SPACECRAFT VISUAL NAVIGATION USING APPEARANCE MATCHING AND MULTI-SPECTRAL SENSOR FUSION

A Dissertation
Presented to
The Academic Faculty

By

Christopher Ryan McBryde

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology

May 2018

Copyright © Christopher Ryan McBryde 2018

SPACECRAFT VISUAL NAVIGATION USING APPEARANCE MATCHING AND MULTI-SPECTRAL SENSOR FUSION

Approved by:

Dr. Glenn Lightsey, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Marcus Holzinger
School of Aerospace Engineering
Georgia Institute of Technology

Mr. Chad Frost
Deputy Director of Engineering
Ames Research Center

Dr. Andrew Johnson
Principal GNC Engineer
Jet Propulsion Laboratory

Dr. Eric Johnson
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: May 3, 2018

It is good to have an end to journey towards; but it is the journey that matters, in the end.

Ursula K. Le Guin, The Left Hand of Darkness

To Alexandra Rodriguez. You've been with me every step of the way on this PhD journey and I could not have done it with out you. Also to my parents, Laura and Ryan McBryde. You always believed I could do whatever I dreamed and gave me the love and support to make those dreams a reality.

ACKNOWLEDGEMENTS

This work was sponsored in part by NASA contracts NNX09M51A and NNX15AD26H.

Thank you to my advisor, Dr. Glenn Lightsey, for your all your support and counsel during my graduate career. I cannot believe we met more than eight years ago. It has been a long road through two degrees and as many states, but we made it.

I would like to acknowledge Dr. Andrew Johnson for serving on my PhD committee and for all his assistance on this research topic. Thank you for your mentorship during my two summers at the Jet Propulsion Laboratory as well as your collaboration. This dissertation would not have been possible without your help. I would also like to acknowledge Chad Frost, who served as my collaborator for the NASA Space Technology Research Fellowship. I appreciate your mentorship during my time interning at Ames Research Center and for your membership on my committee. Thank you to Dr. Marcus Holzinger and Dr. Eric Johnson for your instruction as well as for serving on my committee.

Thank you to my colleagues at Georgia Tech: Peter Schulte, Alexandra Long, Andris Jaunzemis, Hisham Ali, and Timothy Murphy. I'm grateful for your help preparing for qualifying exams and my proposal. I would also like to acknowledge Andrew Fear, Terry Stevenson, and Parker Francis for your friendship and support during our transition.

To my parents, Ryan and Laura: thank you for nurturing my curiosity and love of learning, for your constant support and guidance, and for letting a little boy have National Geographic Picture Atlas of Our Universe as a bedtime story. Thank you to my siblings, Katy and Patrick. It's an honor to call myself your brother, and I hope I have inspired and taught you as much as you have me.

And finally, to my fiancée, Laura Alejandra Rodriguez Arellanos: There are no words that are enough to thank you, or room here to list all the times you helped me. You gave me the strength to go on when things got tough and never let me forget that I could do this. And you were right. I love you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

One of the capabilities necessary for a successful satellite mission is knowledge of its location and orientation in space, especially relative to a target. Relative navigation is an enabling technology for spacecraft formation flying, rendezvous and docking, and hazard avoidance. Cameras are particularly useful for this task since they are less expensive, smaller, and have lower power requirements than many other types of sensors. Object identification and relative pose estimation is therefore a key topic of research for the future of spacecraft.

Using cameras for object identification and relative pose estimation presents a few challenges. Obtaining relative position and orientation data is a two-step process. An object must first be identified so that the image data can provide a meaningful relative pose. Historically, the complete relative navigation process has involved two different algorithms, one for object identification and another for pose estimation, working in tandem. Finally, images in the visible spectrum are susceptible to variations in illumination that affects the perceived shape of the object, if it can be imaged at all.

The approach taken in this research is to apply terrestrial techniques to improve spacecraft navigation. First, appearance matching is used as a common framework for both object identification and pose estimation and is made more robust using background randomization. Consequently, a spacecraft imaging simulation environment is created to both generate the necessary training images as well as verify the systems performance. Additionally, results for multiple sensors are fused to improve the identification and pose estimation as well as increase the operating range over more of the orbit.

The result of this research is that a robust method is demonstrated for object identification and pose estimation of a spacecraft target. A single framework accomplishes both tasks and may be further enhanced using multiple sensors. Appearance matching and sensor fusion will help enable the next generation of spacecraft visual navigation.

CHAPTER 1
INTRODUCTION

1.1 Motivation

The last ten years have seen a fundamental change in the way spacecraft are designed and the purpose for which they are created. In the past, construction and launch of a satellite, even a small one, was out of the financial and logistical capability of all institutions except governments and large corporations. Collaboration was often required between multiple organizations. These devices were expensive, power-hungry, and needed to accomplish several different objectives in order to justify their high cost. Accordingly, they featured expensive sensors for accurate determination of their location and orientation. The result of the rapid miniaturization of electronics in the last decade is that the same capabilities are now available in smaller, less expensive packages that require less power. Low-cost microprocessors that are powerful enough to perform more complex pose estimation and sensor fusion computations in real-time are available to most consumers. As a result, two or more sensors may be combined to take advantage of complementary functionality without excessive cost. Sensor fusion is now a more viable strategy for spacecraft sensing than it was 10 or 15 years ago.

In addition, better on-board processing power means that terrestrial image processing and recognition techniques combined with spacecraft imaging sensors results in improved navigation performance. The appearance matching technique is a powerful algorithm that is well suited to space images. Developed by Murase and Nayar in 1995 [1], the recognition system learns to identify an object and determine its orientation from overall appearance rather than from a set of features. Appearance matching has been used in applications such as facial recognition and identification [2] and optical character recognition [3]. This

chapter describes the applications of satellite relative visual navigation and an outline of contributions.

1.2 Applications

A reliable and accurate method for performing vision-based spacecraft relative navigation has a number of potential applications. These include cooperative control, satellite servicing or de-orbit, and hazard avoidance. These scenarios respectively require either the maintenance of a constant position and orientation relative to a target, approach and rendezvous, or avoidance.

When discussing relative navigation, target objects for coordinated control are generally broken down into three overarching categories: actively cooperative, passively cooperative, and non-cooperative. Actively cooperative targets return some form of attitude and position data to the spacecraft. Passively cooperative targets do not broadcast any information, but have markers that are specifically designed to be picked up by the sensor. In the absence of either of these conditions, the target is non-cooperative. Non-cooperative targets include all natural bodies. The following applications include all three types of targets.

1.2.1 Cooperative control

Given the miniaturization of electronics and power storage improvements that have occurred over the last few decades, organizations have begun exploring the use of several smaller spacecraft to accomplish tasks which had previously been performed by a single satellite. These groups of satellites are referred to differently depending on their capabilities. They are usually referred to as "formations" if the control of one satellite involves the state of another or "swarms" if not. One example of a company utilizing a satellite constellation is Planet Labs. Formed in 2010, Planet Labs' goal is to launch a series of 3U CubeSats to obtain real-time Earth imaging data at the 3 to 5 meter resolution. As of October 2017, there are 190 satellites in orbit, 172 of which are actively imaging [4].

Another area with a great deal of potential application for satellite constellations is meteorology. In December 2016, NASA launched the Cyclone Global Navigation Satellite System (CYGNSS), a joint operation between the University of Michigan and the Southwest Research Institute. Consisting of 8 micro-satellites, CYGNSS aims to improve hurricane forecasting by taking ocean surface wind speed measurements at a much higher rate than previous missions. The CYGNSS constellation has a predicted mean revisit time of 7 hours for any location on Earth. Each satellite measures the wind speed at 4 locations simultaneously, allowing for up to 32 measurements per second [5].

