2013

Vision-based Sensing And Optimal Control For Low-cost And Small Satellite Platforms

Bradley Sease
University of Central Florida

VISION-BASED SENSING AND OPTIMAL CONTROL FOR LOW-COST AND SMALL SATELLITE PLATFORMS

by

BRADLEY SEASE
B.S. University of Central Florida, 2011

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science
in the Department of Mechanical and Aerospace Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2013

ABSTRACT

Current trends in spacecraft are leading to smaller, more inexpensive options whenever possible. This shift has been primarily pursued for the opportunity to open a new frontier for technologies with a small financial obligation. Limited power, processing, pointing, and communication capabilities are all common issues which must be considered when miniaturizing systems and implementing low-cost components. This thesis addresses some of these concerns by applying two methods, in attitude estimation and control. Additionally, these methods are not restricted to only small, inexpensive satellites, but offer a benefit to large-scale spacecraft as well.

First, star cameras are examined for the tendency to generate streaked star images during maneuvers. This issue also comes into play when pointing capabilities and camera hardware quality are low, as is often the case in small, budget-constrained spacecraft. When pointing capabilities are low, small residual velocities can cause movement of the stars in the focal plane during an exposure, causing them to streak across the image. Additionally, if the camera quality is low, longer exposures may be required to gather sufficient light from a star, further contributing to streaking. Rather than improving the pointing or hardware directly, an algorithm is presented to retrieve and utilize the endpoints of streaked stars to provide feedback where traditional methods do not. This allows precise attitude and angular rate estimates to be derived from an image which, with traditional methods, would return large attitude and rate error. Simulation results are presented which demonstrate endpoint error of approximately half a pixel and rate estimates within 2% of the true angular velocity. Three methods are also considered to remove overlapping star streaks and resident space objects from images to improve performance of both attitude and rate estimates. Results from a large-scale Monte Carlo simulation are presented in order to characterize the performance of the method.

Additionally, a rapid optimal attitude guidance method is experimentally validated in a ground-based, pico-scale satellite test bed. Fast slewing performance is demonstrated for an incremental step maneuver with low average power consumption. Though the focus of this thesis is primarily on increasing the capabilities of small, inexpensive spacecraft, the methods discussed have the potential to increase the capabilities of current and future large-scale missions as well.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Yunjun Xu, for his support throughout the entirety of my Master's program. His guidance from undergraduate research to a graduate career has enabled all of the work presented here and has provided a foundation for the remainder of my academic career.

I would also like to acknowledge the two additional members of my advisory committee – Dr. Kuo-Chi Lin and Dr. Eric Todd Bradley – for their support throughout the process.

I am grateful to Dr. Brien Flewelling for his support and direction through two summer internships as well as the invaluable education he provided in image processing techniques and star camera algorithms during that time.

Additionally, I would like to acknowledge the Air Force Research Laboratory for use of their facilities as well as the support and guidance provided by all of the researchers through two summer internships. I specifically want to thank Dr. Fred Leve for initially accepting me into the summer Space Scholars internship and enabling me to establish a continuing relationship with AFRL.

Finally, I would like to recognize Dr. Daniele Mortari for providing the star tracking code which inspired and enabled the long-integration star tracking methods presented here.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Small Satellite Background & Technical Challenges

Pico- and nano-scale satellites are attractive due to their extremely low cost and increasing versatility for scientific missions in low-Earth orbit. A typical pico-scale satellite follows the 10 cm cube form factor and weighs in under 1 kg. Multiple cubes may be combined to form a larger nano-scale satellite which is generally within a weight limit of 4 kg [1]. Some systems, however, extend up to 6 units, with a mass under 8 kg. The benefit of adhering to this form factor is the availability of a standard deployment system with flight heritage, known as the Poly-Picosatellite Orbital Deployer (P-POD) [2]. The P-POD deployment system was developed at California Polytechnic University with the aim of standardizing the launch of pico- and nano-scale satellites. This deployer allows small spacecraft to ride as passengers on a larger primary mission without the associated cost of developing a compatible deployment system for each mission [3].

Small satellite missions are becoming increasingly common for university scientific experiments. Past missions have included earthquake detection [4], imaging [5], and biological research in micro gravity [6]. With the increasing availability of commercial off-the-shelf (COTS) equipment, it is becoming simpler and cheaper to use a small satellite for university research and in developing nations [7]. Additionally, active control methods employed in small satellites are rapidly becoming more precise and increasingly capable of performing imaging and Earth observation missions [8].

The limited volume of these small pico- and nano-scale satellites necessarily leads to tight restrictions on the power available to the systems onboard. Past missions have varied from 10 mW to nearly 7 W of available power [9]. In general, small satellites have averaged a power generation potential of approximately 1 $^{W}/_{kg}$. Power generation is almost always achieved through the use

of surface mounted solar panels — which are seen in approximately 85 % of small satellites since 1997 — however only 16 % employ deployable solar panels [9]. Due to the small physical size, a satellite following the CubeSat specification will only have a maximum area of $0.01\ m^2$ available for pico-scale and up to $0.03\ m^2$ for nano-scale with no deployable solar panels. Further considering the fact that solar panel peak efficiencies are in the range of 30 % and additional losses in transmission and storage, the power constraints on small satellites become readily apparent.

Currently, only about 40% of small satellites have employed active attitude control [9]. This is due to the power, volume, and mass demands of such a system in addition to the lack of precise pointing options for pico- and nano-scale satellites. Missions which have used some form of active control have yet to improve pointing capabilities beyond 1 degree accuracy [9, 10]. Some COTS equipment is planned to reach beyond 1 degree of accuracy, however there are no solutions with flight heritage.

General Spacecraft Limitations

General spacecraft limitations addressed here primarily refer to the operation of star trackers for attitude estimation. Star trackers are common camera-based sensors which are capable of providing extremely precise attitude feedback in the typically measured in the arcsecond range [11, 12, 13, 14]. The operating principles behind these sensors relies primarily on digitally produced images of stars. These stars are de-focused slightly in order to spread the signal across multiple pixels, and a centroiding process is conducted on the resulting image [15]. Centroiding essentially provides a center-of-mass computation for each star in order to produce precise, sub-pixel locations of each within the image. Discovered stars are matched to a catalog of known star locations on the celestial sphere [16, 17, 18]. Attitude estimation may then be performed on the set of discovered and reference locations to determine an optimal attitude estimate. The attitude

estimation problem is known as Wahba's Problem [19] and there exist several optimal solutions [20, 21, 22].

One drawback of star trackers is the fact that they are limited by the maximum angular rate at which they are capable of operating [13]. Due to the nature of the centroiding procedure, the center point of a star signal becomes less precise as the signal is spread asymmetrically across several pixels. This spread, referred to here as a streak, is due to the integrative nature of cameras. In order to generate a useful signal for attitude determination, the camera must collect light over time. The time required for this process means that any movement during the exposure will necessarily cause the star to move across the focal plane. The resulting star image, then, will be streaked according to the direction of rotation. For small angular velocities, this may not be an issues, however for large angular rates, small fields of view, and long integration times, this can lead to significant distortion of the star image.

Lower quality cameras also have the tendency to produce streaked images. For example, small satellites often do not have the available volume to contain large-aperture cameras, resulting in collection of less light during an exposure. Consequently, smaller aperture cameras must integrate for longer durations to achieve similar signal strength to that of a large aperture camera [11]. Long integration times combined with small residual angular velocities spread the signal of a single star over several pixels, reducing accuracy of the standard methods.

This streaking distortion comes into play during commanded maneuvers. Unless the maneuver is maintained within a limited velocity profile, which may not fit within the mission requirements of the spacecraft, it follows that there will be a reduction of accuracy or even the loss of feedback altogether. In such a case, a secondary sensor suite or a numerical propagation of the last known state becomes necessary.

Thesis Outline and Contributions

The chapters of this thesis are organized as follows.

Chapter two details a framework for a realistic star camera model built on a simple pin-hole camera approximation. A widely available star catalog is included in the implementation as a source for actual star location and visible brightness data. Realistic star intensities are extrapolated from published intensity values. Distortions and noise particular to charge-coupled device (CCD) sensors are implemented. Secondary, resident space objects (RSOs) are implemented within the images through consideration of actual orbital dynamics for both the camera spacecraft and the RSO. The RSO implementation serves to provide a foundation for part of the work in chapter four. Finally, the quaternion kinematics of the camera-equipped spacecraft are applied to produce streaked star camera images. This camera model is the foundation for the work discussed in chapters three and four.

In chapter three an algorithm is presented for providing attitude and rate feedback from streaked star camera images by utilizing traditional image processing techniques. Images are segmented to specifically locate regions within the image containing star streaks. These regions of interest are then examined for "corner-like" features, which refers to locations within the image which contain a strong gradient in two directions. Endpoints are discovered, localized, and traditional star identification and attitude estimation methods are applied. A large-scale Monte Carlo simulation is presented which characterizes the performance over a broad range of camera configurations.

Chapter four explores the work of the previous chapter in further detail. A rate-only formulation of the previous algorithm is discussed. Additionally, an automated method for identifying overlapping star streaks and RSOs within the image is presented. Based on the dynamics of the camera-RSO system, two foundational methods are proposed as indicators for corrupted or non-star data within

an image. A third method is proposed which combines the two base methods into a single operation. Simulation results are shown for rate estimation in an RSO-tracking scenario and a 10-orbit Monte Carlo simulation is presented to verify and characterize the expected performance.

The fifth chapter addresses the power constraints of pico- and nano-scale satellites with an optimal path planning algorithm for slewing maneuvers. A path generation control architecture is derived with the aim of reducing overall power consumption of the maneuver. A bio-inspired motion strategy is implemented with the purpose of reducing overall computation time required for the optimization. A linear quadratic tracking controller is developed in order to optimally track the pre-generated path. Simulation results are discussed and the complete system is experimentally verified on a pico-scale satellite prototype suspended within a Helmholtz coil test bed. Results and are discussed and performance metrics are provided.

Finally, the thesis is concluded with a summary and discussion of potential future work which could improve these methods further.

CHAPTER 2: HIGH-FIDELITY STAR CAMERA MODEL

In order to develop and verify the algorithms in the following chapters, a star camera model is first developed. This model generates realistic images of actual stars, including streaking dynamics, noise characteristics, and resident space objects (RSOs).

Pin-Hole Camera Model

In order to generate an image, knowledge of the stars within the celestial sphere is required. Conveniently, such data has been precisely collected and made freely available in several forms. The two primary sources currently in use are the Tycho star catalog [23] and the Hipparcos star catalog [24]. The most recent version of the Tycho catalog, Tycho-2, contains precise measurements of nearly 2.5 million stars, while the Hipparcos catalog pinpoints approximately $120,000$ stars. For simplicity, a subset of the Hipparcos star catalog is used, containing unit vectors corresponding to the location of $7,000$ stars in the Earth-Centered Inertial (ECI) coordinate frame with visible magnitudes ranging from -1.4 to 8.5. For reference, the faintest star visible to the human eye is a magnitude of approximately 5.0, with lower values corresponding to brighter stars [11].

The subset of the star catalogue was chosen to provide a homogenous distribution of stars regardless of orientation. The full catalog may be used, however image generation time increases considerably with the complete catalogue. In order to generate an image from this catalog, each unit vector is considered individually. The Collinearity equation maps points in 3-dimensional space through a common focal point onto a 2-dimensional plane.

Figure 2.1: Full-sky image of the Tycho-2 star catalogue.

Equation 2.1 is referred to as the "Pinhole Camera" model [25].

$$\begin{aligned} x &= -f\frac{R_{11}\left(X-X_0\right)+R_{21}\left(Y-Y_0\right)+R_{31}\left(Z-Z_0\right)}{R_{13}\left(X-X_0\right)+R_{23}\left(Y-Y_0\right)+R_{33}\left(Z-Z_0\right)} \\ y &= -f\frac{R_{12}\left(X-X_0\right)+R_{22}\left(Y-Y_0\right)+R_{32}\left(Z-Z_0\right)}{R_{13}\left(X-X_0\right)+R_{23}\left(Y-Y_0\right)+R_{33}\left(Z-Z_0\right)} \end{aligned} \tag{2.1}$$

Here, f is the focal length of the camera, R refers to the direction cosine matrix (DCM) representing the attitude of the camera in ECI, and (X, Y, Z) refers to the unit vector describing a particular star in ECI. (X_0, Y_0, Z_0), in general, is the absolute location of the camera; in order to be compatible with the star catalogue, however, this is taken as $(0, 0, 0)$. This fixes the camera at the center of the celestial sphere at all times, with only freedom in rotation. Since the stars are very distant from the camera, any changes of the apparent star locations due to movement within the solar system are negligible. x and y are the corresponding coordinates in the 2-D focal plane.

One issue with a simple application of the Collinearity equation is that Equation 2.1 contains no discrimination of stars which lie behind the camera. All stars are projected to the image plane,

meaning that stars directly behind the camera will also fall within the produced image. In order to prevent this, a preliminary step is performed to remove stars well beyond the field of view. The complete star map is rotated into the camera frame with

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}_{Cam} = R \cdot \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}_{ECI} \tag{2.2}$$

for $i = 1 \dots \mathrm{N}$ where N is the number of stars contained in the catalogue. Stars with $Z_{i,Cam} < 0$ are removed as they are physically behind the camera and cannot appear in the resulting image. This has the added benefit of reducing overall computation time.

The Collinearity equation is then applied to each of the remaining star locations to produce the set of camera-plane coordinates, (x_i, y_i). Each set of star coordinates must then be converted from physical dimensions to pixel space by dividing by the physical pixel size characteristic to the camera. Each coordinate is additionally shifted so that the origin corresponds top-left corner of the image.

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix}' = \begin{bmatrix} {}^{x_i}/_{S_x} \\ {}^{y_i}/_{S_y} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} Resolution_x + 1 \\ Resolution_y + 1 \end{bmatrix} \tag{2.3}$$

S_x and S_y are the dimensions of the physical CCD pixels within the camera. These values are derived from the physical camera parameters through the relations in 2.4.

$$\begin{aligned} S_x &= 2f \frac{\tan\left(0.5 \cdot FoV_x\right)}{Resolution_x} \\ S_y &= 2f \frac{\tan\left(0.5 \cdot FoV_y\right)}{Resolution_y} \end{aligned} \tag{2.4}$$

Here, FoV_x and FoV_y correspond to the angular field of view in the x and y axes, respectively. Coordinates beyond the bounds of the camera frame are removed, leaving only coordinates which

fall within the image plane.

Since pixel locations are limited to discrete integer locations, a further step is required to convert from the coordinates which do not perfectly fall within a single pixel. A simple nearest-neighbor approach may be used, where the coordinate is rounded to the nearest pixel, however significant positional error will be introduced. A bi-linear interpolation is implemented here in order to place stars within the image to sub-pixel precision. This process essentially spreads the intensity of the point source across the neighboring pixels, proportional to the coordinate's position among them. For example, for the coordinate $(1.5, 1)$, which falls halfway between $(1, 1)$ and $(2, 1)$, the intensity would be divided evenly between the pixels located at $(1, 1)$ and $(2, 1)$. For the coordinate $(1.25, 1)$, however, 75% of the intensity would fall into the $(1, 1)$ pixel, and 25% would fall into the pixel at $(2, 1)$. The complete relations are shown below. First, two intermediate values are calculated to simplify the subsequent equations.

$$\begin{aligned} A_i &= x_i - floor\,(x_i) \\ B_i &= y_i - floor\,(y_i) \end{aligned} \tag{2.5}$$

A_i and B_i represent only the fractional component of the coordinates x_i and y_i, respectively. The intensities corresponding to the 2×2 pixel area surrounding the coordinate is outlined below. The net intensity added to the image in this process is equal to the original intensity, I_i.

$$\begin{aligned} M_{0,0} &= I_i\,(1 - A_i)\,(1 - B_i) \\ M_{0,1} &= I_i A_i\,(1 - B_i) \\ M_{1,0} &= I_i B_i\,(1 - A_i) \\ M_{1,1} &= I_i A_i B_i \end{aligned} \tag{2.6}$$

Within the full-frame image, $M_{0,0}$ corresponds to the coordinate $(floor(x_i), floor(y_i))$. The in-

tensities, I_i, are extrapolated from a published value for a magnitude 0 star [11]. Intensities are measured in photoelectrons, e^-, per second of exposure, per square meter of exposed aperture.

$$I_0 = 19,000 \frac{e^-}{s \cdot m^2} \tag{2.7}$$

To extrapolate this value to produce the intensity corresponding to any other visible magnitude, M_v, the following equation from [11] is used.

$$I\left(M_v\right) = I_0 \frac{1}{2.5^{M_v}} TA \tag{2.8}$$

T is the exposure time and A is the aperture area. The visible magnitudes represent a logarithmic scale, with smaller values corresponding to brighter stars. For example, a star of magnitude -1 is 2.5 times brighter than a magnitude 0 star, with $I\left(-1\right) = 47500\ {}^{e^-}\!/_{s \cdot m^2}$. A sample single-star image at this step is shown in Figure 2.2. The image has been inverted from the original.

