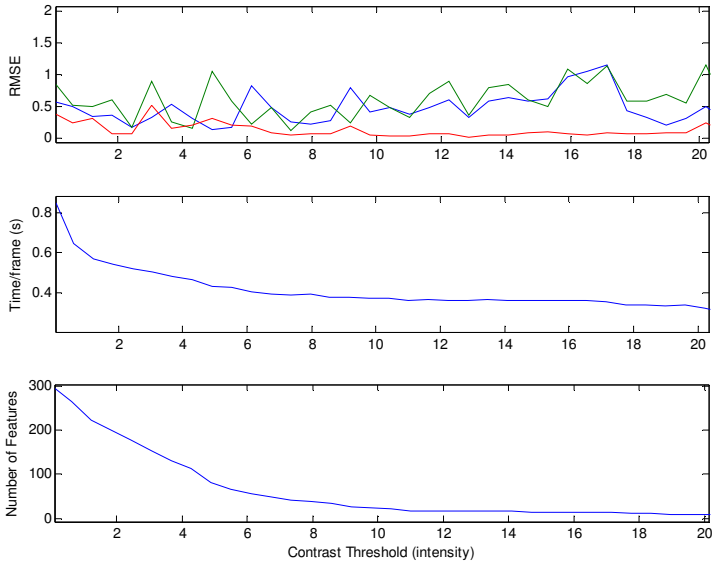


Figure 46: RMSE, time/frame and number of features matched versus contrast threshold

Figure 46 shows that the error is not significantly increased when using a higher threshold but the time it takes to find and match features per frame can be reduced by half. This threshold needs to be chosen carefully according to the stabilization application, notice how the number of features matched is also proportional to the threshold. Choosing a threshold that minimizes the number of features will cause an increase in the condition when the visual measurement cannot determine global motion from the feature matches and thus causing the fusion process to rely solely on the inertial measurement.

# 5 Future Work

In order for a digital stabilization system to work with real time systems it needs to be able to operate at a rate of at least 10Hz. The method discussed in this thesis operates at 2Hz and thus needs to be sped up. The current method can be sped up by porting the algorithm to a language better suited for image processing and using a graphics processor with dedicated memory. The Open CV toolbox developed for C++ is widely regarded as the best open source software for computer vision, porting the algorithm to C++ while using this toolbox can optimize the code and result in a faster implementation.

Using a graphics processor will also optimize the process and its dedicated memory can severely reduce the lengthy time of reading and writing image frames from the video.

The state model of the Kalman filter described in section 3.4 neglects translational measurements from the IMU, mainly because with a far field assumption their effects are negligible compared to rotational motion and by not incorporating them the state model is less complex (adding acceleration in 3 dimensions is 9 more states, and if the bias is included, 12 more states). Adding the acceleration states can marginally improve the performance of the stabilizer, but it opens the door to further implementation. Fusing a camera with 6 degrees of freedom inertial measurement can be used for localization and navigation of a robotic system [42] [43].

The results shown in section 4.3.1 show that dynamic scene performance is not significantly improved through the fusion of the IMU and that its weakness is in the characterization of the visual measurement's strength. Although the strength is defined the same way as Smith et al. in [15] and they accomplished better results, the results in this thesis show that the measurement strength coefficient needs improvement.

The fusion process allows for real time stabilization with a decrease in sampling rate during the visual measurement stage. Using different sample rates was not evaluated in this thesis and has the possibility of significantly decreasing the computational load of the stabilization system. Experimenting with decreased sample rates and evaluating the effect of the error should be performed to further classify the capabilities of this stabilization system.

# 6 Conclusion

This thesis shows the process of digital video stabilization through the fusion of feature detection and inertial measurements. The algorithm was described and results of the stabilizer performance were shown and discussed.

The visual measurement was derived through the process of SIFT feature detection. The affine model of the image plane was found through linear regression of consecutive feature matches and outliers were removed using a RANSAC process. The confidence of the visual measurement was evaluated to provide a coefficient resembling the measurement strength as a function of the tracking error and number of feature matches. The visual measurement was then evaluated to classify the error statistics present in each stage of the process.

The IMU was designed and built specific to the thesis; the circuit design, embedded algorithm and timing characteristics were discussed. The accelerometer was calibrated using the gravity vector; the cross-talk, axis misalignment and constant bias were solved using Newton's method. The gyroscope was calibrated with a rotation table and the axis misalignment and constant bias were solved through linear regression. The random walk drift of the gyroscopes was found and modeled as a first order Gauss-Markov process.