The next step in complexity beyond satellite constellations is coordinated attitude control of multiple satellites. Several applications are feasible for this type of system, including mesh networks to replaced damaged communications satellites or instantaneous synthetic aperture radar systems. A review by Bandyopadhyay, et al. shows the breadth of topics covered by contemporary formation flying missions [6].

1.2.2 Satellite servicing or de-orbit

Another potential application for a satellite with a reliable visual navigation system is rendezvous. One potential rendezvous mission is the servicing of a damaged or spent satellite. The servicing satellite could provide a software patch too large to be radioed from the ground, refuel a spent cold gas thruster system, or boost the satellite into another orbit. The Robotic Servicing of Geosynchronous Satellites (RSGS) program is a mission concept being developed by the Defense Advanced Research Projects Agency (DARPA). RSGS is intended to extend the life of satellites in geosynchronous orbits and identifies four potential mission tasks: high-resolution inspection, anomaly correction, cooperative relocation and upgrade installation [7].

On the other hand, the helper satellite could approach with the intent of retiring the spent satellite. Most modern satellite missions are required to explain how the spacecraft will be de-orbited at the end of the mission, assuming natural forces do not solve that problem

Figure 1.1: Artists rendering of RSGS satellite concept [8]

within a reasonable span of time. Should the intended de-orbit mechanism fail, another spacecraft could be launched with the purpose of pushing the target into an unstable orbit so that it does not contribute to the growing amount of space debris. Singaporean company Astroscale is developing a proof-of-concept mission called ELSA-d with a planned launch in 2019. ELSA-d will approach a simulated target using optical sensors and then push the target into an new orbit for disposal [9]. By changing the perigee to intersect with a denser part of the atmosphere, the orbit decays to the point where the spacecraft burns up due to atmospheric friction.

1.2.3 Hazard avoidance

A visual navigation system would also be useful in assisting hazard avoidance maneuvers. Instead being part of a small, relatively inexpensive satellite, this system would be incorporated as part of an expensive, long-term mission. Given the rise in space debris [10], particularly in low orbit, having a way to autonomously determine if a nearby object is a threat to the spacecraft would be very useful. A system based on this research could identify the debris from a library of objects and provide relative navigation information in order to assist with the threat assessment and, if necessary, help effect an avoidance maneuver.

1.3 Statement of contributions

The overall contribution of this dissertation is the improvement of object identification and pose estimation with methods that have not previously been used in the aerospace field. The computer vision technique called appearance matching performs object identification and attitude determination using the same mathematical framework and is made robust to different environments using background randomization. Pixel data from the image combined with the object identity allows for a complete pose estimate. The spacecraft imaging simulation environment (SISE) generates training images to train the appearance matching method and is used for software-in-the-loop validation. Finally, multi-spectral sensor fusion extends the operating range of a single-spectrum sensor while allowing for the capability of more advanced filtering techniques.

1.3.1 Appearance matching for spacecraft

Contribution 1: Appearance matching is applied to spacecraft object identification and pose estimation for application to uncooperative objects and is extended for use on an arbitrary background with a single training set.

Several algorithms exist for determining the presence and identity of a target, called object identification, and calculating its relative location and orientation in space, called pose estimation. Methods for object identification include 3-D model searches [11], scale-invariant feature transformation (SIFT) [12], and shape context matching [13]. Methods like simultaneous localization and mapping (SLAM) [14], the blobber method [15], and perspective-n-point (PnP) [16] perform pose estimation.

Each of these techniques has advantages and disadvantages. For example, SIFT and PnP both require the algorithm to identify feature points on the object, which may not be possible for smooth, shiny objects or while using lower resolution images. Model searches and context matching are only effective for simple objects that can be modeled from a set of primitives, which could be difficult to construct for larger or more complex spacecraft.

Whichever of these individual techniques is selected, one algorithm will have to be used for object identification and another for pose estimation. These algorithms would then be coded and optimized for performance separately. The advantage to appearance matching is that it performs both processes by the same principle without the need for identifying feature points. Training images are mapped into a series of higher-dimensional spaces to tune the algorithm, and test images are mapped into those same spaces for object identification and pose estimation.

By extending the technique using background randomization, appearance matching uses the same set of training images to identify objects and conducts relative pose estimation on an arbitrary background. The prior version of the algorithm required that the background be the same for the training images and the test image, usually black. This research represents the first time appearance matching has been applied to spacecraft relative navigation and extended in this way using background randomization. Preliminary results for this contribution were presented in McBryde and Lightsey [17] and updated results are under review for publication (McBryde, et al. [18]).

1.3.2 Spacecraft imaging simulation environment

Contribution 2: A versatile software tool was created for generating simulated visible and infrared spectrum images to use in algorithm training and software-in-the-loop verification, with enhanced computational efficiency using graphics processing hardware.

One of the first steps in approaching the development of a visual navigation system is to simulate or acquire high-fidelity images of the target. Generating realistic real-world scenes has been an area of intense research over the past few decades with the increase in the photorealism of video games and computer generated images for film and TV. Many tools exist for simulating objects based on a 3-D model and with particular characteristics [19, 20, 21]. However, these tools are not suited for this application of appearance matching and sensor fusion for a couple of reasons. First and foremost, they do not include the ability

to simulate infrared radiation. Research has been published on the operational physics of infrared sensors [22, 23]. However, the actual process of simulating infrared signals has been limited. Shi, et al. [24] simulated infrared radiation using rudimentary CAD models, but these were very coarse and lacked any features.

Another drawback is that existing tools are mostly "black boxes." They render a scene without providing the user with data such as the radiation emitted from each face of the object or how many photons are incident on the sensor. This data is useful in algorithm tuning and in accounting for error sources. Finally, it will be shown later that automating the generation of simulated images is a necessary part of the training process for the appearance matching algorithm. The ability to access the source code of the software directly means that the tool can be easily modified, e.g., to produce a series of images in time for a software-in-the-loop test or a series of images at different orientations and lighting conditions for a training data set.

The spacecraft imaging simulation environment (SISE) is explicitly designed for the training and testing of a visual navigation system incorporating appearance matching and visible and infrared spectrum sensor fusion. A previous version of SISE is found in McBryde and Lightsey [25].

1.3.3 Multi-spectral sensor fusion

Contribution 3: A relative navigation filter is developed and demonstrated that applies sensor fusion to pose estimation results from appearance matching to allow for a larger operating range over sunlit and eclipse portions of the orbit, under more challenging lighting conditions, and with more accurate object identification and pose estimation.

Multi-sensor fusion, including both visible and infrared spectrum images, is developed, which provides an additional improvement to spacecraft relative sensing. Since radiation generated in the infrared wavelength is based on the temperature of the object, these cameras are useful in the case when the target object is not fully illuminated. For example,

the Earth eclipses the sun for most satellites in low Earth orbits. Thermal cameras allow the system to distinguish the satellite from the background that has a different temperature such as the surface of the Earth. This capability was demonstrated by the Prox-1 mission at Georgia Tech [15]. Thermal cameras should not be relied on exclusively, however. They have lower resolutions than visible-spectrum cameras for similar volume, mass, and power requirements. These cameras also have difficulty distinguishing features on the target object that are the same temperature. Sensor fusion combines the best capabilities of images taken in both spectra. The multi-spectral framework also forms the basis for the implementation of *a priori* or filtered object identification and pose data through the use of virtual sensors. These sensors are a specialized data structure within the sensor fusion framework. They perform like a real sensor but their "data" is based on external information and an attitude model.