Figure 2.2: Inverted single-star image following bilinear interpolation.

In order to produce a more realistic star camera image, it is necessary to also simulate slight blurring effects present due to imperfections in the lens geometry. In fact, this effect is often seen as beneficial in traditional star cameras as it spreads the point sources over multiple pixels, allowing a more precise centroid to be calculated. This, in turn, produces a more precise image coordinate for the measured star [15]. Here, this effect is achieved through a 2-dimensional convolution of a point spread function (PSF) with the raw, full-frame image. Specifically, a Gaussian PSF is used. The parameters of interest for this type of operation are the size of the PSF, measured in pixels

and the standard deviation, σ, of the kernel. Any amount of blurring can be used, depending on the application and the characteristics of the chosen camera, so specific parameters will not be discussed here. A sample 3×3 Gaussian PSF with $\sigma = 1$ is shown below.

$$PSF = \begin{pmatrix} 0.0751 & 0.1238 & 0.0751 \\ 0.1238 & 0.2041 & 0.1238 \\ 0.0751 & 0.1238 & 0.0751 \end{pmatrix} \tag{2.9}$$

The image from Figure 2.2 is seen again in Figure 2.3 following the application of the Gaussian PSF above. The effect is a distribution of the same intensity contained in the original four pixels over a larger pixel area. For this reason, the intensity in any single pixel generally decreases.

Figure 2.3: Inverted single-star image following convolution with a 3×3 Gaussian PSF.

To complete the star camera model, realistic noise characteristics are implemented. Three types of noise are considered: read noise, shot noise, and dark current [26]. Each type of noise will be discussed individually. The first type, read noise (RN), is noise due to transferring data from the detector within the camera. Values for read noise are measured in photoelectrons and can be found in technical specifications for the specific camera in use. Shot noise (SN) is proportional to the intensities present within the image – brighter pixels generate slightly more noise than dimmer

pixels. For the purposes outlined here, the relation in Equation 2.10 is used.

$$SN = \sqrt{Signal} \tag{2.10}$$

Dark current (DC) noise is due to leakage current in the circuitry of the camera. This type of noise is proportional to the exposure time. An image with a longer exposure will contain more noise than a similar image taken with a shorter exposure. Dark current noise is measured in photoelectrons per second.

A complete measure of the total noise present in the system is available through

$$Noise = \sqrt{RN^2 + SN + DC \cdot T} \tag{2.11}$$

where T is the exposure time of the image. The complete noise effect is added in two steps: (i) shot noise is calculated and multiplied by a uniformly-distributed random value between 1 and 0 for each pixel; (ii) the magnitude remaining noise is calculated through $Noise = \sqrt{RN^2 + DC \cdot T}$ and applied to the entire image as uniformly-distributed random noise.

Finally, to complete the resulting image, a conversion from photoelectrons, e^-, to the non-dimensional units (NDUs) which measure pixel intensity in an image is performed. The specific conversion is a characteristic of the camera hardware, referred to as quantum efficiency (QE) [26]. Quantum efficiency is measured in e^- per NDU; conversion is achieved by simply dividing the intensity value of each individual pixel by QE. A saturation is then applied to all pixels, again based on the camera characteristics. Additional image corruption due to over-saturation is possible, however it is not considered here. A complete full-frame image including all of the features discussed in this section is shown in Figure 2.4.

Figure 2.4: Complete full-frame simulated star camera image for an arbitrary parameter set.

A summary of the complete process for generating a single-attitude star image is shown below.

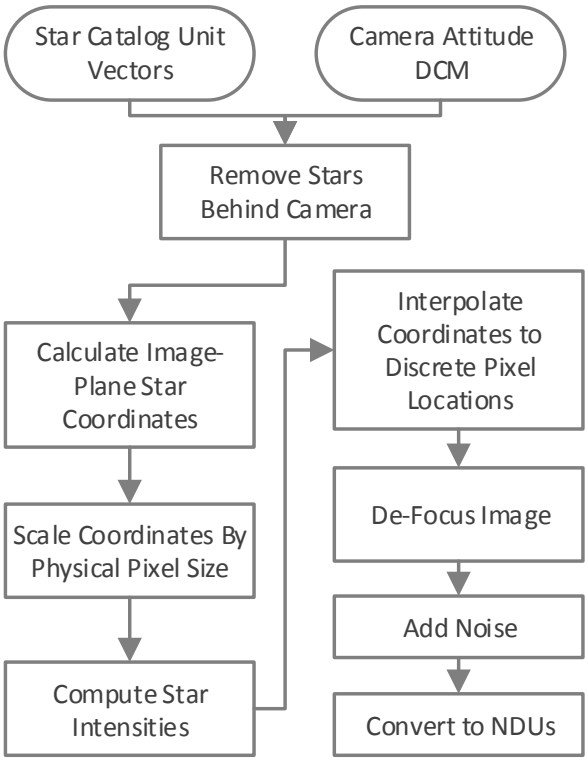

Figure 2.5: Complete process for generation of a single star image.

Streaking Dynamics

Streaking is an effect present in images from a camera undergoing motion relative to its target or rotation. In order to implement this effect, the kinematics of a rotating rigid body are considered. It is worth noting that only the quaternion kinematics are considered here. Therefore, this model is only valid for bodies undergoing very small accelerations, which will not produce significant variations within the resulting image. More significant accelerations will, in reality, lead to variations in brightness and complex geometries within single streaks. The quaternion kinematics for a rigid body are detailed below [27].

$$\dot{\vec{q}} = \frac{1}{2} \begin{pmatrix} \tilde{\omega} & \vec{\omega} \\ -\vec{\omega}^T & 0 \end{pmatrix} \vec{q} \tag{2.12}$$

Here, $\vec{\omega}$ is the angular rate vector, $\tilde{\omega}$ is the skew-symmetric matrix, and $\vec{q}$ is the quaternion attitude. As stated before, $\vec{\omega}$ is considered to be constant or near constant. The kinematics here are considered in ICRS to more easily incorporate with the star catalogue data. To produce an image at a given moment in time, t_i, the initial attitude and angular rate are considered. The kinematics are integrated from these initial conditions with time steps, T_s, to a final time, $t_f = t_i + T$, where T is the exposure time of the camera. Still images are generated for each attitude in the complete attitude history, $(\vec{q}(t_i) \dots \vec{q}(t_f))$. For each individual image, the intensities for each star are calculate from Equation 2.8 as detailed in the previous section, however the exposure time, T, is replaced with the step time, T_s. Therefore, a star intensity present in a still image is spread evenly across several pixels in a streaked image. The complete set of images is then summed to produce a complete streaked star image.

For larger simulated image, this image generation method can quickly become computationally intensive. In past simulations, generation of a 1024×1024 image can take several seconds. To somewhat reduce the required computational power, the "de-focusing" step discussed in the pre-

vious section is postponed until the set of images has been summed. The convolution is applied to the entire streaked image, requiring only one convolution rather than one for each time step. Additionally, parallelizing the computations has provided a significant benefit, as several images can be generated simultaneously.

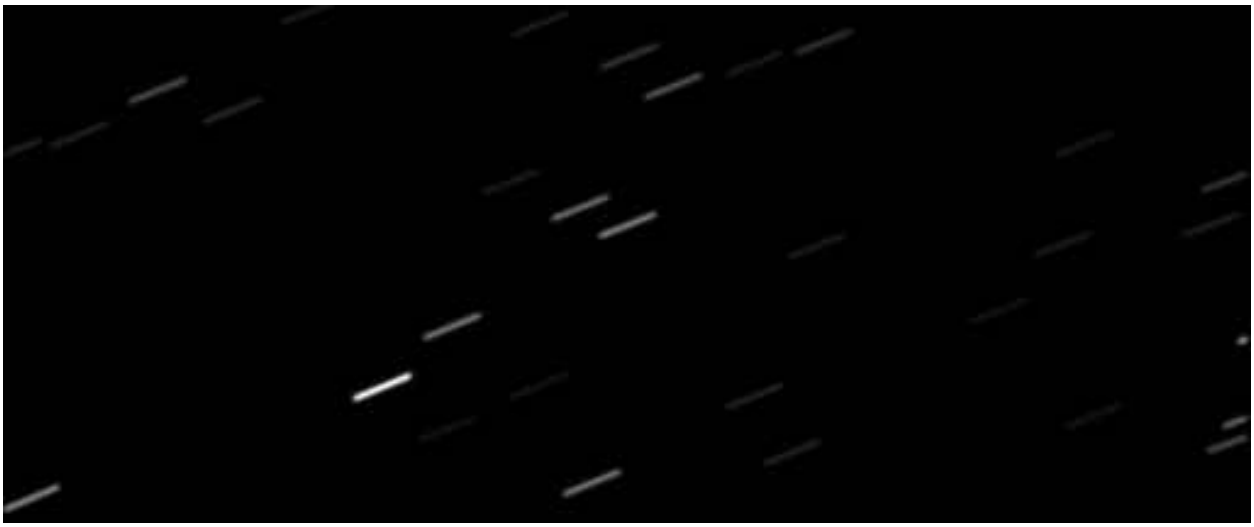

Figure 2.6: Complete full-frame streaked star camera image.

Noise is then added in the same manner as described in the previous section, completing the streaked star camera image.

Resident Space Object Implementation

The final characteristic of star cameras considered here is the tendency to sometimes pick up non-star objects. These are generally referred to as resident space objects (RSOs) and can be anything from a secondary spacecraft in a similar orbit to a comet or meteorite in an extreme hyperbolic orbit. Consideration of such cases is necessary for the analysis which will follow in Chapter 4.

Simple orbital dynamics are considered here, of the form [28]:

$$\begin{aligned} \dot{\vec{r}} &= \vec{v} \\ \dot{\vec{v}} &= -\mu \frac{\vec{r}}{r^3} + \vec{a}_{nonspherical} \end{aligned} \tag{2.13}$$

where $\vec{r}$ and $\vec{v}$ are measured in the Earth-centered inertial (ECI) coordinate frame. One perturbation to the general orbital dynamics is considered – the effect of Earth's oblateness. $\vec{a}_{nonspherical}$ represents the J_2 effect, which is an additional acceleration generated in orbit due to the bulging of Earth at the equator. An approximation of this acceleration is applied, in the form seen in Equation 2.14.

$$\vec{a}_{nonspherical} = \frac{-3J_2\mu {R_\oplus}^2}{2r^5} \begin{bmatrix} r_I \left(1 - {}^{5{r_K}^2}/_{r^2}\right) \\ r_J \left(1 - {}^{5{r_K}^2}/_{r^2}\right) \\ r_K \left(3 - {}^{5{r_K}^2}/_{r^2}\right) \end{bmatrix} \tag{2.14}$$

Though drag effects can also represent a considerable perturbation, consideration in the simulation adds significant complexity without contributing to the results in the following chapters. It is assumed that the spacecraft containing the camera is in an orbit which is only minimally affected by drag in the short-term, however it is expected that the results from Chapter 4 will extend to other orbits, as long as the spacecraft can be precisely tracked.

A set of constraints is set on image generation to provide a more realistic simulated environment. Specifically, eclipses of the RSO, glare from the Sun, and Earth occlusions are considered. Figure 2.7 details the layout of the orbital environment for a spacecraft and a resident space object, from which the following constraints are derived.

Eclipses of the RSO are given by the relation in Equation 2.15, derived in full in [29]. In the

implementation here, the RSO is only rendered if the following is true.

$$\frac{\vec{r}_{\odot} \cdot \vec{r}_{sc}}{r_{\odot} r_{sc}} > \frac{-\left({r_{sc}}^2 - {R_{\oplus}}^2\right)^{1/2}}{r_{sc}} \tag{2.15}$$

$r_{\odot}$ refers to the Sun vector relative to Earth, $R_{\oplus}$ refers to the radius of Earth, and $\vec{r}_{sc}$ is the location of the spacecraft relative to Earth's center. This is essentially a computation of the angle between the spacecraft position vector and the sun vector. When the computed angle falls within the envelope of Earth's shadow, it is considered eclipsed.

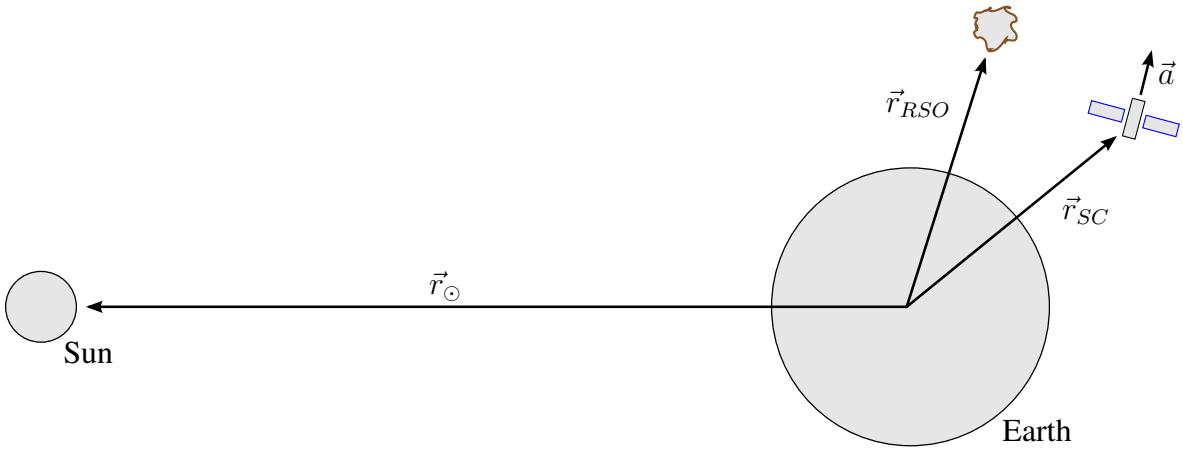

Figure 2.7: Orbital environment geometry.

From Figure 2.7, it can be seen that the angle between the attitude vector, $\vec{a}$, and the sun vector, $r_{\odot}$, is given by

$$\cos\left(\phi_s\right) = \frac{\vec{r}_{\odot} \cdot \vec{a}}{r_{\odot}} \tag{2.16}$$

Where $\vec{a} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} R_{cam}$. Variations in the Sun vector due to the position of the spacecraft about Earth can be neglected with negligible affect on the final result. For additional precision, the Sun vector should be calculated with respect to the current position of the spacecraft. Considering the field of view of the camera, FoV, it can be found that the sun is within the image frame

whenever the relation in Equation 2.17 is true.

$$\phi_s < \sin^{-1}\left(\frac{R_\odot}{r_\odot}\right) + \frac{FoV}{2} \tag{2.17}$$

Here it is assumed that the camera includes a baffle which does not allow stray light to enter the lens. For small and inexpensive satellites, however, this may not be the case. Then, it will be necessary to include a buffer angle in addition to the field of view to account for angles at which the sun is not in the frame, however stray light is entering the lens from extreme angles. A simple binary true or false case is considered here, and images are considered completely corrupted if the sun is anywhere in the frame.

Similarly, the angle between the attitude vector and the position vector of the spacecraft is given as

$$\cos\left(\phi_e\right) = \frac{-\vec{r}_s c \cdot \vec{a}}{r_s c} \tag{2.18}$$

It follows, then, that Earth is within the image if the following is true.

$$\phi_e < \sin^{-1}\left(\frac{R_\oplus}{r_{sc}}\right) + \frac{FoV}{2} \tag{2.19}$$

Again, the image is considered corrupted if Earth falls within the camera frame. This is due to reflected glare from the Earth's surface as well as obstruction of the camera. Similar concern for stray light entering the camera may be considered here, however this simulation does not include it.

RSOs are included in the star camera image in the same way that stars are in the previous sections. The Collinearity equation maps the spacecraft's position into the image plane. Images are generated generated and integrated over the exposure time to create a final, streaked RSO image. This image is then combined with It is assumed that the spacecraft is at a significant distance and

remains unresolved at all points in time. The spacecraft, then, acts as a single point of light, similar to a star. A visible magnitude appropriate for the type of satellite should be chosen based on published data. Specifics of determining the appropriate magnitude will be discussed further Chapter 4. Sudden variations in brightness, such as those due to a tumbling spacecraft, are not considered here. A sample image including a streaking RSO is shown in Figure 2.8.

Figure 2.8: Complete full-frame star camera image with pronounced RSO streak.