The two measurements are coupled through the use of a Kalman filter. The process model and state transition matrix were derived and the covariance matrices of the measurement update and process model were derived and discussed. The measurement matrix was derived and the algorithmic flow of the fusion process was shown and discussed.

The experimental results of the visual measurement, inertial measurement and fusion process were evaluated. The visual measurement errors include those in feature detection and matching, global motion derivation and RANSAC performance. The inertial measurement was evaluated using the visual measurement as the ground truth and the results were discussed. The fusion process is evaluated by RMSE of the frame-to-frame pixel change. The timing characteristics of the stabilization system were also evaluated and discussed.

The work on this thesis provides the fundamental foundation for a robust inertially fused image stabilizer. Each section delivers a ground up description of the process and provides the reader with a virtual roadmap on implementing the stabilization system. Through evaluation of this thesis the reader will have the necessary tools to migrate the implementation into their own specific application and improve on its performance.

# 7 References


[1] A. Broggi, P. Grisleri, T. Graf, and M. Meinecke, "A software video stabilization system for automotive oriented applications," in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, 2005, pp. 2760-2764 Vol. 5.

[2] N. Yazdi, F. Ayazi, and K. Najafi, "Micromachined inertial sensors," *Proceedings of the IEEE,* vol. 86, pp. 1640-1659, 1998.

[3] H. Seidel, H. Riedel, R. Kolbeck, G. Mück, W. Kupke, and M. Königer, "Capacitive silicon accelerometer with highly symmetrical design," *Sensors and Actuators A: Physical,* vol. 21, pp. 312-315, 1990.

[4] S. Sherman, W. Tsang, T. Core, R. Payne, D. Quinn, K. H. L. Chau, J. Farash, and S. Baum, "A low cost monolithic accelerometer; product/technology update," in *Electron Devices Meeting, 1992. IEDM'92. Technical Digest., International*, 1992, pp. 501-504.

[5] A. M. Shkel, C. Acar, and C. Painter, "Two types of micromachined vibratory gyroscopes," in *Sensors, 2005 IEEE*, 2005, p. 6 pp.

[6] S. Bayrak, "Video Stabilization: Digital and Mechanical Approaches," Master of Science in Electrical and Electronics Engineering, Graduate School of Natural and Applied Sciences, Middle East Technical University, 2008.

[7] B. Cardani, "Optical image stabilization for digital cameras," *Control Systems, IEEE,* vol. 26, pp. 21-22, 2006.

[8] Available: http://www.canon.com/bctv/faq/optis.html

[9] S. N. David Sachs, Daniel Goehl, "Image Stabilization Technology Overview ", ed.

[10] K. Sato, S. Ishizuka, A. Nikami, and M. Sato, "Control techniques for optical image stabilizing system," *Consumer Electronics, IEEE Transactions on,* vol. 39, pp. 461-466, 1993.

[11] S. Battiato, G. Gallo, G. Puglisi, and S. Scellato, "SIFT Features Tracking for Video Stabilization," in *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, 2007, pp. 825-830.

[12] K. L. Veon, "Video Stabilization Using SIFT Features, Fuzzy Clustering, and Kalman Filtering," Master of Science, Computer Engineering, University of Denver, 2011.

[13] H. Keng-Yen, T. Yi-Min, T. Chih-Chung, and C. Liang-Gee, "Feature-based video stabilization for vehicular applications," in *Consumer Electronics (ISCE), 2010 IEEE 14th International Symposium on*, 2010, pp. 1-2.

[14] A. Censi, A. Fusiello, and V. Roberto, "Image stabilization by features tracking," in *Image Analysis and Processing, 1999. Proceedings. International Conference on*, 1999, pp. 665-667.

[15] M. J. Smith, A. Boxerbaum, G. L. Peterson, and R. D. Quinn, "Electronic image stabilization using optical flow with inertial fusion," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 1146-1153.

[16] B. Pinto and P. R. Anurenjan, "Video stabilization using Speeded Up Robust Features," in *Communications and Signal Processing (ICCSP), 2011 International Conference on*, 2011, pp. 527-531.

[17] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," presented at the Proceedings of The Fourth Alvey Vision Conference, 1988.

[18] B. D. L. a. T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proceedings of Imaging Understanding Workshop,* pp. 121-130, 1981.