This research represents the first application of a sensor fusion framework to appearance matching for any application, including relative navigation.

CHAPTER 2
BACKGROUND

Computer vision has been a topic of consistent research and hardware development over the past 30 years. Machine vision, image identification on the internet, and object character recognition all have their own unique challenges and methods of solution. Relative pose estimation is another category of work in computer vision, a process that sometimes includes object identification as well. Security applications techniques have led to the combination of images in the visible and infrared spectra.

Several past space missions have made use of optical sensors to accomplish relative navigation. These include rendezvous and docking tasks as well as interplanetary navigation. Devices have also been developed to take advantage of multiple spectra, including near-infrared sensitivity and single-device sensor fusion.

This chapter describes the current state of visual navigation software and hardware. It identifies terrestrial options which have potential on-orbit applications and those currently available for spacecraft navigation as well as their potential drawbacks.

2.1 Current algorithms

2.1.1 Object identification

Methods for object identification include scale-invariant feature transformation (SIFT), 3D model searches, and shape context matching (SCM).

2.1.1.1 Scale-invariant feature transformation

SIFT is a method of identifying key points that are maintained even when the image is transformed. Developed in a seminal work by Lowe [12], these points are invariant to scale,

translation, and rotation. The SIFT features are also somewhat invariant to illumination changes and affine projection, which stretches one part of the image relative to the rest. According to Lindeberg [26], Gaussian functions and their derivatives must be used for a scale-space analysis, in which the object is able to be identified at different sizes in the image. To additionally preserve rotation and translation invariance, the extrema of the difference of two different Gaussian functions can be used (Lindeberg [27]). Candidates for these key points are therefore found by using the extrema of a two-dimensional Gaussian function. This type of function is separable, so in practice the function is applied in a horizontal and then a vertical path to increase computational efficiency. A sample 1-D function for such a formulation is given by Equation 2.1.

$$g(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2} \tag{2.1}$$

The candidates are validated by checking if they continue to be key points at two different levels of resampling, one higher and one lower. By corresponding key features in a library image to those in a test image, an object can be identified. An example of such a test is given in Figure 2.1.

SIFT is very useful for situations like the one given above, where multiple objects, possibly occluded, need to be identified in the same image. However, a typical spacecraft navigation scenario deals with only one single target at a time, though the algorithm might attempt to identify parts of the spacecraft as separate targets. Also, SIFT only has partial invariance to illumination changes. Illumination on objects in space varies significantly, especially for objects passing through eclipse by a celestial body. Therefore SIFT was determined to have limited application for object identification in space.

2.1.1.2 Shape context matching

Shape context matching (SCM) works on a similar principle to SIFT. However, instead of the individual key points being important, SCM focuses on the relative positioning of all

Figure 2.1: Model images of planar objects (top), test image (center), and recognition results showing model outlines and image keys used for matching (bottom) [12]

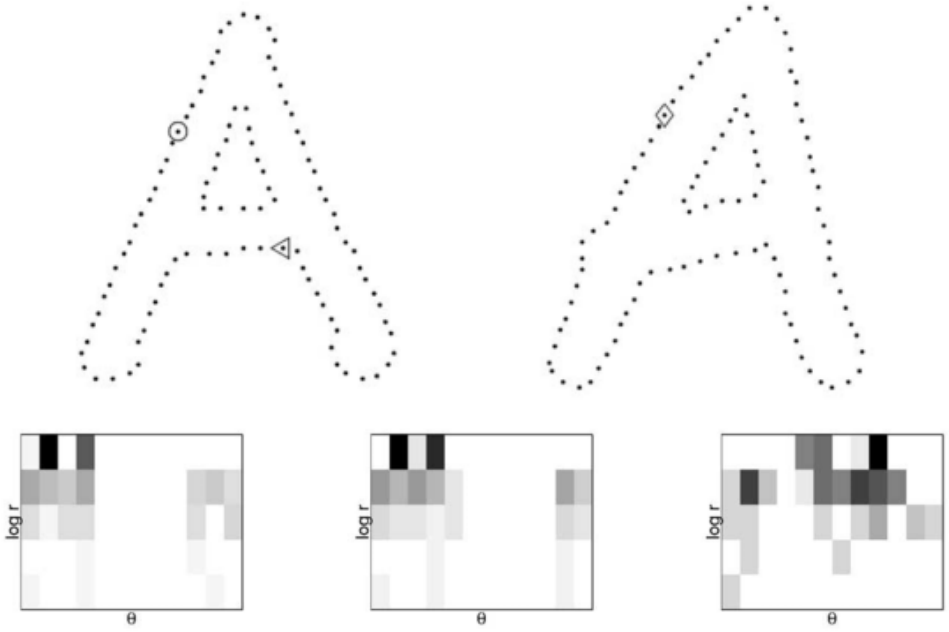

Figure 2.2: Example of sampling for two images (top). Shape contexts for circle (bottom left), diamond (bottom center), and triangle (bottom right) [13]

of the feature points (i.e., the *context*). An edge detector is run on the image and feature points are sampled from the edges. Figure 2.2 shows an edge detection and sampling of two letter A's from Belongie, et al. [13].

For each point the context is found. This context is represented by a 2D histogram. The bins for the histogram are defined to be uniform in log-polar space, which makes the descriptor more sensitive to points nearby the focus point than to those further away. For object detection, the shape contexts of a test image are compared to those in an image library using the principles of bipartite graph matching. In mathematics, a bipartite graph is defined as having vertices in two independent sets such that edges connect a vertex in one set to one in the other. Graph matching is a way to create such a set, and SCM uses the improved linear assignment method developed in Jonker and Volgenant [28].

SCM is intrinsically invariant to translation since the algorithm is based on relative position. It is made robust to scale by normalizing the distance to the feature points and to rotation by implementing a relative instead of an absolute reference frame. The relative

Figure 2.3: Sample images from the COIL-20 library [30]

frame chosen is the thin plate spline model developed by Duchon [29].

SCM presents one main drawback for spacecraft appearance matching. In order for the matching to be performed consistently, the shape of the object must be relatively consistent between the library and the test images. If an object is partially illuminated, as is common for objects in space, then the shape will vary even though the object is the same. The 3D object tests presented in [13] used the COIL-20 library [30], a sample of which is show in Figure 2.3. That image set consists of household objects imaged under full illumination on a black background. Since neither of these conditions can be guaranteed in space, the given case study does not demonstrate that this technique would be successful on orbit.

2.1.1.3 3D model search

The 3D model search method arose out of a desire to improve upon search tools for such models online. Funkhouser, et al. [11] describe various types of search queries: model queries in 3D, sketch queries represented in 2D and 3D, and text queries. The method that would be used for visual navigation is a search based on a three-dimensional sketch using the test image from the camera as the "sketch." The 3D model search works by first rasterizing the model into a voxel grid. Each coordinate is assigned a value of 1 if the surface of the model is present within the voxel and 0 if it is not. This grid is treated as a binary real-valued function and is then decomposed using spherical harmonics by

restricting the sphere to various radii. The result is Equation 2.2.

$$f_r^m(\theta,\phi) = \sum_{n=-m}^{m} a_{mn} \sqrt{\frac{(2m+1)}{4\pi}\frac{(m-|n|)!}{(m+|n|)!}} P_{mn}(\cos\theta) e^{in\phi} \tag{2.2}$$

The algorithm thus obtains a two-dimensional, rotation-invariant descriptor for the object using spherical harmonics. Specifically, the value at index (m_0, r_0) corresponds to the amplitude of the m_0-th frequency when the function f is restricted to radius r_0. An analogous descriptor is obtained based on the 2D contours of the test image and two are compared in higher-dimensional space in order to identify the image. This second method is described in greater detail in Funkhouser, et al. [11] as well as in work by Zahn and Roskies [31].