CHAPTER 3: LONG-INTEGRATION STAR TRACKER IMAGE PROCESSING FOR COMBINED ATTITUDE - ATTITUDE RATE ESTIMATION

Star trackers are limited to producing quality feedback at only small angular rates due to the tendency of stars to streak across the focal plane during an exposure [13]. Some previous work has approached a solution to this problem through geometric analyses of the signal, however no method is capable of approaching the accuracy of rotationally-constrained star tracking. Liebe et al. approached the problem by fitting a spherical circle to discovered star streak with results in the 55 arcsecond range and rate estimates within 5% of actual [30]. Simms applied a line fit to streaks, with an iterative least squares correction in the local region of each endpoint [31]. This produced more precise estimates, between $\frac{1}{10}$ and $\frac{1}{5}$ of a pixel for bright streaks, however this method is primarily intended for the handling of non-star objects which exhibit little curvature.

An algorithm is proposed here which employs traditional image processing methods to retrieve endpoint data from images otherwise corrupted due to streaking. The implemented methods rely primarily on the detection of corners within an image, for which there exist many well-known procedures [32, 33, 34]. This particular method is chosen for the geometry particular to streaked star signals. Endpoints of streaks will be shown to demonstrate very similar responses to corners when these algorithms are applied.

A step-by-step process for extracting streaks from an image, collecting and grouping endpoints, and obtaining attitude and rate estimates is presented. Monte Carlo simulation results are shown in order to characterize the behavior of the algorithm, and implications of the results are discussed.

Algorithm Description

Streak Identification & Separation

In order to enable the methods discussed in the following sections, which operate on the image locally, it is first necessary examine the image globally to separate out regions of interest. Regions of interest, in this case, are defined as large, contiguous areas above a certain threshold value in the grayscale image. First, then, the image must be thresholded at a certain value. A value of the mean value of the entire image is chosen in order to remove noise which could potentially connect streaks. This operation is performed as a binary logical operation so that areas below the threshold are set to zero and areas above to one. This converts the image into the desired format for centroiding.

Centroiding identifies contiguous areas in the image and computes the center point for each area. The specific method applied for centroiding is discussed by Mortari et al. in [15]. Computed center points are desirable in this case to provide discrete locations for each possible streak. A bounding box may also be computed for each discovered centroid to separate regions of interest for the remaining operations. For the subsequent endpoint detection, discussed below, the operators only act locally on the regions of interest containing each streak. The binary image is discarded and the locations in the preserved initial image are referenced. In this way, required computational time is reduced by removing operations on unnecessary data.

Some smoothing may be necessary at this point in order to reduce noise within the image. This, however, depends on the expected characteristics of the noise and the possibility of other types of corruption in the image. Operations specific to the expected behavior of the camera hardware should be applied for noise reduction.

Endpoint Detection

Gaussian Derivative Kernel

The foundation of the following corner detection methods is the ability to take a spatial derivative within an image. A simple method for producing a derivative in a given direction is to subtract neighboring pixels to calculate a difference. For example, the derivative at pixel (x, y) in the x direction may be given as

$$\frac{\partial M_{x,y}}{\partial x} = M_{x-1,y} - M_{x+1,y} \tag{3.1}$$

Here, M represents the complete image as a large matrix of intensity (brightness) values. This operation is effectively a convolution of the entire image with the kernel

$$h = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \tag{3.2}$$

The resulting image contains the approximate derivative at each pixel. Due to the presence of noise and low order of the derivative, this method often does not provide an accurate derivative. The Gaussian derivative kernel combines a smoothing operation with the derivative operation, providing a simultaneous filtering effect to produce an improved response in the presence of noise. The Gaussian derivative kernel follows the 2-dimensional Gaussian distribution [35].

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{3.3}$$

A simple 3×3, x derivative Gaussian kernel with $\sigma = 0.5$ is shown in Equation 3.4, below.

$$h = \begin{pmatrix} 0.0466 & 0 & -0.0466 \\ 0.3446 & 0 & -0.3446 \\ 0.0466 & 0 & -0.0466 \end{pmatrix} \tag{3.4}$$

Convolution of this kernel with the image provides the necessary gradients in the following two methods. For each of the four methods discussed below, the resulting cornerness measures are computed for a typical star streak, seen in Figure 3.1.

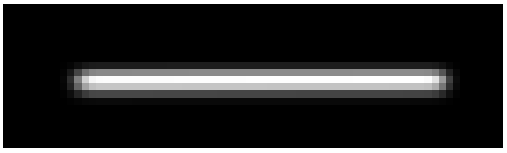

Figure 3.1: Sample star streak.

Minimum Eigenvalue Method

The primary component behind the Minimum Eigenvalue method and the following Harris method is the structure tensor, A, detailed in Equation 3.5. I_x and I_y refer to the calculated gradient in the corresponding directions – in this case, the Gaussian derivative kernel discussed in the previous section is used. This tensor is used as an indicator for certain features within an image [33].

$$A = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} \tag{3.5}$$

In this method, as the name suggests, the Eigenvalues of A, λ_1 and λ_2, are considered to describe the presence of corner-like features. For example, if both λ_1 and λ_2 are small for a given pixel, the

pixel is not considered to contain a feature. A strong gradient in a single direction, with $\lambda_1 \gg \lambda_2$ or $\lambda_2 \gg \lambda_1$, suggests that the pixel contains an edge. Two large eigenvalues, then, suggests a strong gradient in the two principle directions, which further suggests that the area contains a corner. This method aims to measure this property by generating a cornerness measurement from the minimum eigenvalue for each pixel [33]. Formally,

$$R = \min(\lambda_1, \lambda_2) \tag{3.6}$$

The resulting response contains the same dimensions as the original image, with brighter pixels suggesting a higher likelihood that a corner resides there.

Figure 3.2: Minimum eigenvalue cornerness measure computed for a typical star streak.

Harris Method

The Harris method attempts to provide a more streamlined approach to generate a cornerness measurement. Again, the structural tensor from Equation 3.5 is used. Here, however, the Eigenvalues are not directly calculated. Instead, the cornerness score is given as a function of the calculated gradients as follows [32]

$$R = I_x^2 I_y^2 - I_{xy}^2 - k\left(I_x^2 + I_y^2\right)^2 \tag{3.7}$$

k is a user-defined parameter in the range of $0 \leq k \leq 0.25$. As before, Gaussian derivative kernels are used for the calculation of gradients. This further reduces to

$$R = Det\,(A) - kTr\,(A)^2 \tag{3.8}$$

Since this method does not require calculation of $Resolution^2$ sets of Eigenvalues (one for each pixel within the image) it requires considerably less computational effort. For the purposes of the work discussed here, a k value of 0.15 is used. This particular value was tuned manually, however additional work could be performed to iteratively determine a more optimal value. The resulting cornerness measure is seen in Figure 3.3.

Figure 3.3: Harris cornerness measure computed for a typical star streak.

Trajkovic Method

The Trajkovic corner detector comes in two forms: one method which includes four neighboring pixels and an extended method which includes all eight neighboring pixels. Both methods follow the same general format so only the four neighbor method will be discussed here. The reader is referred to [34] for a complete derivation and discussion of the method outlined here as well as the extended 8-neighbor method.

To initiate the corner detection process, a scaled-down version of the original image is considered

to determine likely corner locations. A simple cornerness is calculated of the form

$$R_{simple} = \min\left(r_A, r_B\right) \tag{3.9}$$

where

$$\begin{aligned} r_A &= \left(I_{x+1,y} - I_{x,y}\right)^2 + \left(I_{x-1,y} - I_{x,y}\right)^2 \\ r_B &= \left(I_{x,y+1} - I_{x,y}\right)^2 + \left(I_{x,y-1} - I_{x,y}\right)^2 \end{aligned} \tag{3.10}$$

$I_{x,y}$ refers to the intensity for the pixel at location (x, y). This simple method is similar to the previous methods in that it attempts to calculate the minimum gradient about a specific pixel. For the initial calculation, however, only two directions are considered, providing a coarse estimate of the gradient at each pixel. A user-defined threshold, T_1, is applied to the result in order to prevent some false positives.

Then, an "inter–pixel" cornerness measure is calculated. This cornerness measure is essentially an interpolation between the simple gradients calculated above. Such an interpolation allows one to determine, analytically, the minimum gradient in any direction. This is similar to both of the preceding methods. The inter–pixel cornerness measure is given by

$$R_{Interpixel}\left(x, y\right) = \begin{cases} C - \frac{B^2}{A} & \text{if } B < 0 \text{ and } (A + B) > 0 \\ C_{Simple}\left(x, y\right) & \text{else} \end{cases} \tag{3.11}$$

Here, B is calculated from B_1 and B_2, two intermediate values [34].

$$\begin{aligned} B_1 &= \left(I_{x,y+1} - I_{x+1,y}\right)\left(I_{x+1,y} - I_{x,y}\right) + \left(I_{x,y-1} - I_{x+1,y}\right)\left(I_{x-1,y} - I_{x,y}\right) \\ B_2 &= \left(I_{x,y+1} - I_{x-1,y}\right)\left(I_{x-1,y} - I_{x,y}\right) + \left(I_{x,y-1} - I_{x+1,y}\right)\left(I_{x+1,y} - I_{x,y}\right) \end{aligned} \tag{3.12}$$

The value of B is then given as $B = \min\left(B_1, B_2\right)$. The remaining values, C and A are of the form

$$C = r_A$$
$$A = r_B - r_A - 2B \tag{3.13}$$

Again, the 8–neighbor method follows a similar procedure, however all eight of the neighboring pixels are included in the simple and inter–pixel cornerness measures [34]. Cornerness measures for both methods are shown in Figure 3.4.

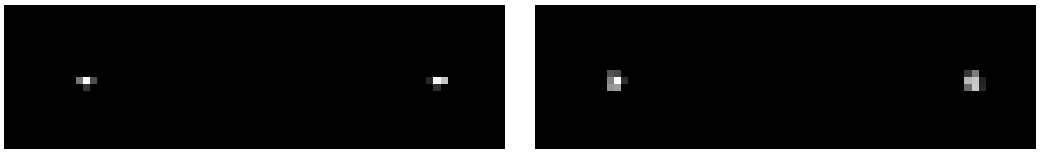

Figure 3.4: Trajkovic 4- and 8- neighbor cornerness measure for a typical star streak.

Endpoint Localization

Each of the preceding methods produces a measure of "cornerness" of the same dimensions as the original image. Pixels with higher intensity correspond to a higher likelihood of a corner being contained within that pixel. In order to produce an (x, y) coordinate for the endpoint, two localization methods are applied. The first, sub-maximal suppression, is an iterative method. Each pixel within the cornerness response is considered, and only pixels which are greater than all of the neighboring eight pixels are returned as potential endpoints. Such pixels correspond to a peak in the cornerness data. This process provides a set of pixel-accurate locations for the location of the endpoint in the form of an integer (x, y) coordinate.

At this point it becomes possible to remove some corrupted data. When a single streak is passed through the corner detector and then subsequently the sub-maximal suppression operation, only two endpoints should appear within the data. Samples which return more than two endpoints may

be removed from consideration as there is likely some form of corruption present in that portion of the image. This may be due to noise within the image, overlapping star streaks, or another non-star object within the image.

Since the location produced by sub-maximal suppression is only pixel accurate, up to half a pixel of error remains in the localization of the endpoint. This corresponds to an angular error up to

$$\theta_{err} \leq \frac{1}{2} \frac{FoV}{Resolution} \tag{3.14}$$

and potentially discards some of the information remaining within the image. This of course does not consider other sources of error, which could cause it to be greater than this value. To reduce this as a potential source of error, sub-pixel localization is performed. The chosen method here is a "center of intensity" calculation of the same form as a traditional center of mass calculation. The pixel-accurate location is chosen as the origin, and the surrounding pixels are combined in a weighted summation proportional to distance from the origin to calculate an updated endpoint location. The full relation is seen in Equation 3.15. N defines the size neighborhood about the origin to consider. Any value may be used for N, however there is diminishing benefit as N grows.

$$\begin{aligned} X &= \frac{1}{w} \sum_{\substack{i=-N \\ i \neq 0}}^{N} \sum_{j=-N}^{N} I_{i,j} \\ Y &= \frac{1}{w} \sum_{i=-N}^{N} \sum_{\substack{j=-N \\ j \neq 0}}^{N} I_{i,j} \end{aligned} \tag{3.15}$$

w is the sum of all pixels within the neighborhood.

$$w = \sum_{i=-N}^{N} \sum_{j=-N}^{N} I_{i,j} \tag{3.16}$$

The resulting (X, Y) pair is a sub-pixel accurate endpoint location. This operation is performed for all possible pairs of endpoints discovered within the image.

Isolating Endpoint Groups

With a complete set of discovered endpoints, it is then necessary to group the endpoints in time. Though it is not possible to determine if a point occurred at the initial time or final time, it is possible to determine which endpoints occurred simultaneously. This is achieved through the calculation of the center of rotation and a vector cross-product operation. First, the set of midpoints for each streak are calculated.

$$\begin{bmatrix} x_{i,c} \\ y_{i,c} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_{i,1} + x_{i,2} \\ y_{i,1} + y_{i,2} \end{bmatrix} \tag{3.17}$$

$x_{i,1}$, $x_{i,2}$, $y_{i,1}$, and $y_{i,2}$ are the coordinates describing the two discovered endpoints for the i^{th} streak. The designations "1" and "2" are arbitrary and do not correspond to occurrence in time. Additionally, the slope, m_i, is calculated for each set $(x_{i,1}, y_{i,1})$ and $(x_{i,2}, y_{i,2})$. With these values, it is then possible to construct a set of equations describing a line which bisects each streak. The benefit of this is that the common intersection point of this set of lines corresponds to the center of rotation as projected into the image plane. A least squares solution for the coordinates of the

center of rotation is constructed as

$$\begin{pmatrix} 1 & \frac{1}{m_1} \\ 1 & \frac{1}{m_2} \\ \vdots & \vdots \\ 1 & \frac{1}{m_N} \end{pmatrix} X = \begin{bmatrix} m_1 x_{1,c} + y_{1,c} \\ m_1 x_{2,c} + y_{2,c} \\ \vdots \\ m_1 x_{N,c} + y_{N,c} \end{bmatrix} \tag{3.18}$$

N corresponds to the number of discovered streaks. An additional set of vectors, V_1 and V_2, is calculated from the midpoint of each streak to its corresponding endpoints. Additionally, V_c is calculated from the center of rotation to each midpoint.

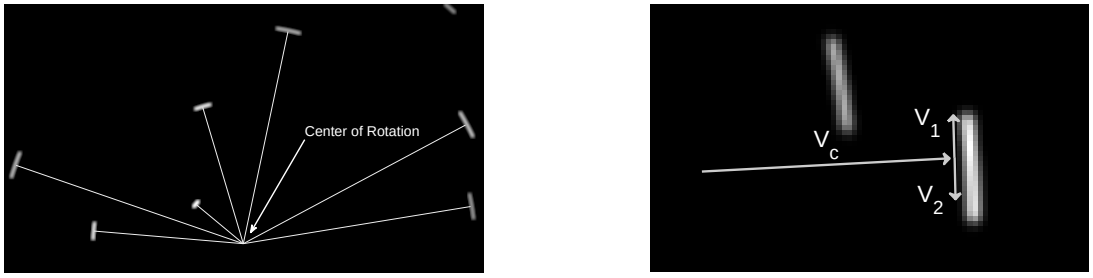

Figure 3.5: Location of the center of rotation in a streaked star image (left) and detail of the complete vector set constructed for each star streak (right).

Since these vectors are in the 2-D image plane, an arbitrary scalar third component is added in order to allow for a vector cross product. Here, a value of 1 is used. Since V_1 and V_2 represent opposite rotations about the center of rotation, the following relation holds true.

$$[V_c \times V_1]_3 = -[V_c \times V_2]_3 \tag{3.19}$$

The resulting third component of a cross product with V_c provides an invariant detector of endpoint groups.

Attitude and Rate Estimation

To perform attitude and rate estimation on the resulting endpoints, the data must be converted to a usable form for the appropriate estimation routines. The (x, y) coordinates are converted to 3-dimensional unit vectors which describe the direction of the point relative to the camera plane. This is achieved through Equation 3.20, where each coordinate is scaled by the physical pixel dimensions of the CCD and the focal length, f, is appended.

$$V_{Body} = \begin{bmatrix} \vec{S} \cdot V_{Plane} & f \end{bmatrix}^T \tag{3.20}$$

$\vec{S}$ is the set of physical pixel dimensions, of the form

$$S = \begin{bmatrix} s_x & s_y \end{bmatrix}^T \tag{3.21}$$

where s_x and s_y correspond to the dimensions in the subscripted axes. Following this correction, any traditional star identification and attitude estimation techniques may be applied. Here, the k-vector search algorithm, proposed by Mortari in [18], is used to identify the discovered vectors within the Hipparcos star catalog. This method considers sets of three stars in triangle formations. Off-line, a set of possible triangular star structures existing within a pre-selected catalog is constructed. This data can later be searched on-line – at a significant reduction in computation time – for triangular structures discovered within a star image. A structure match results in a positive ID of the stars existing within the structure as well as the surrounding stars. Any star which is not part of any structure in the catalog is returned as a possible RSO. This particular algorithm was chosen for its low incidence of false-positives and the low computation time required for a positive identification.