[19] A. Rav-Acha and S. Peleg, "Lucas-Kanade without Iterative Warping," in *Image Processing, 2006 IEEE International Conference on*, 2006, pp. 1097-1100.

[20] S. Jianbo and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, 1994, pp. 593-600.
[21] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. C. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences,* vol. 454, pp. 903-995, 1998.
[22] K. Ioannidis and I. Andreadis, "A Digital Image Stabilization Method Based on the Hilbert & Huang Transform," *Instrumentation and Measurement, IEEE Transactions on,* vol. PP, pp. 1-12, 2012.
[23] J. S. Jin, Z. Zhu, and G. Xu, "Digital Video Sequence Stabilization Based on 2.5D Motion Estimation and Inertial Motion Filtering," *Real-Time Imaging,* vol. 7, pp. 357-365, 2001.
[24] S. Wu, D. C. Zhang, Y. Zhang, J. Basso, and M. Melle, "Adaptive smoothing in real-time image stabilization," in *Proc. of SPIE Vol*, 2012, pp. 83990L-1.
[25] S. B. Balakirsky and R. Chellappa, "Performance characterization of image stabilization algorithms," in *Image Processing, 1996. Proceedings., International Conference on*, 1996, pp. 413-416 vol.2.
[26] S. Balakirsky, P. David, P. Emmerman, F. Fisher, and P. Gaylord, "Semi-autonomous mobile target engagement system," in *Proc. Symposium of the Association for Unmanned Vehicle Systems*, 1993.
[27] C. Morimoto and R. Chellappa, "Fast Electronic Digital Image Stabilization for Off-Road Navigation," *Real-Time Imaging,* vol. 2, pp. 285-296, 1996.
[28] C. Zhang, P. Chockalingam, A. Kumar, P. Burt, and A. Lakshmikumar, "Qualitative assessment of video stabilization and mosaicking systems," in *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*, 2008, pp. 1-6.
[29] (2012). Available: http://www.ptgrey.com/products/fireflymv/fireflymv_firewire_cmos_camera.asp
[30] A. Vedaldi and B. Fulkerson, "VLFeat," 0.9.13 ed, 2007.
[31] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision,* vol. 60, pp. 91-110, 2004.
[32] J. Koenderink and A. van Doorn, "Representation of local geometry in the visual system," *Biological Cybernetics,* vol. 55, pp. 367-375, 1987.
[33] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, 1997, pp. 1000-1006.
[34] R. Szeliski, "Image alignment and stitching: A tutorial," *Foundations and Trends® in Computer Graphics and Vision,* vol. 2, pp. 1-104, 2006.
[35] W. Flenniken, J. Wall, and D. Bevly, "Characterization of various IMU error sources and the effect on navigation performance," in *ION GNSS*, 2005, pp. 13-16.
[36] J. D. Hol, T. B. Schön, and F. Gustafsson, "Modeling and calibration of inertial and vision sensors," *The international journal of robotics research,* vol. 29, pp. 231-244, 2010.
[37] T. Mineta, S. Kobayashi, Y. Watanabe, S. Kanauchi, I. Nakagawa, E. Suganuma, and M. Esashi, "Three-axis capacitive accelerometer with uniform axial sensitivities," *Journal of Micromechanics and Microengineering,* vol. 6, p. 431, 1999.
[38] I. Frosio, F. Pedersini, and N. A. Borghese, "Autocalibration of MEMS accelerometers," *Instrumentation and Measurement, IEEE Transactions on,* vol. 58, pp. 2034-2041, 2009.
[39] K. Madsen, H. Bruun, and O. Tingleff, "Methods for non-linear least squares problems," 1999.
[40] Invensense, "MPU-6000 and MPU-6050 Product Specification Revision 3.2," ed, 2011.

[41] R. Steenweg. (2008). *Alpine Habitat Selection by Wolves and Image Mosaicking in the Parsnip River Area, BC*. Available: http://www.gis.unbc.ca/courses/geog432/projects/2008/steenweg/index.htm

[42] G. Nützi, D. Scaramuzza, S. Weiss, and R. Siegwart, "Fusion of IMU and Vision for Absolute Scale Estimation."

[43] A. P. Aguiar and J. P. Hespanha, "State estimation for systems with implicit outputs for the integration of vision and inertial sensors," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, 2005, pp. 621-626.