The 3D model search method was designed to quickly identify a model out of a large repository, potentially 20,000 entries or more. This type of search is not as useful for identification of spacecraft, where the potential library would be orders of magnitude smaller. In addition, the way in which the search identifies the object precludes the ability to determine the attitude of the object, making a second method necessary for that step.

2.1.2 Relative pose estimation

The central problem in visual navigation is to determine the 3D relative position and attitude of a target in the image frame with respect to the observer. The relative position and attitude together are known as the pose, and determining the pose from a 2D image is referred to as pose estimation. Pose estimation methods include simultaneous localization and mapping (SLAM), the blobber method, and perspective-*n*-point (PnP).

2.1.2.1 Simultaneous localization and mapping

SLAM is perhaps the most well-known pose estimation algorithm. It was first introduced in 1991 in the fundamental work by Leonard and Durrant-Whyte [14]. As the name suggests, SLAM calculates the relative position of the camera to the environment at the same

time as it is estimating the environment itself. It developed out of apparently conflicting requirements in the robotics field. In order to estimate its environment precisely, a robot must know its location, and in order to know its location, it must have a a precise model of the environment. SLAM provided a methodology for the environment and the robot's location to be estimated and refined simultaneously.

SLAM may be performed with different types of sensors. Sonar was chosen in the original research, but it is clearly not an option in the space environment. Instead, LIDAR is often used for space-based applications of SLAM [32]. When SLAM is implemented with cameras only, it is referred to as visual SLAM (vSLAM) and is another option for implementing SLAM on orbit. Taketomi et al. [33] provide a thorough survey of recent vSLAM algorithms. These methods are divided into two categories: feature-based and direct.

Feature-based methods work by initializing the map with feature points on a known object and then performing SLAM from there. Direct methods, also known as feature-less methods, eschew any kind of abstraction and use synthetic view images to estimate the camera motion and continue to improve the map and camera location information. Typically, direct methods employ stereo images to determine scale, which otherwise has to be supplied *a priori* or by a different sensor.

SLAM and vSLAM are incredibly useful tools in robotics. However, implementing a solely vSLAM strategy on orbit would be problematic. Identifying points for feature-based vSLAM runs into problems with invariance to illumination that are similar to those that are discussed for SIFT in Section 2.1.1.1. Direct vSLAM methods also require the simulation of images on-board the spacecraft, which would be a large strain in processing load and data storage on embedded systems.

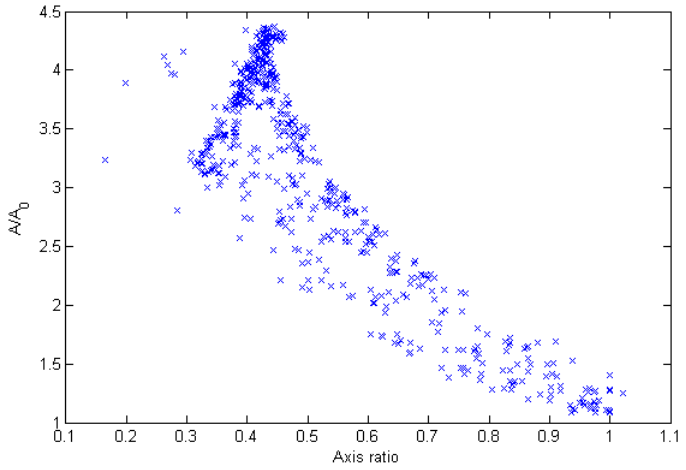

Figure 2.4: Example of projected area blobber distribution for a 3U CubeSat

2.1.2.2 Blobber method

In order to provide relative position information in the vicinity of another, non-cooperative body, another option is the blobber algorithm (Walker [15]). This algorithm acquires relative position information in a two-stage process. First, a unit vector to the target is determined by locating the center of brightness on the image plane. Then, using 2D image coordinates and the camera geometry, a 3D unit vector approximately directed toward the center of the body is determined in the camera's reference frame.

The algorithm then estimates the range to the object. First, the major and minor axes of the blob are determined as well as the ratio of the lengths of these axes, called the axis ratio. Next, estimates for the maximum and minimum projected areas are found using numerical methods. The projected area, known as A/A_0, is the ratio of the area viewed on the two-dimensional image plane to the minimum possible area. For example, A_0 for a 3U CubeSat would be the area of its smallest face, 100 cm^2. A distribution of possible projected areas versus the axis ratio is found using randomly generated orientations of the target object. An example distribution for the 3U CubeSat example is given in Figure 2.4. These data were generated by simulating images of the satellite at random attitudes.

A maximum and minimum area curve is determined from this distribution of points,

from which an estimated range can be found via Equation 2.3 [15].

$$\rho_{mean} = f\sqrt{\frac{A_{mean}}{N_{blob} \cdot p^2}} \tag{2.3}$$

In this equation, f is focal length of the camera, A_{mean} is the average projected area for a given ratio of axes, N_{blob} is number of pixels the imaged object illuminates, and p^2 is the physical area of one pixel on the sensor.

The blobber algorithm approaches the relative navigation problem by analyzing the geometric properties of the image foreground, which is subject to error when the body is only partially illuminated. This error results from the fact that while the blobber algorithm takes into account varying orientations of the object, all of the orientations are viewed with the same illumination conditions. This method also does not provide relative attitude information, and thus is not a complete pose estimate.

2.1.2.3 Perspective-n-point

The PnP technique determines the pose of a camera relative to a target by using the intrinsic properties of the camera and a set of n 3D-to-2D correspondences. In order to perform PnP, feature points on the object must be selected in the image. These features are also located on the object in 3D to generated the required correspondences.

The PnP method used as an example here is known as efficient perspective-n-point (ePnP) and was developed by Lepetit et al. [16]. The ePnP algorithm expresses the reference points in 3D space ($\mathbf{p}_i,\ i = 1, ..., n$) as weighted sums of 4 control points ($\mathbf{c}_j,\ i = 1, ..., 4$) as shown in Equation 2.4. The weights are given by the α terms.

$$\mathbf{p}_i = \sum_{j=1}^{4} \alpha_{ij}\mathbf{c}_j \tag{2.4}$$

These control points may be selected arbitrarily; however, the authors suggest using the centroid of the reference points as one control point and selecting the rest to form a basis

aligned with the data's principal directions. The correspondences lead to a linear system of the form given by Equation 2.5.

$$\mathbf{M}\mathbf{x} = \mathbf{0} \tag{2.5}$$

In this case, $\mathbf{x}$ is a vector of length 12 consisting of the control point coordinates and $\mathbf{M}$ is generated using the 3D-to-2D correspondences. The details of this process are found in [16], and the result is the relative pose of the target object in space.

PnP needs this set of correspondences in order to function. Thus, the identity of the object must be known beforehand or be determined with some other method. There also must be a way to locate the key points in the image in order to correspond them to the 3D points on the object, which is a difficult problem if the target does not have fiducial markers.

2.1.3 Sensor fusion and filtering

Visible and infrared spectrum sensor fusion is commonly used in terrestrial security applications. One example is from Han and Bhanu [34]. The goal behind their work is to extract a moving human silhouette from synchronized visible and thermal images. That silhouette is then used as an initial correspondence point for a hierarchical genetic algorithm to register the color and thermal images. This registration improves the silhouette detection. A sample color-thermal registration is given in Figure 2.5. Once the images from the two spectra have been registered in this way, they can be combined to locate the silhouette of the human figure in both images (Figure 2.6).