The k-vector algorithm is run independently on each of the two sets of corrected endpoint vectors,

and returns a set of reference unit vectors, r_i, corresponding to each correctly-identified star. The resulting vector sets are of the form

$$b_i = Ar_i,\ i = 1 \ldots N \tag{3.22}$$

b_i describes the unit vectors discovered within the image. A describes the unknown rotation between the vector sets. This form is desirable as it has been discussed extensively from an estimation point of view. Specifically, Wahba posed this formulation in [19], and it is known as "Wahba's Problem". The problem is to find an optimal estimation for the rotation, $\hat{A}$, given that one or both sets of vectors contain some noise. Formally,

$$L_w\left(\hat{A}\right) = \frac{1}{2}\sum_{i=1}^{N} \alpha_i \left\| b_i - \hat{A}r_i \right\|^2 \tag{3.23}$$

Here, α_i refers to weighting which may be applied to each set of vectors – for example, if one source of measurements is known to include additional uncertainty, it may be weighted lower. In general, and for the purposes of this work, the weighting is defined as $\alpha_i = {}^1/_N$ for all i. Many analytical solutions exists for Wahba's problem [21, 22]. For the following analysis, the Second EStimator of the Optimal Quaternion (ESOQ2) is applied. This algorithm contains an analytical solution for Wahba's problem and provides the optimal quaternion rotation between the vector sets. For a full discussion of the operation of ESOQ2, the reader is referred to [21]. The ESOQ2 method is applied to each set of endpoints and their respectively identified reference vectors from the star catalog. At this point, two separate attitudes are computed, one for the initial time and one for the final time. An ambiguity still remains in the ordering in time of the two attitudes, however some form of tracking between images removes this concern. Additionally, if there is some *a priori* knowledge of the rotation direction, it is possible to overcome this with no additional processing.

In order to calculate a rotation rate from the previous data, the principle angle between the two

measurements is calculated. First, Δ is computed

$$\Delta = \hat{A}_1 \hat{A}_2^T \tag{3.24}$$

This corresponds to the complete rotation DCM from the first measurement to the second. Then, calculation of the principle angle follows simply through the relation

$$\Phi = \cos^{-1}\left\{\frac{1}{2}\left[tr(\Delta) - 1\right]\right\} \tag{3.25}$$

For complete data, the principle axis is computed as well.

$$[a\times] = \frac{1}{2\sin\Phi}\left(\Delta - \Delta^T\right) \tag{3.26}$$

This corresponds directly to the axis of rotation with respect to the camera's coordinate frame. Finally, $\vec{\omega}$ follows simply by considering the integration time, T.

$$[\omega\times] = R_{cam}^T \frac{\Phi}{T}\left[a\times\right] \tag{3.27}$$

Applying R_{cam}^T accounts for differences between the camera frame and the body frame. For the results in the next section, this is assumed to be identity. $[\omega\times]$ denotes the skew-symmetric matrix.

Results and Discussion

A large-scale Monte Carlo analysis was run in order to characterize the performance of the algorithm detailed above in a wide range of scenarios. Specifically, different camera orientations with respect to the rotation axis were the primary concern. A diagram of the camera-to-spacecraft orientation is seen in Figure 3.6. As the camera becomes more aligned with the rotation axis, the

streaks produced become shorter and more curved. When the camera is completely aligned with the rotation axis, there exists a wide range of possible streak lengths within a single image. A star at the center of such an image would not streak at all, however a star near the edge would eventually describe a complete circle about the center of rotation. On the other hand when the camera is placed orthogonally to the rotation axis, the streaks are straight and parallel throughout the image. For large fields of view, some curvature will become apparent at the edges even for the orthogonal case due to the nonlinearity of the camera projection.

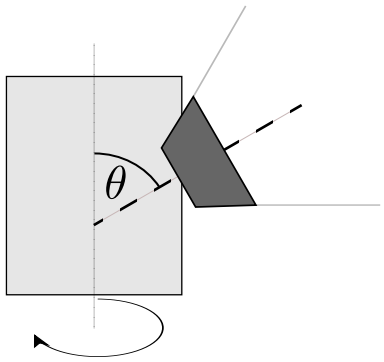

Figure 3.6: Camera position relative to the axis of rotation.

Orientation from $\theta = 0^\circ$ to $\theta = 90^\circ$ were considered. Symmetric behavior was observed about $\theta = 90^\circ$, therefore it was not necessary to simulate the entire 180°. Additionally, the streak lengths were varied from 5% of the field of view to 50%. This was achieved by varying the integration time according to

$$\%FoV = \frac{\|\vec{\omega}\| \cdot T}{FoV} \tag{3.28}$$

$\vec{\omega}$ was considered to be constant throughout the simulation. To further generalize the results, single-point attitude error values were computed as a pixel measurement. Since these results are highly dependent on the parameters of the specific camera employed, error measurements in pixels can be extrapolated to angular error for different camera properties. This error calculation is achieved

through

$$e = \Phi_e \cdot \frac{Resolution}{FoV} \tag{3.29}$$

where Φ_e is the angular error between the measured vector and the truth. Results for a specific resolution and field of view may be calculated from Φ_e through the same relation. Error in the rate estimation was computed as a percentage variation from the truth. Figures 3.7 and 3.8 detail the results for attitude error and Figures 3.9 and 3.10 detail the rate error. Both attitude error and rate error are plotted as camera orientation and steak length versus the respective error metric. All results seen here have been smoothed in order to provide estimates of the average performance at every parameter set.

The first characteristic to note in all of the figures is the degenerate behavior for short streak lengths in the neighborhood of $\theta = 0^\circ$. This is caused by the fact that, for these parameters, the stars within the image are streaking very little or not at all. In this case, the algorithm described here no longer functions, however traditional methods are still valid. If a spacecraft is expected to experience the full range of conditions here, it is necessary to choose a transition point to and from traditional star camera analysis. A similar feature is seen in all of the figures as θ approaches 0 for streak lengths above 20% of the field of view. This is caused primarily due to the shortness of streaks due to the high curvature present when the camera is aligned with the rotation axis. One possible solution is to increase the integration time of the camera in order to achieve longer streak lengths. The Trajkovic methods, however, have a higher tendency to find false corners with increased curvature.

The remaining characteristics are dependent on the endpoint detection method applied, and will be discussed individually. Accuracies for single-endpoint attitude for the Harris (top) and Minimum Eigenvalue method (bottom) are detailed in Figure 3.7. These two methods showed the best performance overall, with the Harris method performing slightly better than the Minimum Eigenvalue method. Both displayed accuracies within $\frac{1}{5}$ of a pixel.

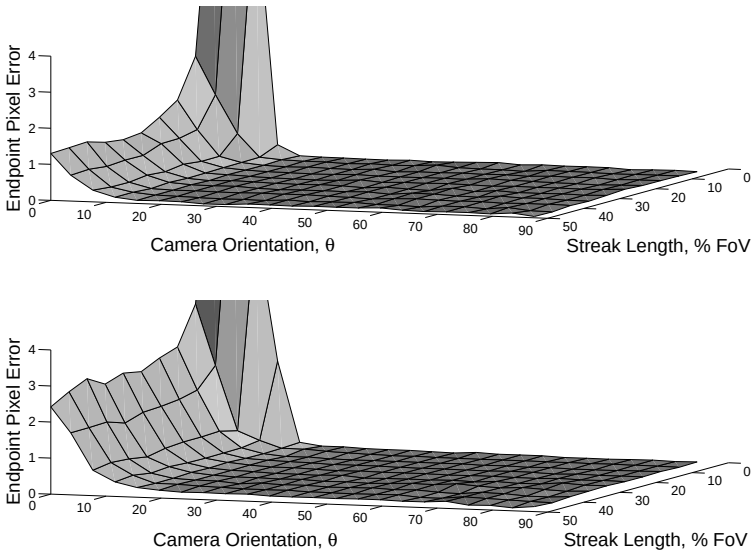

Figure 3.7: Attitude accuracies for Harris (top) and Minimum Eigenvalue (bottom) methods.

An important point to note here is that each method has a "precise region" wherein the algorithm demonstrates relatively stable performance. This is primarily due to the slope encountered at small streak lengths and θ. The precise region of each method was characterized and all accuracies are reported within this precise region. The Harris detector had the largest precise region of $\theta > 20°$. The precise region of all other methods falls roughly within $\theta > 30°$. The Trajkovic methods performed worse over the precise region, with attitude accuracies in the half-pixel range.

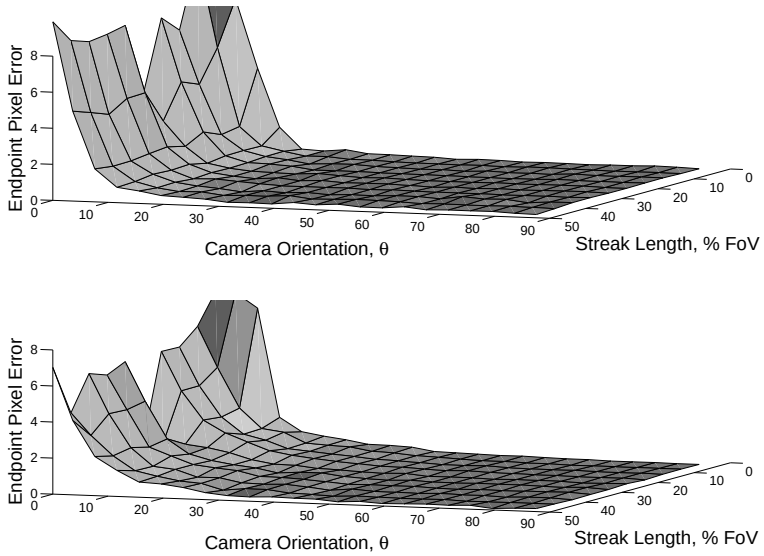

Figure 3.8: Attitude accuracies for Trajkovic 4-neighbor (top) and 8-neighbor (bottom) methods.

Angular rate estimation performance for the Harris and Minimum Eigenvalue methods are shown in Figure 3.9. Similar behavior to the previous figures is seen beyond the precise regions. Rate error for the Harris and Minimum Eigenvalue methods were within approximately 0.7% and 1.2%, respectively. Another important point to note here is that the simulation did not consider streak lengths below 10% of the field of view. Performance in this area degrades rapidly, much like the degenerate region, due to the shortening of the streaks. As streaks become shorter, the streak approaches a boundary, known as the Rayleigh limit, where is becomes impossible to distinguish two signals and it is effectively a single star [36]. This algorithm does not operate on single stars, so performance becomes increasingly poor for streak lengths below 10% of the field of view. If a spacecraft is expected to encounter all regions, it is necessary to choose a transition point from this algorithm to traditional methods. Performance should be characterized on the specific hardware in order to choose an optimal transition point which minimizes error in all expected scenarios.

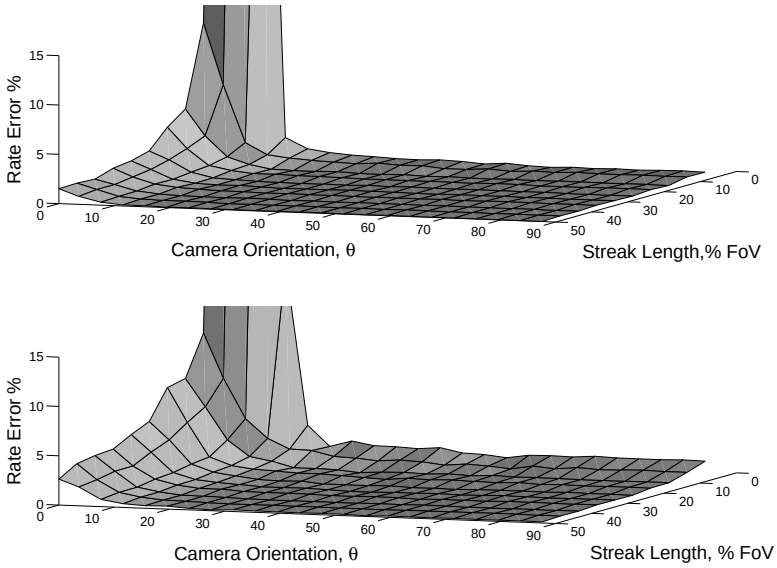

Figure 3.9: Angular rate error for Harris (top) and Minimum Eigenvalue (bottom) methods.

Rate error for the Trajkovic methods is detailed in Figure 3.10. Both methods performed more poorly than the previous two, within approximately 3.5% error. In the rate data, another feature of the algorithm becomes apparent: rate accuracy increases with streak length. This is visualized in the figures by the increase of error as streak lengths approach smaller percentages of the field of view. Small errors at the endpoints of streaks become less significant as the length of the streaks increases. It is important to note, however, that eventually the streaks will have a tendency to only fall partially within the image or completely across the field of view. In these cases the method may not discover any endpoints, causing a failure to return any data.

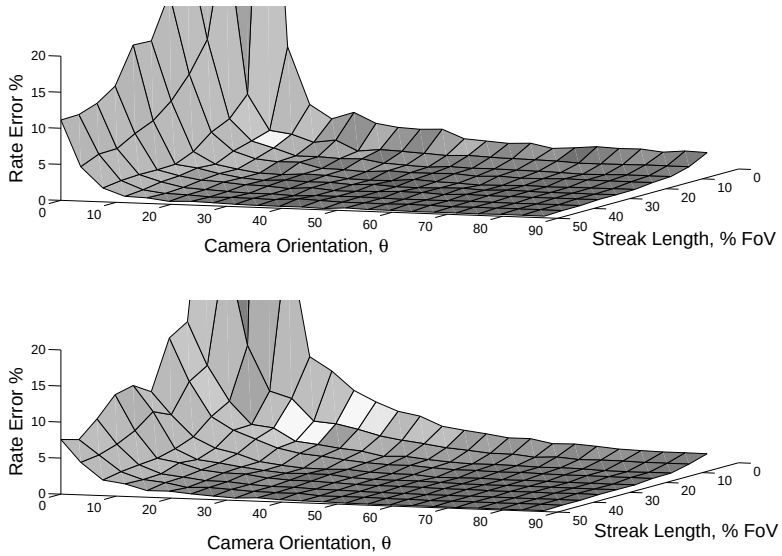

Figure 3.10: Angular rate error for Trajkovic 4-neighbor (top) and 8-neighbor (bottom) methods.

A complete summary of the results contained within these figures is detailed below. All performance characteristics are measured within the precise region listed for each method.

Table 3.1: Monte Carlo result summary for all endpoint detection methods.

Method	*Attitude Accuracy*		*Rate Accuracy*		*Precise Region*
	10% S.L.	50% S.L.	10% S.L.	50% S.L.	
Harris	< 0.10 px	< 0.10 px	$< 0.69\%$	$< 0.07\%$	$\theta > 20°$
Min. Eig	< 0.19 px	< 0.20 px	$< 1.20\%$	$< 0.19\%$	$\theta > 30°$
Trajkovic-4	< 0.39 px	< 0.37 px	$< 2.44\%$	$< 0.38\%$	$\theta > 30°$
Trajkovic-8	< 0.55 px	< 0.29 px	$< 3.49\%$	$< 0.47\%$	$\theta > 30°$

A final characteristic to note is that much of the error present in the endpoint locations, and subsequently the attitude and rate, is due to a bias caused by the image processing. During the smoothing processes which are applied in the Gaussian derivative kernel as well as the corner detector operations, the locations of the endpoints are note necessarily preserved. In fact, it has been observed

that each gaussian convolution biases the endpoints further inward, ultimately converging to a point source centered between the original endpoints. Currently there is no known procedure for preventing this effect and producing un-biased results. It has been observed, however, that the bias remains constant for constant camera parameters, so it is possible to characterize the effect experimentally and provide some compensation.

CHAPTER 4: CATALOG-FREE ANGULAR RATE ESTIMATION WITH ON-LINE RESIDENT SPACE OBJECT DETECTION

An extension to techniques introduced in chapter three is possible, in the form of a rate-only estimation method. Rather than utilizing a star catalog reference to produce two attitude fixes per image, which may not be necessary in all cases, the algorithm can be modified to produce only a rate estimate and no attitude data. This has an added benefit of reducing required computation time for a costly star identification process.