Another similar application is from Saeedi and Faez [35]. Instead of extraction, a synthesized image is created using the background detail of the visible spectrum image and the thermal detection of the infrared camera. This fusion is based on wavelets and uses fuzzy logic and population-based optimization. The result is an enhanced security camera feed that shows the presence and location of a human in the image, like the one shown in Figure 2.7.

Figure 2.5: Color and thermal registration results from Han and Bhanu [34]: original color images (first row), original thermal images (second row), and transformed color images (third row)

Figure 2.6: Transformed color image (first), thermal image (second), and silhouette fusion using two different methods (third and fourth) [34]

Figure 2.7: Infrared image (left), visible image (center), and fused image (right) [35]

Figure 2.8: Infrared image (left), visible image (center), and human detection (right) [36]

Lastly, Zhao and Cheung [36] have put forward a method for segmentation of a human from an image using visible and infrared fusion. This concept is similar to Han and Bhanu [34], but instead of just the silhouette, the detail of the human target in both spectra is extracted. The method employs an initial blob-wise registration which is then improved using temporal inferencing and a two-tier tracking algorithm on both the individual and combined signals. This method is an improvement over [34] because it is accurate at different depths, as shown in Figure 2.8.

The concepts from these methods translate well to the requirements of a visual navigation system for spacecraft. In particular, utilizing the infrared camera to locate and scale and object within the image frame is a concept that is further explored in Chapter 5.

2.2 Current hardware

Recent visual navigation hardware is divided in two categories: single-spectrum and multi-spectra. Single-spectrum sensors rely on a single wavelength to obtain visual data. The

sensors described in the following section operate in the visible spectrum. Multi-spectra sensors use data from more than one spectrum, namely the visible and infrared spectra.

2.2.1 Single-spectrum sensors

The following three missions employed visual navigation sensors in a single spectrum. They relied either on fiducial markers or a target with a known geometry.

2.2.1.1 Orbital Express

Orbital Express (OE) was a mission sponsored by DARPA that performed the first on-orbit, fully autonomous rendezvous and capture [37], using no instructions after maneuver initiation or input from the ground. The Advanced Video Guidance System (AVGS) used on OE consisted of two laser diodes, operating at 800 and 850 nanometers, a retroreflective mirror on the target, and a camera to image the laser return. Due to the specialized mirrors on the target object, AVGS would be considered a passively cooperative target.

AVGS worked by subtracting one image of the object illuminated at one wavelength from another illuminated at the other wavelength. The result eliminated the entire object from the image except for the retroreflectors. AVGS operated well in the challenging lighting conditions of space. Once the fiduciary markers were identified, their centroids were found and tracked in order to provide a relative navigation solution. A block diagram of this procedure is given in Figure 2.9.

OE was a highly successful mission but the AVGS navigation concept has a few limitations as a visual navigation sensor. The first and most obvious is that a particular retroreflector setup is needed on the target object to perform navigation. This requirement precludes using AVGS for non-cooperative targets. In addition, operating two lasers in addition to a camera is a high power requirement for smaller or lower-cost spacecraft.

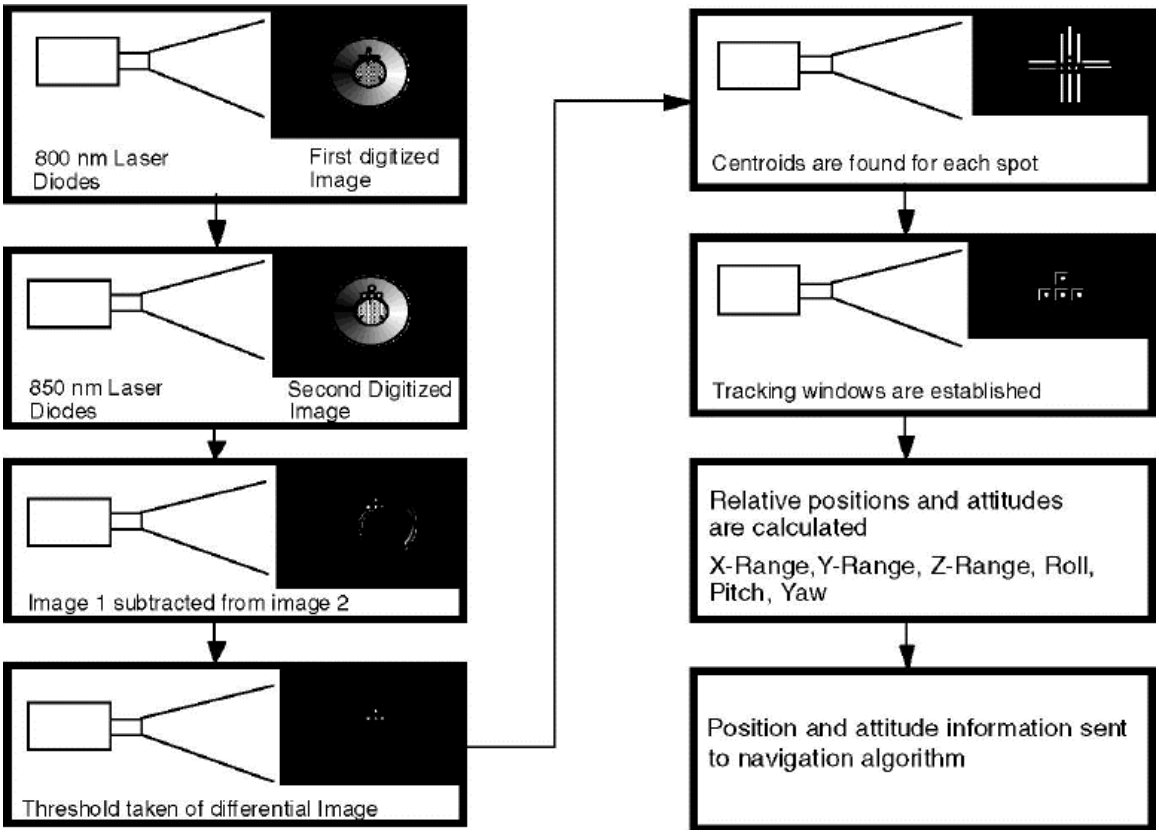

Figure 2.9: AVGS illumination and processing procedure [37]

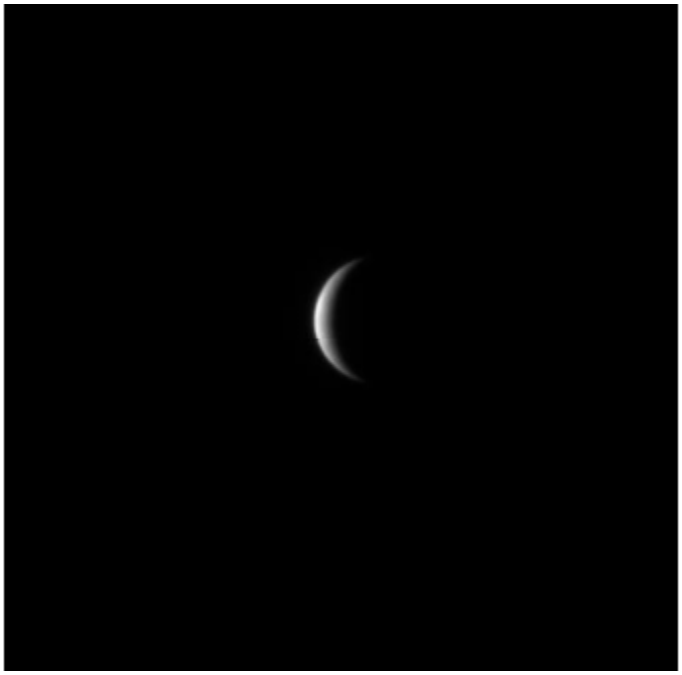

Figure 2.10: Mercury flyby narrow-angle camera image from OpNav number 3 [38]

2.2.1.2 MESSENGER

The Mercury Surface, Space Environment, Geochemistry, and Ranging (MESSENGER) mission was launched in 2004 with arrival at Mercury in March 2011. An optical navigation system was employed on the spacecraft in part to support the gravity assist flybys prior to Mercury orbit insertion [38].

The optical navigation on board MESSENGER used the illuminated limb of the planet (Figure 2.10) in order to locate Mercury within the camera's field-of-view. Given knowledge about the camera geometry and the size of Mercury, the algorithm determined the position of the spacecraft relative to the planet. This information was intended to help make a decision on a potential contingency maneuver during the first Mercury approach. While the optical navigation solution was not ultimately used for this purpose, it did later support final confirmation of the injection maneuver and trajectory reconstruction.

The type of visual navigation implemented on MESSENGER is not extendable to other mission types, despite being useful for navigating with respect to a planet. Locating the

centroid of an object using its illuminated limb is a technique restricted to spheroids.

2.2.1.3 Hubble Space Telescope Servicing Mission 4

The Relative Navigation Sensor (RNS) system was developed by NASA's Goddard Space Flight Center and flown in the cargo bay of the Space Shuttle Atlantis during the Hubble Space Telescope's Servicing Mission 4 (SM4) in May 2009. The system processed images in real time during the rendezvous, docking, and deployment phases of the servicing mission. RNS had three cameras whose images were processed using two different algorithms: GNFIR and ULTOR P3E [39].

The Goddard Natural Feature Image Recognition (GNFIR) system is based on an early real-time 3D tracking method known as Real-time Attitude and Position Determination (RAPiD) [40]. GNFIR works by utilizing features on the target, like edges, and therefore is useful on a non-cooperative target. It does require a predetermined stick-model of the edges in question. The input image is pre-processed through an edge-detector algorithm before processing by the feature recognition system. GNFIR can initialize itself or utilize a commanded initial condition or another attitude determination solution. A wireframe is projected using this initial pose and then the error is minimized in a least-squares fashion. A block diagram for this process is given in Figure 2.11.

By utilizing edge-detection, GNFIR gets around some of the issues that harsh illumination presents in space images. However, it would still be ineffective if the object were partially illuminated, since the wireframe model would not match up with the edge-detected image. A reasonable number of edge features are also necessary for this method to be successful. Depending on the image and the target object, enough edges may not be present or may be hard to detect.

The ULTOR Passive Pose and Position Engine (P3E) uses training data to identify features on the target object, which in this case is the Hubble Space Telescope. Predetermined filters of the target object are able to isolate features within an image. These same filters

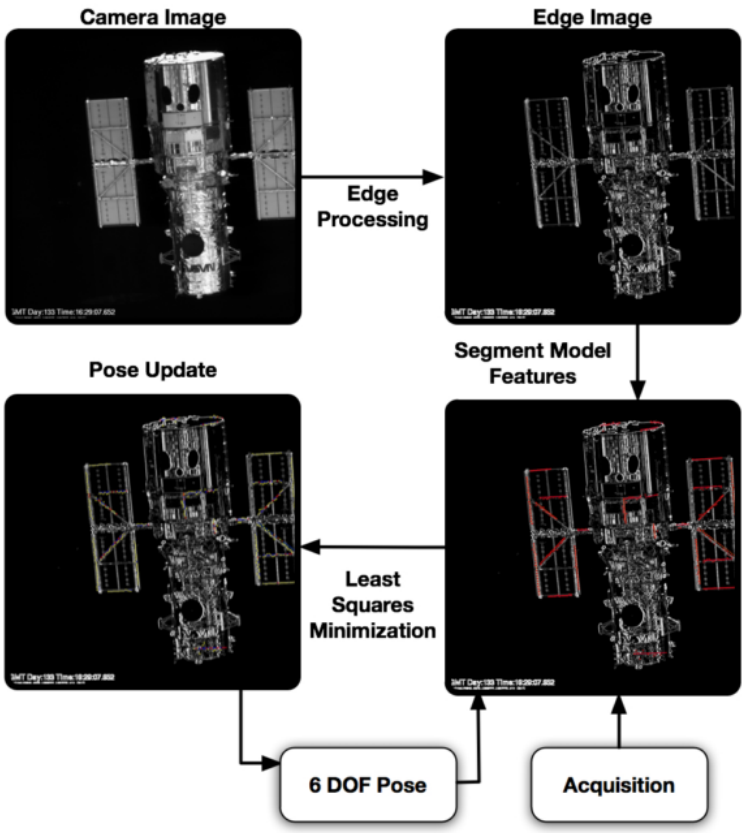

Figure 2.11: GNFIR Pose Process Loop [39]

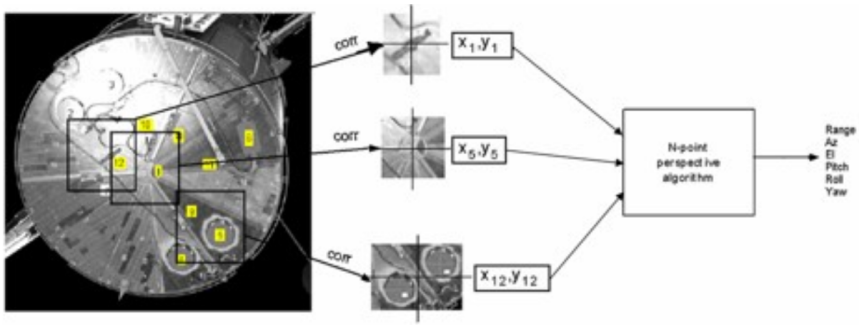

Figure 2.12: Feature-based ULTOR P3E overview [39]

are then applied to a test object, and based on the location of the features within the image and a mechanical model of the target object, the relative position and attitude of the target can be determined.

Specific details on the operation of the ULTOR P3E are not available because the technology is proprietary. However, it seems to operate in a similar manner to methods like SIFT. Features are identified and then are used in combination with a 3D model to determine the pose using PnP or a related algorithm. Therefore, it possesses the same major drawback as SIFT: solution sensitivity to harsh lighting conditions.

2.2.2 Multi-spectrum sensors

There are two main strategies for creating combined visible and infrared hardware. One option is to use a single device which is primarily a visible spectrum camera, but which also is sensitive in the near-infrared spectrum. Another is a hybrid sensor: a visible and an infrared detector in the same device. While multi-spectra cameras exist for terrestrial applications, there are fewer examples of hardware which have space flight heritage.

2.2.2.1 Near-infrared sensitivity

One simple way to combine visible and infrared data is to use a camera which is sensitive in both spectra. Several terrestrial security cameras operate in this way, including those by Gamut [41] (Figure 2.13) and by Baluff [42]. Most CCD sensors have sensitivity that

Table 2.1: Infrared spectrum subdivision

Division	Abbreviation	Wavelength	Temperature
Near-infrared	NIR	0.75 to 1.4 μm	3591 to 1797 °C
Short-wavelength infrared	SWIR	1.4 to 3 μm	1797 to 693 °C
Mid-wavelength infrared	MWIR	3 to 8 μm	693 to 89 °C
Long-wavelength infrared	LWIR	8 to 15 μm	89 to -80 °C
Far infrared	FIR	15 to 1000 μm	-80.15 to -270.15 °C

extends into the near-infrared (NIR) regime. Security cameras take advantage of this by illuminating their field-of-view with infrared LEDs. This light is invisible to the human eye, but allows the cameras to see objects in the dark. This type of system has not yet been implemented on spacecraft.