Additionally, A caveat in the algorithm discussed in the previous section is that, in some cases, data which seems valid will appear in the endpoint data set despite being corrupted or corresponding to a non-star object. To reduce this effect, and offer the ability to detect RSOs for tracking maneuvers, three indicators of non-star behavior are proposed. The first two provide a complete characterization of the apparent motion present in RSOs captured in a star image. The third indicator combines the two basic methods into a single process.

The combined ability to estimate angular rate and detect an RSO opens the possibility of precisely tracking an RSO with no hardware other than a camera and a processing unit, and no catalog reference. Results for two simple cases are presented – an RSO-tracking case and a constant slew maneuver containing a streaked RSO. A final 10-orbit simulation is conducted and results are discussed.

Catalog-Free Rate Estimation

A rate-only estimation scheme based on the analysis contained in the previous chapter is proposed. The same procedure from the previous section for obtaining endpoints and grouping them accord-

ing to the center of rotation is utilized here. From the set of data collected from a star camera image, it is possible to calculate an angular rate without the need for a precise attitude calculation and star catalog. This is performed in a fashion similar to Wahba's problem for attitude, shown previously in Equation 3.23. Rather than comparing each set of endpoints to a set of reference vectors, a solution for Wahba's problem is found for the rotation between the two sets of endpoints. The Optimal Linear Attitude Estimator (OLAE) is applied here, rather than ESOQ2 which was used in the previous section. This is due to the simplicity of calculating a residual with the OLAE method, which will be discussed in further detail in the next section. First, each step of the OLAE method is summarized here. For the full derivation of the method, the reader is referred to [22].

First, two initial values are defined from the set of observed and reference vectors, b_i and r_i, respectively. The subscript i is the index of the observation-reference pair, from 1 to N, where N is the number of observations.

$$\begin{aligned} \tilde{s}_i &\equiv b_i + r_i \\ \tilde{d}_i &\equiv b_i - r_i \end{aligned} \tag{4.1}$$

Then, two additional intermediate values are computed. $\tilde{v}$ is in the form

$$\tilde{v} = \frac{1}{2} \sum_{i=1}^{n} \xi_i \left[\tilde{s}_i \times\right] \tilde{d}_i \tag{4.2}$$

where ξ_i is the weighting provided with each measurement and $[\tilde{s}_i \times]$ is the skew-symmetric form of $\tilde{s}_i$. The simplest option for the weightings is a uniform distribution, where $\xi_i = \frac{1}{N}$. This is the form used here for all uncorrupted data. The final required value, $\tilde{M}_m$, is calculated in a similar form,

$$\tilde{M}_m = \frac{1}{2} \sum_{i=1}^{n} \xi_i \left[\tilde{s}_i \times\right] \left[\tilde{s}_i \times\right] \tag{4.3}$$

The optimal solution is provided in the Rodrigues parameter space. The estimated set of three

Rodrigues parameters, $\hat{g}$, is related to $\tilde{M}_m$ and $\tilde{v}$ through the following relation.

$$\tilde{M}_m \hat{g} = \tilde{v} \tag{4.4}$$

This can easily be solved for the attitude solution in Rodrigues parameters. An additional step is necessary here to convert from the Rodrigues solution to the optimal quaternion estimate. First, a scalar 1 is appended to $\hat{g}$.

$$\hat{q} = \begin{bmatrix} \hat{g}^T & 1 \end{bmatrix} \tag{4.5}$$

Then, the optimal estimation of the quaternion follows simply by normalizing the resulting vector.

$$q_{opt} = \frac{\hat{q}}{\|\hat{q}\|} \tag{4.6}$$

This is the optimal quaternion solution to Wahba's problem provided by OLAE. A residual for each measurement may be computed through the relation

$$e_i = \left\| [\tilde{s}_i \times] \, \hat{g} - \tilde{d}_i \right\| \tag{4.7}$$

Since this method provides simple calculation of the residual, it allows some simple noise filtering to be applied. For example, it is possible to threshold measurements with large residuals and recompute the solution to reduce the affect of outlying measurements.

Now that the rotation between the two sets of endpoints is known, it is possible to derive a rate estimate. First, the quaternion solution is converted to a direction cosine matrix (DCM) as in

Equation 4.8 [27].

$$A(q) = \begin{bmatrix} {q_1}^2 - {q_2}^2 - {q_3}^2 + {q_4}^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & -{q_1}^2 + {q_2}^2 - {q_3}^2 + {q_4}^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_1q_2 + q_3q_4) & -{q_1}^2 - {q_2}^2 + {q_3}^2 + {q_4}^2 \end{bmatrix} \tag{4.8}$$

As in the previous section, the principle angle, Φ, and the axis of rotation, $[a\times]$ are computed.

$$\begin{aligned} \Phi &= \cos^{-1}\left\{\frac{1}{2}\left[tr(\hat{A}) - 1\right]\right\} \\ [a\times] &= \frac{1}{2\sin\Phi}\left(\hat{A} - \hat{A}^T\right) \end{aligned} \tag{4.9}$$

$\vec{\omega}$ follows through consideration of the principle angle and axis as well as the integration time of the camera, T.

$$[\omega\times] = R_{cam}^T \frac{\Phi}{T}[a\times] \tag{4.10}$$

RSO Indicators

Three indicators of a streak corresponding to a resident space object are considered here. Figure 4.5 provides a visual reference for the methods discussed here. In this simulated scenario, the camera-equipped spacecraft performed a constant-rate slewing maneuver. A star camera image was captured during the maneuver which also contained an RSO with some apparent motion. A large visible magnitude of $M_v = 1$ was used to for the RSO for visualization purposes. All other streaks within the image correspond to stars from the Hipparcos star catalog.

In discussion of the following methods, this example case will be used for reference.

Figure 4.1: Sample star camera image containing and RSO with apparent motion.

Center of Rotation

Two basic methods exist for choosing outliers in a data set from apparent motion alone. The first is based on the computed axis of rotation. All of the stars within the image rotate about a common location, which corresponds to the axis of rotation as projected to the image plane. Any RSO with apparent motion will create a larger residual in the resulting calculation. The least squares center of rotation solution, discussed in the previous chapter, is shown again here.

$$\begin{pmatrix} 1 & \frac{1}{m_1} \\ 1 & \frac{1}{m_2} \\ \vdots & \vdots \\ 1 & \frac{1}{m_N} \end{pmatrix} X = \begin{bmatrix} m_1 x_{1,c} + y_{1,c} \\ m_1 x_{2,c} + y_{2,c} \\ \vdots \\ m_1 x_{N,c} + y_{N,c} \end{bmatrix} \tag{4.11}$$

Again, m_i is the slope of the i^{th} set of endpoints and $(x_{i,c}, y_{i,c})$ is the midpoint of the same set of endpoints. Referring to Equation 4.11 in the form

$$AX = b \tag{4.12}$$

The residual is given as

$$e_i = AX - b \tag{4.13}$$

This residual is the first indicator for an RSO within the image. A simple method of thresholding the residuals is applied here, however and relevant statistical analysis may be applied to the data for a more precise analysis.

The example case from Figure 4.5 is an important case to note here, as it will produce a degenerate solution for the center of rotation location and the residuals will no longer provide useful data. Consideration should be given to the expected behavior of the spacecraft – if the rotation axis is not expected to be entirely about the x and y axes of the focal plane, this method is valid. If a wide range of operating conditions are expected, this indicator transitions from the center of rotation residual to the slopes, m_i, as an indicator for each streak. A more highly sloped streak likely corresponds to an RSO. A pseudo-inverse and the norm of the resulting computed center of rotation has been determined to be a useful trigger for this transition; the center of rotation will rapidly become distant from the center of the image as the camera becomes orthogonal to the rotation axis.

Since the camera in this scenario is oriented orthogonally to the rotation axis in this scenario, the slope indicator is used. The computed slopes for all streaks in Figure 4.5 are plotted in Figure 4.2. One clear outlier can be seen in the data, 3.6σ from the mean.

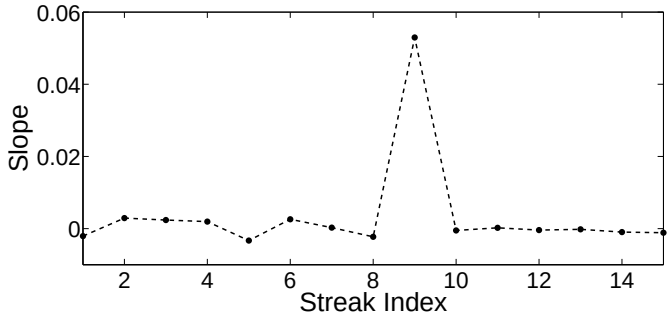

Figure 4.2: RSO indicator data for the center of rotation.

Streak index 9 is confirmed to correspond to the RSO, marking a successful identification.

Angular Displacement

The second basic indicator corresponds to the angular displacement subtended by each streak. True stars in the image frame will undergo motion consistent with the angular displacement of the camera during the exposure. Any apparent motion of the RSO along the rotational direction will thereby produce a useful variation which may not be apparent in the center of rotation residual alone. To reveal the angular displacement of an individual streak, it is necessary to return to the vector set derived for endpoint grouping in Equation 3.19 in the previous chapter. To reiterate, the relative position vector from the center of rotation to each endpoint is calculated and designated as $\vec{V}_{i,1}$ and $\vec{V}_{i,2}$. As before, the designations "1" and "2" are arbitrary and do not have any significance to the occurrence in time. i designates the index of the streak being considered, from 1 to N where N is the total number of discovered streaks. With this set of vectors, a simple dot product operation

reveals the angular displacement.

$$\phi_i = \cos^{-1}\left(\frac{\vec{V}_{i,1}\cdot\vec{V}_{i,2}}{\left\|\vec{V}_{i,1}\right\|\left\|\vec{V}_{i,2}\right\|}\right) \tag{4.14}$$

Here, again, consideration must be given to the degenerate case for the center of rotation. If the computation of the center of rotation returns an invalid result, then the vectors, $\vec{V}_1$ and $\vec{V}_2$ cannot be computed. In this case, the angular displacement indicator transitions to a simple length calculation between the two endpoints.

$$L_i = \sqrt{\left(x_{i,2} - x_{i,1}\right)^2 + \left(y_{i,2} - y_{i,1}\right)^2} \tag{4.15}$$

For the example scenario, the lengths for all streaks were computed and can be seen in Figure 4.3. The same index which generated an outlier in the slope data is seen again as an outlier, 3.5σ from the mean.

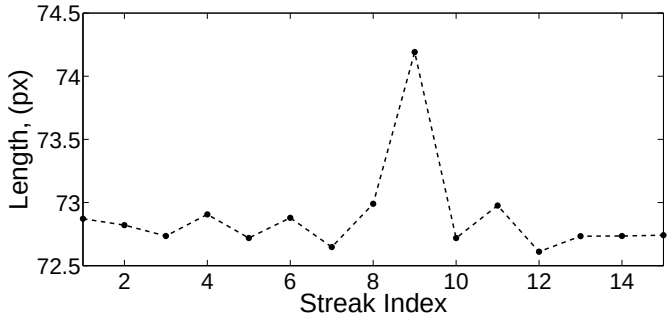

Figure 4.3: RSO indicator data for the magnitude of rotation.

Combined with the center of rotation indication, this provides the complete picture of the RSOs apparent motion relative to the background stars. From this data, it can be correctly concluded that streak index 9 corresponds to a non-star object.

Combined Indicator

The previous two indicators consider the complete information available about the rotation which the spacecraft underwent during image generation – the rotation axis and the angular displacement about that axis. In order to simplify the process of identifying outlying data, a third indicator is proposed. The OLAE computation used for rate estimation considers the complete rotational data to provide an estimate of both the rotational axis and the displacement. For this reason, it can be considered a combined indicator which joins the previous two processes into a single operation.

Residuals were generated for the complete data in the example scenario and plotted in Figure 4.4.

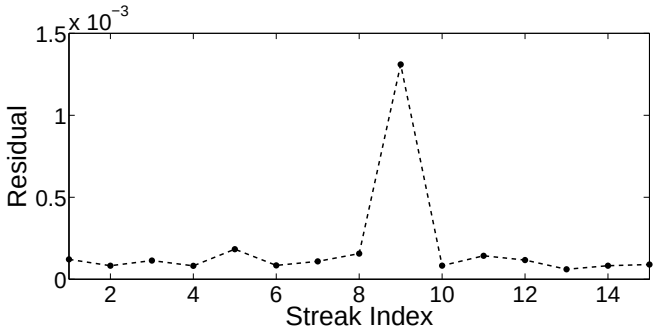

Figure 4.4: RSO indicator data for OLAE residuals.

Streak index 9, the RSO, corresponds to an outlier 3.6σ from the mean, marking a successful identification. The added benefit of choosing this indicator over the previous two is that much of the processing is already necessary for rate estimation. Producing an indicator requires only the batch computation of the residuals from Equation 4.7.

Simulation Results and Discussion

Two simple, proof-of-concept examples are presented, followed by a large-scale simulation. The first example addresses the case of producing feedback during a maneuver. The image produced is seen in Figure 4.5. In this case, the spacecraft is undergoing a maneuver at a constant angular rate while tracking an RSO. Consequently, the RSO is a single point of light in the image, while the surrounding stars are streaking.

Figure 4.5: Sample rate-track image.

The angular rate of the spacecraft is approximately 0.05 rad/s and, combined with an integration time of 0.5 seconds, would cause significant error in traditional trackers. The image was processed to produce a rate estimate; the results are summarized in Table 4.1. The computed rate error was approximately 0.9% and the axis misalignment was 0.65°. As in the previous section, it is possible to reduce this error through an increase in streak length, however there is a trade-off with a smaller number of streaks falling within the image plane and an increased likelihood of overlapping streaks.

Table 4.1: Summary of the rate-track simulation scenario results.

	Estimated	True
Rate	$1.986\ ^\circ/s$	$2.000\ ^\circ/s$
Displacement	0.993°	1.000°
Rotation Axis	$\begin{pmatrix} 0.02 \\ 0.99 \\ 0.00 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$

The second case considered is the demonstrative case which was analyzed previously in the RSO indicator section. This scenario could contain an RSO streak purely by chance or as part of a tracking maneuver which has not yet converged. Regardless, the RSO produces a valid set of endpoints which potentially pollute the data set and cause an increase in error. In situations where only a small number of streaks are discovered, this could contribute to significant error in the resulting rate solution. If the aim is tracking of the RSO, then the current location of the RSO is a valuable addition to the maneuver feedback. The rate-only algorithm was again applied to this image. Figure 4.6 shows the resulting detections side-by-side with the original image. Each discovered endpoint is marked with an "x", while the potential RSO endpoints are designated with a circle.

Figure 4.6: Sample RSO detection during spacecraft slew with constant angular rate.

As can be seen, the RSO is easily separated from the rest of the data set to produce a quality rate estimate and provide feedback data on the RSO location as well. Endpoint error for the RSO is on the order of half a pixel, which corresponds to approximately 40 arcseconds for this camera configuration. Complete results of the scenario are summarized below. Rate error is approximately 0.67% with a rotation axis misalignment of $0.2°$.

Table 4.2: Summary of the constant slew simulation scenario results.

	Estimated	True
Rate	$0.0497\ rad/s$	$0.0500\ rad/s$
Displacement	0.0248 rad	0.025 rad
Rotation Axis	$\begin{pmatrix} -0.003 \\ 0.999 \\ 0.000 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$

As a final demonstration of the algorithms presented here, a 10-orbit, approximately 14-hour simulation was conducted. Images were only generated when the RSO was in view of the camera and not eclipsed by the Earth. Initial orbital parameters were chosen for both the camera-equipped spacecraft and the RSO.

Table 4.3: 10-orbit simulation orbit parameters.

	a	e	i	Ω	ω	f
Camera	6748 km	0.0	$28.5°$	$0°$	$0°$	$0°$
Spacecraft	7078 km	0.0	$89°$	$0°$	$30°$	$120°$

The camera-equipped spacecraft was also given a constant angular rate of 0.05 rad/s about the focal-plane y-axis with an initial quaternion attitude of $\vec{q} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$. A sample four-image sequence generated by the simulation is seen in Figure 4.7. The RSO can be seen moving from the center-left portion of the image gradually upwards throughout the sequence while all other streaks move directly from left to right. A more realistic visible magnitude of $M_v = 4$ was chosen,

creating a much dimmer RSO. For this reason, the contrast in Figure 4.7 has been enhanced to increase visibility of the image contents.

Figure 4.7: Four sequential RSO observation opportunities (contrast enhanced).