Note that this type of sensor is not an infrared sensor in truest sense. It does not record the infrared radiation emanating from a body; rather, it illuminates the target with infrared light in the NIR division. Table 2.1 gives a common subdivision of the infrared spectrum. Each division has a name, abbreviation, wavelength and temperature for which the infrared radiation peaks according to Wien's displacement law. That law states that the black body radiation curve peaks at a wavelength (λ_{max}) inversely proportional to the absolute temperature (T) with proportionality constant (b) called Wien's displacement constant, which is approximately equal to 2900 μm ·K (Equation 2.6). Thus, in order for a infrared sensor to record radiation from a spacecraft in a typical operating temperature range, it must be sensitive in the LWIR division.

$$\lambda_{max} = \frac{b}{T} \tag{2.6}$$

2.2.2.2 Single-device sensor fusion

Hybrid sensors that have detectors for multiple spectra in the same device are more rare, even for terrestrial applications. These are devices which employ full visible and infrared spectrum cameras combined into one housing. One example of this is the ARTCAM 320-

Figure 2.13: Gamut 1080p HD TVI CCTV Bullet Camera [41]

Figure 2.14: ARTCAM 320-THERMO-HYBRID [43]

THERMO-HYBRID [43] that is shown in Figure 2.14. This camera uses a 1280 by 1024 pixel visible spectrum camera with a 3.6 millimeter focal length as well as a 320 by 240 pixel uncooled bolometer with and 8 millimeter focal length and spectral sensitivity from 8 to 14 micrometers in the LWIR division.

2.3 Summary

This survey represents the current state of the art in spacecraft image-based relative navigation. Several image-based relative navigation algorithms exist with potential application to spacecraft hardware. Similarly, different hardware solutions have been implemented in a single spectrum or are available in multiple spectra. However, none of these options

available to date have provided spacecraft visual navigation in a unified, end-to-end frame work incorporating sensor fusion. This observation is the motivation for the research that is conducted in this dissertation. The following chapters build upon existing techniques and take advantage of advances in miniaturization to outline such a solution.

CHAPTER 3
SPACECRAFT APPEARANCE MATCHING

After consideration of many different algorithms for visual navigation as described in Chapter 2, appearance matching was selected as the preferred solution approach for the space application. The appearance matching algorithm is presented in this chapter along with a discussion of its advantages and limitations. A new technique is created for training the appearance matching algorithm with randomized image backgrounds. This randomized background technique is demonstrated to have improved robustness with respect to varying background lighting conditions such as the presence of the Earth's surface in the image background.

3.1 Overview

3.1.1 Original work

The fundamental research in the topic of appearance matching was presented by Murase and Nayar in 1995 [1]. Hiroshi Murase had previously done work in the area of pattern matching, publishing two papers on recognition of characters in the Japanese writing system called *hiragana* [3, 44]. Shree K. Nayar had done significant work in modeling the illumination of objects with regard to computer vision [45], including work which tied directly into the training of an appearance matching algorithm [46].

The idea behind Murase and Nayar's appearance matching algorithm is fairly straightforward. Instead of focusing on particular feature points in an image to do object identification and attitude determination, the entire appearance of the object is analyzed. An object's appearance in an image depends on several different factors. The shape and reflectance of the object are intrinsic properties and do not vary from scene to scene. The appearance

also depends on the sensor taking the image: detector type, camera and lens geometry, and exposure time. It is assumed that these properties, too, are held constant. The last set of factors come from the scene itself: the pose of the object and its illumination.

In order to account for variations in these final parameters, Murase and Nayar proposed to create a library of images of each object under various lighting conditions and orientations. A test image is then compared to this library to find its closest match, which determines the identity, orientation, and lighting of the test image. However, comparing the entire image pixel by pixel to the image library is extremely computationally intensive.

Therefore, Murase and Nayar compressed the images into an eigenspace with smaller dimensions than the image library. This compression was accomplished using principal component analysis (PCA). The result was a series of points in an eigenspace representing the object in various poses and under different illuminations. Since these parameters varied only slightly from image to image, the location of the resulting points varied only slightly from image to image and the result was a series of higher-dimensional surfaces, one for each object. Figure 3.1 shows the objects used for this original research, and Figure 3.2 shows the resulting surfaces. Note that only the first three dimensions are shown for each object. Due to the way in which the PCA is constructed, these are the three directions of greatest variance.

The results from Murase and Nayar's initial work were promising. Using a training library of 4 objects, 5 illumination conditions, and 90 poses, the result was a successful object identification rate of 100% and an average absolute pose error of 0.5 degrees. It should be noted, however, that these results were found by varying both the pose of the object and the illumination about a single axis each. In order for appearance matching to be a viable method of relative attitude determination, these results must be extended to three attitude axes and two illumination direction axes per light source. The assumption is that the lighting is not circularly polarized and therefore is the same for any rotation about the direction to the light source. Thus, a third lighting direction is not needed.

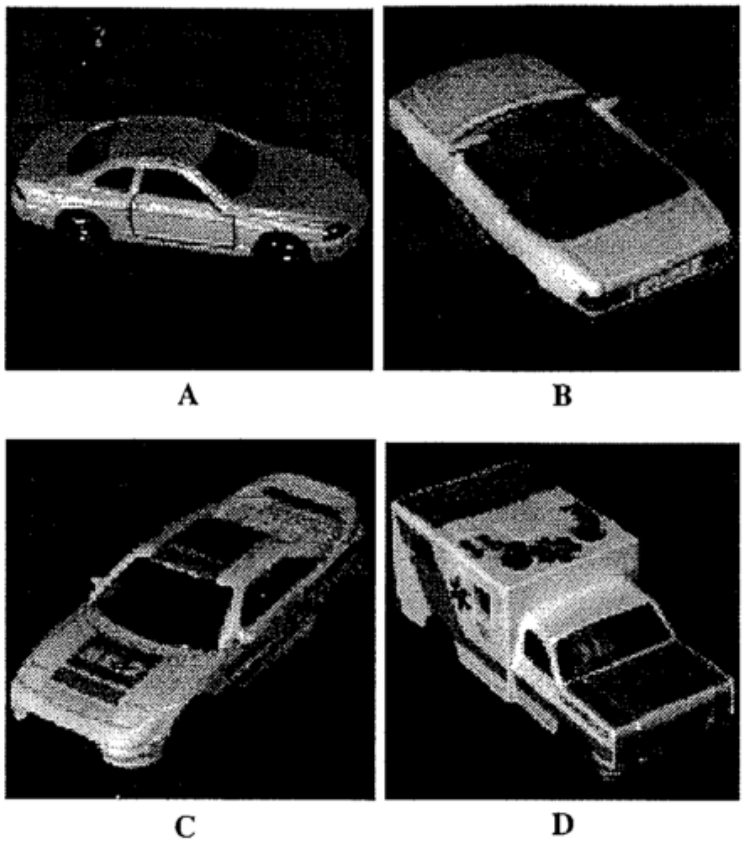

Figure 3.1: Experimental objects used by Murase and Nayar [1]