Over the entire 10-orbit duration, 14 observation opportunities were produced. Rate estimation produced an average error of less than 1% with an average axis misalignment of $0.4°$. RSO detection rates are seen in the table below. With a strict outlier threshold of 2.5σ, the center of rotation and OLAE methods demonstrated an 85% detection rate. The magnitude of rotation method performed slightly worse with 2 less detections. Only 3 false detections were made, by the center of rotation method, and these were most likely due to the lack of a lower threshold for outlier detection. In order to prevent false-positives, it is necessary to characterize the the performance with no RSOs or overlapping streaks in view to produce a lower-bound for the expected variance between normal measurements.

Table 4.4: RSO indicator identification results summary.

	Correct	Overlapping Streak	False
Center of Rotation	12	0	3
Magnitude of Rotation	10	2	0
Combined Indicator	12	2	0

CHAPTER 5: BIO-INSPIRED, OPTIMAL ATTITUDE CONTROL AND HARDWARE DEMONSTRATION

Nearly 40 % of past small satellite missions have employed some form of active control, primarily for the purpose of power generation and communication [9]. In some cases, however, it is necessary to also actively control the satellite for the purpose of pointing a scientific instrument. The benefit of an optimal, active control system, then, is twofold: (i) power generation potential is increased with fast slewing and precise pointing capabilities; (ii) power losses due to employing active control are reduced.

In order to address these power constraints, an optimal path planning method combined with an LQ tracking controller for pico- and nano-scale satellites is detailed here. In the following sections, a bio-inspired, optimal path planning method utilizing the principles of virtual motion camouflage is presented, as well as an LQ tracking controller used to track the desired path and provide some disturbance rejection. Simulation results are presented to verify the principles and provide predictions for expected performance. Finally, the method is implemented in C on the KnightCube pico-scale satellite testbed and verified for expected functionality. Performance characteristics are calculated and discussed.

Method Formulation

Optimal Path Planning

A performance function is chosen to minimize total power consumption for any maneuver, and takes the form of Equation 5.1.

$$J = \frac{1}{2}\int_0^{t_f} I^2 R dt \tag{5.1}$$

Here, I is the current applied to the torque coil and R is the resistance of the torque coil used to actuate the system. The rigid body satellite dynamics are

$$\begin{aligned} \dot{\omega} &= J^{-1}\left(\omega \times J\omega - T\right) \\ \dot{q} &= \frac{1}{2}\begin{pmatrix} \tilde{\omega} & \omega \\ -\omega & 0 \end{pmatrix} q \end{aligned} \tag{5.2}$$

Where ω is the set of angular rates and q is the quaternion representation of the attitude. $\tilde{\omega}$ is the skew-symmetric matrix. The magnetic coil input torque to the system is equal to the cross product of the area vector, A, and the magnetic field vector, B, multiplied by the current and the number of turns, n, as shown in Equation 5.3.

$$T = InA \times B \tag{5.3}$$

Some simplifying assumptions are made in order to reduce these equations to simpler forms. The inertia matrix, J, contains negligible off-diagonal components, therefore only the diagonal components are used. Additionally, since the test bed is constrained to one-dimensional motion, $\omega = \begin{bmatrix} 0 & \omega_2 & 0 \end{bmatrix}^T$. The first equation of 5.2, then simplifies to $T = J_{22}\dot{\omega}_2$. Combining this with

Equation 5.3 leads to

$$T = InA\,|B|\cos\theta = J_{22}\dot{\omega}_2 \tag{5.4}$$

A relation for the current can then be found explicitly.

$$I = \frac{J_{22}\dot{\omega}_2}{nA\,|B|\cos\theta} \tag{5.5}$$

From the definition of the quaternion attitude, $q_2 = e_y \sin\left(\frac{\theta}{2}\right)$, where e_y is the principle rotation axis and θ is the rotation about that axis. In this case, since motion is restricted to a single axis, this can be simplified to $q_2 = \sin\left(\frac{\theta}{2}\right)$. Using the trigonometric identity, $\cos(2\theta) = 1 - 2\sin^2(\theta)$, it can be found that

$$I = \frac{J_{22}\dot{\omega}_2}{nA\,|B|\,(1 - 2{q_2}^2)} \tag{5.6}$$

Substituting this into 5.1 leads to

$$J = \frac{1}{2}\int_0^{t_f} \left(\frac{\dot{\omega}_2 J_{22}}{nA\,|B|\,(1 - 2{q_2}^2)}\right)^2 Rdt \tag{5.7}$$

The Pseudo-Spectral discretization method is applied in order to reduce the number of nodes required for a precise solution [37].

$$J = \frac{t_f R}{2}\left(\frac{J_{22}}{nA}\right)^2 \sum_{i=0}^{N} \left(\frac{\dot{\omega}_{2,i}}{|B|\,(1 - 2{q_{2,i}}^2)}\right)^2 W_i \tag{5.8}$$

W_i is the set of Pseudo-Spectral weights applied to each discretized step. Additionally a set of times, $T_{1\ldots N}$, corresponding to each node are generated. Since $q = \begin{bmatrix} 0 & q_2 & 0 & \sqrt{1 - {q_2}^2} \end{bmatrix}$, it can be found that

$$\dot{q}_2 = \frac{\omega_2}{2}\sqrt{1 - {q_2}^2} \tag{5.9}$$

To further reduce the complexity of the problem, a bio-inspired control law is applied to quaternion attitude.

$$\begin{aligned} q_{2,i} &= q_r + \nu \left(q_{p,i} - q_r \right) \\ \dot{q}_{2,i} &= \dot{\nu}_i \left(q_{p,i} - q_r \right) + \nu_i \dot{q}_{p,i} \end{aligned} \tag{5.10}$$

Here, q_r is a reference point chosen to be beyond the range of the expected path, $q_{p,i}$ is the user-defined prey motion, and ν_i is the path control parameter (PCP). The PCP relates the planned path, referred to as the aggressor path, to the prey path. The benefit of this parameterization is that it allows a reduction of nodes to be optimized. The prey path is chosen so that $q_{2,1} = q_{p,1}$ and $q_{2,N} = q_{p,N}$ where N is the number of nodes. Therefore, $\nu_1 = \nu_N = 1$. Additionally, since the initial angular rate is known, and the final angular rate is a known desired value, it is possible to solve for ν_2 and ν_{n-1} [38].

$$\begin{bmatrix} \nu_2 \\ \nu_{N-1} \end{bmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}^{-1} \begin{bmatrix} \dot{q}_{2,1} - \nu_0 \dot{q}_{p,1} - A_1 \left(D_{0,0}\nu_0 - \sum_{k=2}^{N-2} D_{0,k}\nu_k - D_{0,N}\nu_N \right) \\ \dot{q}_{2,N} - \nu_0 \dot{q}_{p,N} - B_1 \left(D_{N,0}\nu_0 + \sum_{k=2}^{N-2} D_{N,k}\nu_k + D_{N,N}\nu_N \right) \end{bmatrix} \tag{5.11}$$

The coefficients from above can be found analytically to be

$$\begin{aligned} a_{11} &= D_{0,1} A_1 \\ a_{12} &= D_{0,N-1} A_1 \\ a_{21} &= D_{N,1} B_1 \\ a_{22} &= D_{N,N-1} B_1 \end{aligned} \tag{5.12}$$

where $A_1 = q_{p,1} - q_r$ and $B_1 = q_{p,N} - q_r$. From this solution, the dimensionality of the optimization problem is reduced by 4. By substituting 5.10 into 5.9, a relation for ω_i can be found.

$$\frac{\omega_{2,i}}{2} = \frac{\dot{\nu}_i \left(q_{p,i} - q_r\right) + \nu_i \dot{q}_{p,i}}{\left[1 - q_r - \nu_i \left(q_{p,i} - q_r\right)\right]^{1/2}} \tag{5.13}$$

To produce $\dot{\omega}$ for 5.8, the D'Legendre differentiation matrix can be applied: $\dot{\omega} = D \cdot \omega$ [37]. The set of $N - 4$ PCPs can then be optimized with any nonlinear optimization package. For the purposes of the KnightCube testbed, an open-source library, NLopt, was used to simplify development [39]. The optimization is constrained to

$$|I_i| \leq I_{max} \tag{5.14}$$

due to the electrical limitations of the actuation circuit, and

$$|q_{2,i}| \leq 1 \tag{5.15}$$

to maintain the quaternion normalization requirement, ${q_2}^2 + {q_4}^2 = 1$. The maximum producible current by the actuation circuit for this hardware test is defined as $I_{max} = 0.148$ A.

Sequential optimizations are performed in a finite horizon framework. The initial optimization calculates a path solution for time $t = 0$ to $t = t_f$, considering the initial conditions. Each following optimization calculates a path solution by considering the current state as the initial condition and the current time as $t = 0$. The "horizon" is then a varying final time which advances with each successive iteration. The time between each iteration and the final time, t_f, is a user-defined parameter which is tuned for desired performance characteristics.

Path Tracking Controller

A linear-quadratic (LQ) tracking controller is applied in order to ensure tracking of the desired path and provide some robustness to noise and other disturbances present in the system. First, an input-output linearization was derived for the testbed system in order to simplify the dynamics for the LQ controller. First, the error quaternion between the true quaternion, $\vec{q}$, and the desired quaternion, $\vec{p}$, is considered [27].

$$q_E = \begin{bmatrix} p_4 & p_3 & -p_2 & p_1 \\ -p_3 & p_4 & p_1 & p_2 \\ p_2 & -p_1 & p_4 & p_3 \\ -p_1 & -p_2 & -p_3 & p_4 \end{bmatrix} \begin{bmatrix} -q_1 \\ -q_2 \\ -q_3 \\ q_4 \end{bmatrix} \tag{5.16}$$

For the single-axis case, this can be simplified to

$$q_E = \begin{pmatrix} 0 \\ -p_4 q_2 + q_4 p_2 \\ 0 \\ p_4 q_4 + q_2 p_2 \end{pmatrix} \tag{5.17}$$

This is the 1-dimensional error quaternion. Recall the relation between quaternions and Euler angles from [27],

$$\begin{aligned} q_{E,2} &= a_y \sin\left(\frac{e}{2}\right) \\ q_{E,4} &= \cos\left(\frac{e}{2}\right) \end{aligned} \tag{5.18}$$

Where a_y is the y component of the rotational axis and ϕ is the angular displacement. In the 1-D case, a_y simple reduces to 1. Since, through trigonometric identities, the following is true,

$$\sin e = 2 \sin\left(\frac{e}{2}\right) \cos\left(\frac{e}{2}\right) \tag{5.19}$$

the error may be approximated as

$$e \approx 2 q_{E,2} q_{E,4} \tag{5.20}$$

Since the LQ controller will be tracking a pre-generated path, the small angle approximation is valid here. The path will be re-generated frequently, with the initial desired condition at the current location. Therefore, the LQ controller should not have the opportunity to deviate to large angular errors. In the case that a large perturbation causes large angular error, the path will be re-generated quickly at the new location. The error approximation may now be expanded to

$$e = 2\left[p_2 p_4 \left(1 - 2{q_2}^2\right) - q_2 q_4 \left(1 - 2{p_2}^2\right)\right] \tag{5.21}$$

Several intermediate terms are defined in order to simplify the following derivation and final results.

$$\begin{aligned} D_q &\equiv q_2 q_4 \\ D_p &\equiv p_2 p_4 \\ C_q &\equiv 1 - 2{q_2}^2 \\ C_p &\equiv 1 - 2{p_2}^2 \end{aligned} \tag{5.22}$$

Equation 5.21 then simplifies to

$$e = 2\left(D_p C_q - D_q C_p\right) \tag{5.23}$$

The first derivative, $\dot{e}$, is calculated

$$\dot{e} = -\omega_2 \left(4D_pD_q + C_pC_q\right) + 2A_pC_q + 8p_2\dot{p}_2D_q \tag{5.24}$$

where $A_p \equiv \dot{p}_2p_4 + p_2\dot{p}_4$. The second time derivative is also found

$$\ddot{e} = -\dot{\omega_2} \left(4D_pD_q + C_qC_p\right) - \omega_2 \left(4A_pD_q + 2\omega_2C_q - 4p_2\dot{p}_2C_q - 2\omega_2D_qC_p\right) \tag{5.25}$$

From 5.6, an explicit relation between the control input, I, and $\dot{\omega}_2$ exists, and a solution for I is found in the form of Equation 5.26.

$$I_{eq} = \frac{1}{\alpha}\left(v - \beta_1 - \beta_2 - \beta_3 - \beta_4\right) \tag{5.26}$$

where v is the control input to the equivalent linear system. The coefficients α and $\beta_{1\ldots4}$ are detailed below.

$$\begin{aligned}
\alpha &= \frac{1}{J_2}\left(4D_pD_q + C_pC_q\right)|B|\,C_q \\
\beta_1 &= -2{\omega_2}^2\left(D_pC_q - D_qC_p\right) \\
\beta_2 &= -4\omega_2\left[2A_pD_q + p_2\dot{p}_2\left(4D_q - C_q\right)\right] \\
\beta_3 &= 2\left(\ddot{p}_2p_4 + 2\dot{p}_2\dot{p}_4 + p_2\ddot{p}_4\right)C_q \\
\beta_4 &= 8\left(\dot{p}_2^2 + p_2\ddot{p}_2\right)C_q
\end{aligned} \tag{5.27}$$

The equivalent linear system then reduces to

$$\begin{aligned}
\dot{x} &= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} v \\
y &= \begin{bmatrix} 0 & 1 \end{bmatrix} x
\end{aligned} \tag{5.28}$$

with

$$x = \begin{bmatrix} \dot{e} \\ e \end{bmatrix} \tag{5.29}$$

An LQ regulator is applied to this linearized system in the form

$$v = -Kx \tag{5.30}$$

where K is the control gain

$$K = R^{-1}B^{T}P \tag{5.31}$$

and P is the solution to the algebraic Riccatti equation [40],

$$A^{T}P + PA - PBR^{-1}B^{T}P + Q = 0 \tag{5.32}$$

Q and R are user-defined control gains. Since the linearized system is time-invariant, a solution for P is found offline to reduce the required computational demand. For the purposes of the test bed demonstration, gain values of $R = 1$ and $Q = 20 \cdot [I_{2\times 2}]$ were used.

To interface between the planned path and the LQ controller, a simple interpolation method was adapted to take advantage of the derivatives available through the Pseudo-Spectral discretization and reduce necessary processing time. Any interpolation method may be used, however, more precise methods may be needed to maintain precision for lower update rates. The desired path for time t is computed from the VMC-generated path through this process.

Hardware Testbed

Pico-Scale Satellite Prototype

The KnightCube CubeSat prototype is intended to be a testbed for magnetic control methods and software architecture, as well as a foundation for future design iterations. The prototype is a 10 cm, completely self-contained pico-scale satellite which conforms to the 1U CubeSat specification [1]. Through these specifications, the prototype is compatible with the Cal-Poly P-Pod pico-sat deployer, which is the standard for deploying CubeSats as secondary payloads [2]. The prototype is constructed entirely from Aluminum 6061 and, with hardware, is less than 1 kg. Complete structural properties are listed in Table 5.

Table 5.1: KnightCube structural properties.

Material	Aluminum 6061
Dimensions (cm)	$10 \times 10 \times 10$
Mass (g)	611
Inertia (kg $\cdot$ m^2)	$J = \begin{pmatrix} 1.10 & 0.00 & 0.00 \\ 0.00 & 1.05 & 0.00 \\ 0.00 & 0.00 & 0.99 \end{pmatrix} \times 10^{-2}$

The internal hardware falls under three major subsystems: the inertial measurement unit, the central processing unit, and the power management circuitry. Inexpensive, commercial off-the-shelf (COTS) solutions were used for all subsystems. None of the hardware in the testbed is intended for on-orbit operation, rather it is intended to serve as a functional analogue for future space-rated hardware. A breakdown of the connections and data flow within the prototype is illustrated in Figure 5.1 and an image of the complete prototype is seen in Figure 5.2.

The inertial measurement unit (IMU) is a VectorNav VN-100 attitude and heading reference system (AHRS). This device contains a three axis gyro for detecting angular rates, a three axis accelerom-

eter for detecting linear accelerations, and a three axis magnetometer for measuring magnetic field strength and orientation. These three sensors are fused onboard and can be output through an RS-232 serial data connection at 115200 bits per second with an update rate of up to 200 Hz [41].

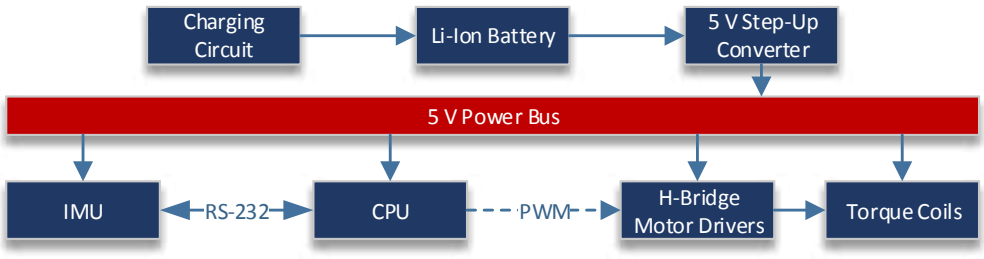

Figure 5.1: KnightCube prototype complete system diagram.

The central processing unit (CPU) is an ICOP VDX-6315 pc-on-a-board [42]. This CPU contains an 800 MHz processor and 512 MB of on-board flash memory. The flash memory was loaded with a light-weight, open source operating system called X-Linux. The operating system has an 11 MB footprint and boots quickly in approximately 5 seconds [43]. Though using a complete operating system uses additional resources, the open-source, unix-based nature of the operating system significantly shortens development time. Since the primary goal of a CubeSat is, in general, to produce a low-cost experimental testbed, open-source software is ideal as it has the benefit of being completely free and widely available.

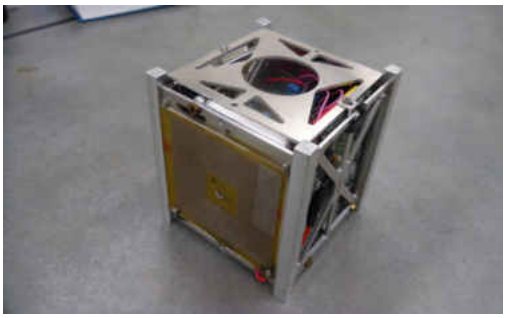

Figure 5.2: Complete KnightCube pico-scale satellite prototype.

Actuation is performed through the use of up to three orthogonal magnetic torque coils. The coils

are simple, printed circuit boards containing dual-sided coils for a total of 80 turns. Each coil has a total resistance between 15 and 25 ohms. The CPU handles actuation of the torque coils through the use of pulse-width modulation. Pulse-width modulation (PWM) is essentially a method for varying the effective voltage applied to a circuit by rapidly switching the supply voltage on and off. For each user-determined period — 20 ms in this case — the supply voltage is switched on for a certain percentage of the period, known as the duty cycle. For the remaining duration, the applied voltage is zero. The equivalent voltage applied to the circuit is equal to the duty cycle multiplied by the supply voltage.

$$V_{equivalent} = \frac{T_{on}}{T_{on} + T_{off}} V_{supply} \tag{5.33}$$

Since the CPU itself is incapable of supplying large currents through a single pin, an intermediate chip is required. Two SN754410 H-Bridge motor driver chips — each capable of controlling two independent torque coils — serve this purpose. An H-Bridge essentially accepts a supply voltage and two control voltages as input. The state of the two inputs determines the state of two output pins, each of which may either be equal to the supply voltage or connected to ground. H-Bridge control pins are connected to two PWM pins on the CPU to allow relaying of the signal in either current-flow direction. Table 5 contains all of the possible states for the H-Bridge inputs as well as the corresponding output states to the coils [44].

Table 5.2: Truth table for the SN754410 H-Bridge chip connected to a magnetic torque coil.

Input A	Input B	Coil Voltage	Current Direction
H	H	0 V	—
H	L	V_{supply}	CW
L	H	V_{supply}	CCW
L	L	0 V	—

Here, a high input voltage is denoted as “H”, while “L” designates a low input voltage. The current directions are denoted as clockwise (CW) and counter-clockwise (CCW) in reference to the coil when viewed externally in the testbed. After all losses within the circuit, V_{supply} is equal

to 3.4 volts. Considering the minimum coil resistance of approximately 17 ohms, the maximum producible current is approximately 0.2 amps. Since each coil varies slightly in resistance, measurements have been taken for each coil and are used within the code to ensure accurate current output. For the purposes of the hardware demonstration in the following sections, the active coil has been found to have a maximum current of 0.148 A.

The power system consists of three components: a charging circuit, a step-up converter, and a lithium-ion battery. The charging circuit is a Texas Instruments BQ24070 complete lithium-ion battery management system [45]. The battery is connected through this circuit, and charged via a 5 V external cable. A step-up converter converts the standard 3.7 V battery supply voltage to a 5 V power bus for powering each of the subsystems. A 1.8 A·h lithium-ion battery is used for maintaining the systems during testing.

Helmholtz Coil

A Helmholtz coil is employed within the testbed in order to increase the maximum producible torque by the CubeSat and help the prototype overcome some of the disturbances present in the system. For example, the mechanism for suspending the prototype for testing is a thin thread attached to two opposite faces. While this allows movement in a single axis, there is a small torque which is produced by the thread, which increases as angular displacements increase. With only the ambient magnetic field, the torque coils do not produce enough torque to overcome this perturbation. From 5.3 it can be seen that increasing either the current or the magnetic field will cause an increase in the resulting torque. Since the system produces only a fixed amount of current, the Helmholtz coil is needed to increase the magnetic field, and subsequently the produced torque. An equation for the magnetic field through the center of a Helmholtz coil is seen in Equation 5.34

[46].

$$B = \left(\frac{4}{5}\right)^{3/2} \frac{\mu_0 n I}{R} \tag{5.34}$$

Here, $\mu_0 = 4\pi \times 10^{-7}\ {}^{T \cdot m}/_{A}$ is the vacuum permeability, n is the number of turns in the coil, I is the current through the coil, and R is the radius of the coil. The actual coil radius is approximately 15 cm and has been measured to produce 9.98 gauss through the center at a current of 2.19 A. The IMU used in the prototype has a range of ± 6 gauss, so the input current is set to A, which creates an expected magnetic field strength of 4.5 gauss, which is approximately 9 times Earth's ambient magnetic field strength of 0.5 gauss. The complete Helmholtz Coil is seen in Figure 5.3.

Figure 5.3: KnightCube testbed Helmholtz coil.

The prototype from the previous section is suspended in the center of the Helmholtz coil, and a fixed magnetic field is produced for testing.

Simulation Results

A MATLAB simulation was prepared in order to produce an initial tuning offline and verify the expected performance of the method detailed in the previous section. First, the performance function from 5.8 and the constraints from 5.14 and 5.15 were implemented. The MATLAB constrained nonlinear minimization solver, fmincon, was used in order to produce a single planned path. Initial

conditions were chosen as $q_{2,initial} = 0.4$ and $\omega_{2,initial} = 0.015$ rad/s, with desired final conditions of $q_{2,final} = -0.3$ and $\omega_{2,final} = 0$ rad/s. The reference point, q_r, was chosen to be 2, which is beyond the maximum possible value of q_2, thereby avoiding a degenerate case for 5.10 where $q_{p,i} - q_r = 0$ and ν has no effect. Final time was set as $t_f = 10$ s with 10 discrete nodes. The PCP initial guess was set as $\nu_{1...N} = 1$, providing an initial path equal to the prey motion.

The prey motion, q_p, was chosen as a continuous third order polynomial to provide a smooth initial guess to the optimization.

$$q_p = C_0 + C_1 t + C_2 t^2 + C_3 t^3 \tag{5.35}$$

The derivative of the prey motion is of the form

$$\dot{q}_p = C_1 + 2C_2 t + 3C_3 t^2 \tag{5.36}$$

Here, $C_{0...3}$, can be found analytically based on the initial conditions for q_2 and ω_2 as well as the two desired final conditions. For $t = 0$, C_0 and C_1 are easily available.

$$\begin{aligned} C_0 &= q_{2,initial} \\ C_1 &= \dot{q}_{2,initial} \end{aligned} \tag{5.37}$$

The value for $\dot{q}_{2,initial}$ can be computed from the angular rate and attitude through the relation in 5.9. The final two coefficients were found by choosing $t = t_f$, $q_p = q_{2,desired}$ and $\dot{q} = \dot{q}_{2,desired}$, and solving analytically.

$$\begin{aligned} C_2 &= \frac{1}{{t_f}^2} \left[3 \left(q_{2,desired} - C_0 \right) - \left(2C_1 + \dot{q}_{2,desired} \right) t_f \right] \\ C_3 &= \frac{1}{3{t_f}^2} \left(\dot{q}_{2,desired} - C_1 - 2C_2 t_f \right) \end{aligned} \tag{5.38}$$

Equations 5.35 and 5.36 were evaluated at $T_{1...N}$ to generate the discretized prey motion, $q_{p,i}$. Figures 5.4 – 5.6 illustrate the results produced in this simple initial simulation. It can be seen the solver successfully produced a smooth solution very near the prey motion. Deviations are likely caused by the variation in the effective magnetic field during a maneuver.

The control authority of a torque coil necessarily varies with attitude; therefore the favors actuation during periods where control authority is highest. Figure 5.6 illustrates this effect in the usage of additional current in the early portion of the simulation. It can also be seen that the entire desired current profile is within the 0.148 amp constraint.

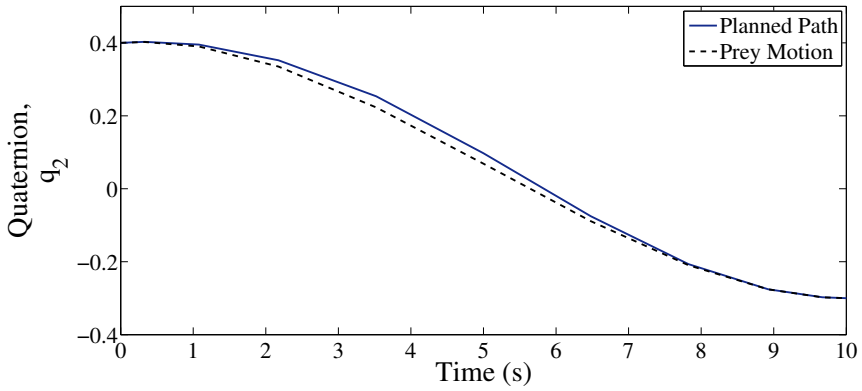

Figure 5.4: Sample optimized path for $q_{2,initial} = 0.4$, $q_{2,desired} = -0.3$ $\omega_{2,initial} = 0.0015$ rad/s, and $\omega_{2,desired} = 0$ rad/s with $t_f = 10$ s.

It is important to note that, though the results here are based on the actual dynamics, they do not include sensor noise or the perturbations present within the testbed. Therefore, in a more realistic situation, the resulting path will be significantly different.

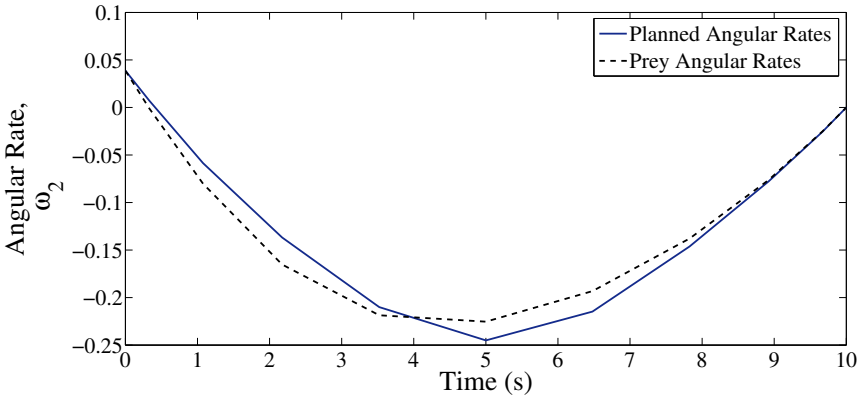

Figure 5.5: Sample angular rate profile calculated from the optimized path.

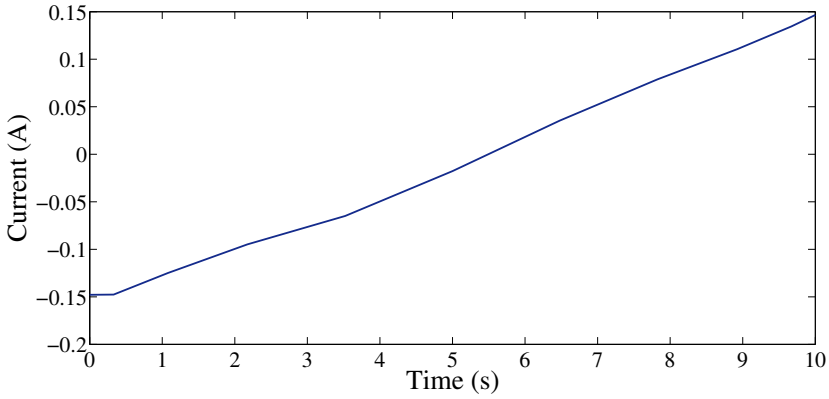

Figure 5.6: Required current profile calculated from the optimized path.

In order to verify the performance in a more realistic environment accounting for both noise and torque uncertainty, a continuous Simulink model was created, and a full-system simulation was conducted. Two primary components the optimal path planner and the LQ tracking controller were implemented. The optimal path was planned initially, and then re-planned in 5 second intervals. The LQ tracking controller operated at 40 Hz, acting to track the most recently generated planned path. The parameters for the path optimization, including the initial conditions, were the same as described for the previous simulation. Additionally, in 100 second intervals the desired

attitude was incremented by 0.1 for a total of 7 step maneuvers. The control gains for the tracking controller were chosen as $R = 1$ and $Q = 20 \cdot [I_{2\times 2}]$. The Riccatti equation was solved off-line to reduce required computational time. The results of the simulation can be seen in Figures 5.7 – 5.8.

Realistic noise characteristics, according to what would be expected from the hardware test bed, were included in this simulation. Noise in the angular velocity and quaternion measurements were modeled as $\omega_2 = \hat{\omega}_2 + \Delta\omega$ and $q_2 = \hat{q}_2 + \Delta q$, where $\Delta\omega$ and Δq are the uncertainties in the angular velocity and the quaternion, respectively. From the datasheet for the IMU in the KnightCube testbed, these values can be found to be bound by $|\Delta\omega| \leq 1.75\times10^{-}4$ rad/s and $|\Delta q| \leq 1.75\times10^{-}3$ [41]. Uncertainty in the torque coil current was modeled similarly as $I = \hat{I} + \Delta I$ with $|\Delta I| = 3.5\times10^{-}3$ to account for the precision limitations of the pulse-width modulation actuation method.

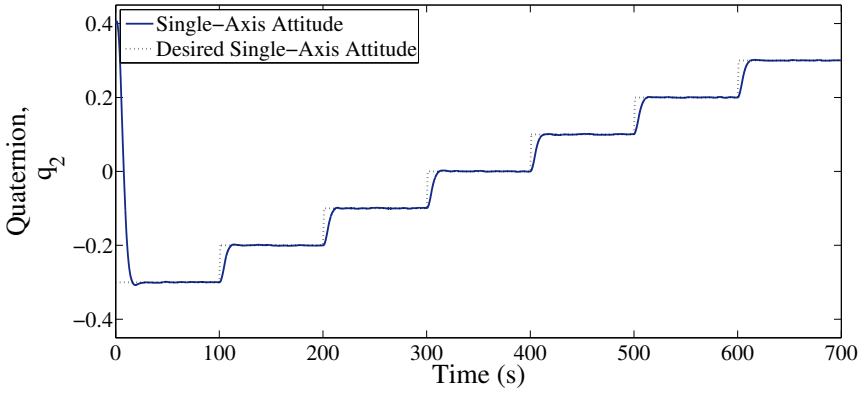

Figure 5.7: Attitude results for 700 s step simulation considering sensor noise and torque uncertainty.

It can be seen from Figure 5.8 that the required current remains within the 0.148 amp limit. A saturation value was applied within the simulation to enforce the maximum current constraint.

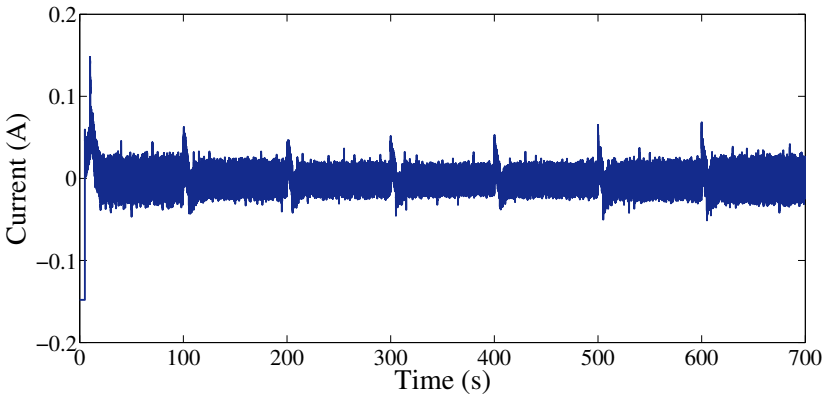

Figure 5.8: Desired current profile for a complete 700 s simulation.

Hardware Results and Discussion

The algorithm detailed in the previous sections was implemented completely on the KnightCube testbed hardware in C. Multiple iterations were then performed in order to tune the LQ tracking controller and improve performance. The testbed was initially set to approximately $q_{2,initial} = 0.2$ manually and released with a low initial velocity, $\omega_{2,initial} = -0.1$ rad/s. Initially, $q_{2,desired}$ was chosen to be -0.3. In 100 second intervals, $q_{2,desired}$ was incremented by 0.1. The complete demonstration was run for 700 seconds. The LQ tracking controller tuning parameters were chosen to be identical to those used in the previous simulations, $R = 1$ and $Q = 20 \cdot [I_{2\times 2}]$. t_f for the optimization was chosen to be 10 seconds. Complete results from the hardware demonstration can be seen in Figures 5.9 – 5.11.

The controller was successfully able to complete 7 attitude maneuvers from $q_2 = -0.3$ to $q_2 = 0.4$. Peak steady-state error of approximately 3.6 degrees is seen in the first 100 second interval. The differences in steady-state error seen between step maneuvers are due to perturbations present within the testbed. The suspension mechanism caused a small torque on the prototype throughout the demonstration. This torque increases with greater rotational displacement, leading to a greater

perturbation at the more extreme displacements. Such a perturbation would not be present in the low Earth orbit environment for which the control method is intended. Steady state error is as low as 1.25 degrees in the fifth 100-second interval.

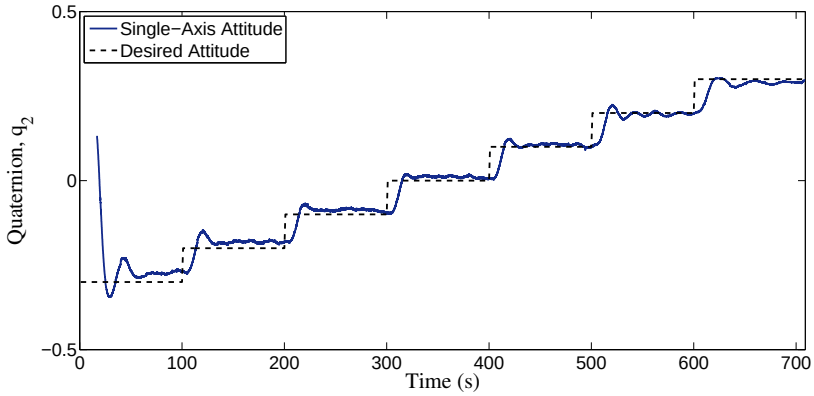

Figure 5.9: Attitude results from 700 s hardware step demonstration.

The control current used for actuation can be seen to be well within the saturation limit of $I_{max} \leq 0.148$ A. Figure 5.10 shows the complete set of data for the entire duration.

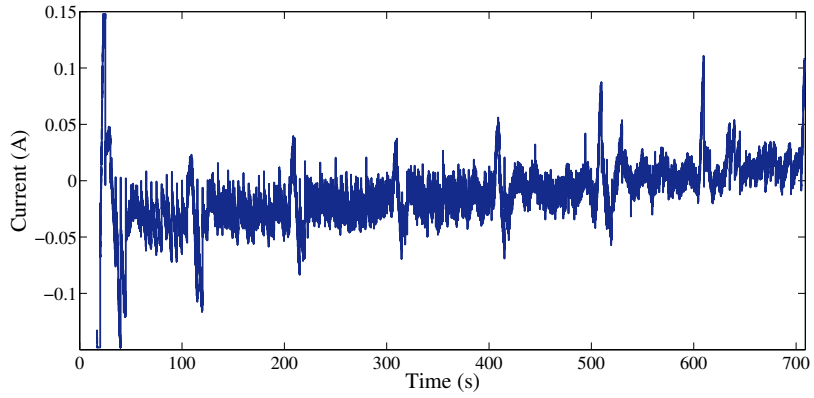

Figure 5.10: Torque coil current output over 700 s hardware step demonstration.

Due to the performance metric outlined in 5.1, it is important to consider the power consumption

profile of the demonstration in evaluating the performance of this control method. Figure 5.11 contains the entire power consumption profile for the duration of the test. The peak power consumption can be seen to be approximately 0.5 watts. The average power consumed for the entire test was on the order of 0.10 W. This power consumption is significantly lower than both PID and adaptive control methods which have been implemented on the same testbed, with the PID controller showing an average power consumption of 0.15 W and the adaptive controller using 0.66 W.

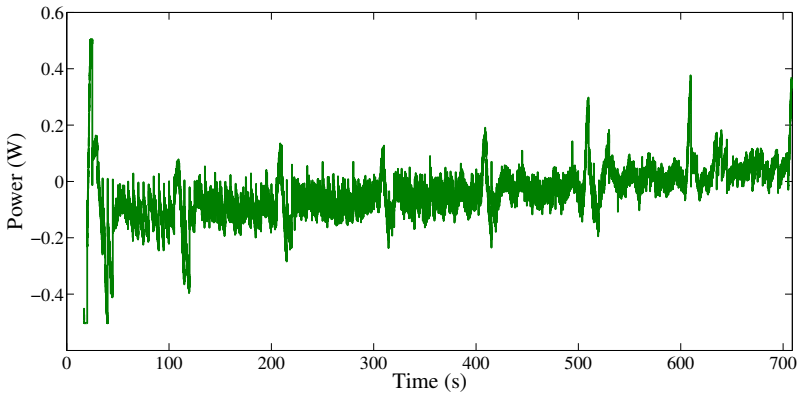

Figure 5.11: Power consumption over 700 s hardware step demonstration.

In addition to low power consumption, a secondary performance goal for this method was fast slewing performance for large attitude maneuvers. Table 5 shows the calculated performance values for the hardware test. All values are given as upper bounds, due to the variation present between each individual step response. It is expected that, without perturbations introduced by the suspension mechanism, the results would be further improved. Regardless, the method is capable of performing fast slewing maneuvers and may be combined with more robust pointing algorithms for precision pointing and disturbance rejection.

Table 5.3: Calculated performance parameters for optimal control demonstration.

Steady-State Error	$< 4.5^{\circ}$
Overshoot	$< 20\%$
Rise Time	< 8 s
Settling Time	< 30 s

CHAPTER 6: CONCLUSIONS

The work of this thesis is intended to provide two methods which have the potential of expanding the capabilities of both small satellites and larger-scale spacecraft.

Summary of Work

The star camera model detailed and demonstrated in chapter two serves as a simple, concise framework for the simulation of star camera systems. The inclusion of actual star data from the Hipparcos star catalog, in addition to realistic visible brightness and noise, produces a quality approximation of true star camera images. Future work will be based on the same model, with the possibility of expansion to include additional levels of fidelity.

The algorithm presented in chapter three is an attempt to extend the capabilities of low-cost star camera systems as well as existing, currently on-orbit systems. The process allows attitude and rate feedback data to be retrieved in situations where current star cameras are incapable. The theoretical single endpoint accuracy of approximately $\frac{1}{10}$ of a pixel approaches the accuracy of current star cameras, however additional work is required to reduce the remaining bias completely. For systems with less stringent pointing needs, this algorithm in its current state provides a viable solution for the handling of streaked star images. A conference paper has been presented and published in the proceedings of the 2013 SPIE Security, Defense, and Sensing conference [47].

Chapter four's rate-only formulation of the image processing algorithm extends the capabilities of chapter three and reduces the overall computation required for the case where only rate estimates are required. Additionally, it enables rate feedback during maneuvers as well as the ability to track an RSO with no additional feedback necessary. The produced rate estimates within 1% of the

truth provide an adequate solution for use in feedback, and it is expected that it will be possible to improve this further with consideration of the bias caused in endpoint detection. The detection of outlying behavior in the data set offers some protection against overlapping streaks and enables the discovery of an RSO within the data.

The optimal path generation algorithm discussed in chapter five serves to improve the capabilities of pico- and nano-scale satellites. Due to the significant power restrictions inherent in the form factor of these spacecraft, low-power, precise attitude control is absolutely essential to the success of many past and future scientific missions. Additionally, the rapid slewing maneuvers performed in under 30 seconds potentially maximize power generation opportunities, if the system is applied to solar panel pointing. The 0.1 W average power consumption makes up only a small fraction of the average 1 W/kg produced by small satellites, ensuring maximum power available for other mission-critical components.

Future Work

The work presented here is a step in the process of improving small satellite capabilities. In order to advance these algorithms to maturity, additional consideration must be given to certain aspects of the work.

The primary remaining issue with the algorithm presented in chapter three is the endpoint bias caused by the corner detection algorithms. Future work will consider more advanced image processing techniques in attempt to preserve the structure of the streaks to return a highly localized sub-pixel endpoint location. Some other concepts have been considered, however, such as a filtering process which operates over several images to characterize the bias and reduce the error over time. Though the bias is not observable from the endpoint measurements alone, the center point of

the streak, occurring at time $\frac{T}{2}$, can be found without bias. Combining this feature with the previously discovered endpoint data may provide an estimate of the bias and subsequently a correction. Though a streamlined method entirely based in image processing techniques is desired, this may provide an adequate solution.

An additional issue in the work of chapter three is the large variance present in the resulting data. There are many contributing factors to deviations between subsequent measurements, including overlapping or corrupted streaks and false star identifications. Future work should attempt to characterize and further minimize this inherent variance in the process. The work of chapter four is a first step in that direction as it allows filtering of overlapping streak geometry and RSOs, which potentially pollute the data set.

The optimal path generation method discussed in chapter five provides a foundation for low-power pico- and nano-scale satellite attitude control, however additional work will be needed for an on-orbit implementation. The method will require expansion to three-axis, with additional consideration for the under-actuated nature of magnetic torque-coil control. An additional actuation method, such as a reaction wheel or control moment gyro, may be required to provide complete three-axis actuation. Additionally, a secondary control algorithm may be implemented for disturbance rejection following large slewing maneuvers produced in the optimal path generation. A secondary method may potentially reduce the steady-state error further, without additional power consumption for the overall maneuver.

LIST OF REFERENCES

[1] S. Lee, A. Hutputanasin, A. Toorian, W. Lan, and R. Munakata. Cubesat design specification, 2009.

[2] W. Lan. Poly picosatellite orbital deployer mk. III ICD, 2007.

[3] M.A. Swartwout. A brief history of rideshares (and attack of the cubesats). In *IEEE Aerospace Conference*, pages 1–15, 2011.

[4] S. Flagg, T. Bleier, C. Dunson, et al. Using nanosats as a proof of concept for space science missions: Quakesat as an operational example. In *Proceedings of the 18th Annual AIAA/USU Conference on Small Satellites*, 2004.

[5] K. Dontchev and et al. M-cubed: University of michigan multipurpose minisatellite with optical imager payload. In *Proceedings of the AIAA Space 2010 Conference & Exhibition*, 2010.

[6] C. Kitts, K. Ronzano, R. Rasay, et al. Flight results from the genesat-1 biological microsatellite mission. In *Proceedings of the 21st Annual AIAA/USU Conference on Small Satellites*, 2007.

[7] K. Woellert, P. Ehrenfreund, A. Ricco, and H. Hertzfeld. Cubesats: Cost-effective science and technology platforms for emerging and developing nations. *Advances in Space Research*, 47(4):663–684, 2011.

[8] D. Selva and D. Krejci. A survey and assessment of the capabilities of cubesats for earth observation. *Acta Astronautica*, 74, 2012.

[9] J. Bouwmeester and J. Guo. Survey of worldwide pico- and nanosatellite missions, distributions and subsystem technology. *Acta Astronautica*, 67(7–8), 2010.

[10] S. Greenland and S. Clark. Cubesat platforms as an on-orbit technology validation and verification vehicle. In *Proceedings of the European Small Satellite Services Symposium*, 2010.

[11] C. C. Liebe. Accuracy performance of star trackers—a tutorial. *IEEE Transactions on Aerospace and Electronics Systems*, 38(2):587–599, 2002.

[12] C.C. Liebe, E.W. Dennison, B. Hancock, R.C. Stirbl, and B. Pain. Active pixel sensor (aps) based star tracker. In *IEEE Aerospace Conference*, volume 1, pages 119–127, 1998.

[13] V. C. Thomas, J. W. Alexander, E. W. Dennison, et al. Cassini star tracking and identification architecture. pages 15–26, 1994.

[14] C.R. McBryde and E.G. Lightsey. A star tracker design for cubesats. In *IEEE Aerospace Conference*, pages 1–14, 2012.

[15] D. Mortari, C. Bruccoleri, S. La Rosa, and Junkins J.L. Ccd data processing improvements. In *Proceedings of the International Conference on Dynamics and Control of Systems and Structures in Space*, 2002.

[16] S. Udomkesmalee, J. Alexander, and A. Tolivar. Stochastic star identification. *Journal of Guidance, Control, and Dynamics*, 17:1283–1286, 1994.

[17] D. Mortari. Search-less algorithm for star pattern recognition. *Journal of Astronautical Sciences*, 45:179–194, 1997.

[18] D. Mortari and B. Neta. k-vector range searching techniques. *Advances in the Astronautical Sciences*, 105:449–464, 2000.

[19] G. Wahba. Problem 65-1: A least squares estimate of spacecraft attitude. *SIAM Review*, 7(3):409, July 1965.

[20] F. L. Markley and D. Mortari. How to estimate attitude from vector observations. In *AAS/AIAA Astrodynamics Specialist Conference*, August 1999.

[21] D. Mortari. Second estimator of the optimal quaternion. *Journal of Guidance, Control, and Dynamics*, 23(5):885–888, 2000.

[22] D. Mortari, F. Markley, and P. Singla. An optimal linear attitude estimator. *Journal of Guidance, Control, and Dynamics*, 30(6):1619–1627, 2007.

[23] E. Høg, C. Fabricius, V. V. Makarov, et al. The Tycho-2 catalogue of the 2.5 million brightest stars. *Astronomy and Astrophysics*, 355:L27–L30, March 2000.

[24] M. A. C. Perryman, L. Lindegren, J. Kovalevsky, et al. The HIPPARCOS Catalogue. *Astronomy and Astrophysics*, 323:L49–L52, July 1997.

[25] J. Crassidis and J. Junkins. *Optimal Estimation of Dynamic Systems*. CRC Press, 2nd edition, 2012.

[26] S. B. Howell. *Handbook of CCD Astronomy*. Cambride University Press, 2006.

[27] J. R. Wertz. *Fundamentals of Astrodynamics and Applications*. D. Reidel, 1978.

[28] D. Vallado. *Fundamentals of Astrodynamics and Applications*. Microcosm Press, Hawthorne CA, 3rd edition, 2007.

[29] P. Escobal. *Methods of Orbit Determination*. R. E. Krieger Pub. Co., Malabar, Florida, 2nd edition, 1976.

[30] C. C Liebe, K. Gromov, and D. M. Meller. Toward a stellar gyroscope for spacecraft attitude determination. *Journal of Guidance, Control, and Dynamics*, 27:91–99, 2004.

[31] C. C Liebe, K. Gromov, and D. M. Meller. Autonomous subpixel satellite track end point determination for space-based images. *Journal of Applied Optics*, 50(22), 2011.

[32] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[33] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, June 1994.

[34] M. Trajkovic and Hedley M. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, 1998.

[35] J. Blom, B.M. ter Haar Romenij, A. Bel, and J.J. Koenderink. Spatial derivatives and the propagation of noise in gaussian scale space. *Jounal of Visual Communication and Image Representation*, 4(1):1–13, 1993.

[36] M. Born and E. Wolf. *Principles of Optics*. Cambridge University Press, 1999.

[37] J. Hesthaven, S. Gottlieb, and D. Gottlieb. *Spectral Methods for Time–Dependent Problems*. Cambridge University Press, 2007.

[38] M. DeVelle, Y. Xu, K. Pham, and G. Chen. Fast relative guidance approach for autonomous rendezvous and docking control. In *SPIE Defense, Security, and Sensing Conference*, April 2011.

[39] S. Johnson. The NLopt nonlinear-optimization package, 2013.

[40] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*, volume 172. Wiley-Interscience New York, 1972.

[41] VectorNav, College Station TX. *VectorNav VN-100(T) User Manual*, 2009.

[42] ICOP. *VDX-6315 / VDX-6315-512 Users Manual*, 2009.

[43] DM&P Group. *DM&P X-Linux*, 2010.

[44] Texas Instruments. *SN754410 quadruple half-h driver*, 1995.

[45] Texas Instruments. *Single-chip Li-Ion charge and system power-path management IC*, 2009.

[46] R. Linhart. A three axis magnetometer for use in a small satellite. In *International Conference of Applied Electronics*, pages 113–116, April 2006.

[47] B. Sease, R. Koglin, and B. Flewelling. Long-integration star tracker image processing for combined attitude-attitude rate estimation. In *Proceedings of SPIE Defense, Security, and Sensing*, 2013.