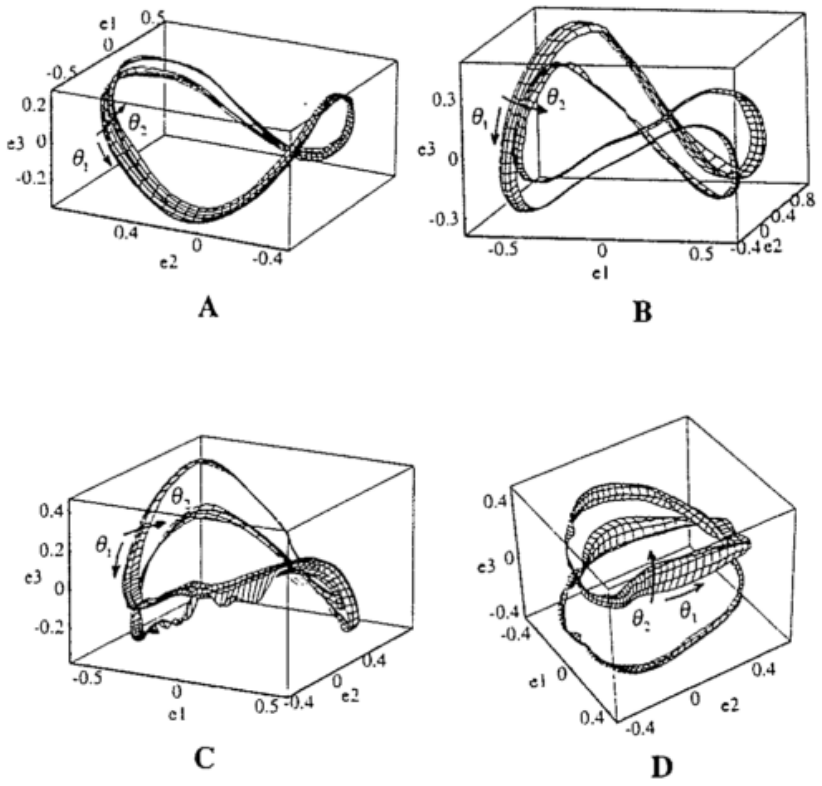

Figure 3.2: Hypersurfaces computed for each object (first three dimensions) [1]

3.1.2 Appearance matching for spacecraft visual navigation

Appearance matching has a few advantages over the methods described in Chapter 2. First, the measurement step for spacecraft navigation requires a two-part process. Existing methods that have been employed perform either object identification (Section 2.1.1) or pose estimation (Section 2.1.2). Not only is appearance matching able to execute both parts of the measurement step, but it does so within the same mathematical framework. The procedure needs to be optimized only once for the spacecraft on-board computer and stored only once on the on-board memory. Note that the relative position is actually found by combining the object identity and pixel data, but this is not a computationally intensive process.

Appearance matching is also robust to variable on-orbit lighting conditions. The training procedure accounts not only for different spacecraft orientations but also different illuminations, allowing accurate relative pose measurement even if the target is only partially illuminated. Additionally, because the entire appearance of the target is used for object identification and pose estimation, it performs well for smooth or shiny surfaces from which feature points may not be easily extracted.

There are a few limitations to appearance matching. First, while much of the computational complexity of the algorithm is performed on the ground before launch, the nearest-neighbor search has to be executed on-board the spacecraft and requires an optimized algorithm to run efficiently (Section 3.3.2.3). Also, since the relative pose measurement relies entirely on the object's appearance, the algorithm may experience problems when the object is symmetric. This difficulty results from the fact that multiple orientations may have similar or identical appearances. Possible ways to mitigate this deficiency are discussed in Section 3.5.2, and any strictly visual navigation process suffers from the same issue.

3.2 Theoretical basis

3.2.1 Image compression

One of the key components to appearance matching is image compression. Even a small image, e.g. 150 by 150 pixels, results in 22500 array elements for each training or test image. In appearance matching, each image is represented by a point in a multi-dimensional space. Searching such a space with 22500 dimensions is not ideal from a mathematical or computational perspective. Therefore, the dimensionality of each image must be reduced. This is done through image compression.

One type of image compression is singular value decomposition (SVD). SVD consists of decomposing an $m \times n$ matrix $\mathbf{A}$ into three component matrices: two unitary matrices $\mathbf{U}$ ($m \times m$) and $\mathbf{V}$ ($n \times n$) as well as a rectangular diagonal matrix $\mathbf{\Sigma}$ with same dimensions as $\mathbf{A}$, $m \times n$. The original matrix $\mathbf{A}$ can be reconstructed from these component matrices (Equation 3.1)

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{3.1}$$

The diagonal values of $\mathbf{\Sigma}$, denoted σ_i, are referred to as "singular values" and are ordered from greatest to least. Using an SVD, a matrix can be approximated by reconstructing using a subset of these principal values. Typically, that subset is the largest k values, where $k < r$, and r is the smaller of m and n. Thus, the matrix $\mathbf{A}$ can be approximated by Equation 3.2, where $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the ith column vectors of the $\mathbf{U}$ and $\mathbf{V}$ matrices and are known as the principal $\boldsymbol{u}$- and $\boldsymbol{v}$-components.

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{3.2}$$

When the $\mathbf{A}$ matrix in question is an image, this process is known as SVD image compression. The fewer of these singular values are used in the reconstruction, the more distorted is the resulting image that is produced. The Eckhart-Young-Mirsky Theorem [47, 48] proves

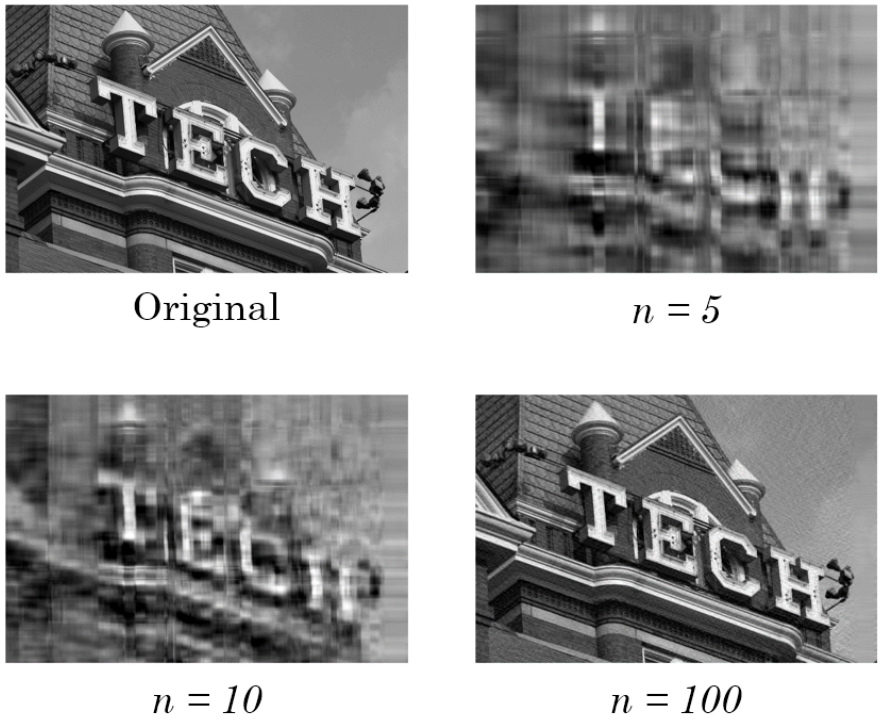

Figure 3.3: Image of tech tower under various levels of compression (Original image [49])

that a singular value decomposition minimizes the error of compression when compared with other types of transformations. Figure 3.3 shows an undistorted image and a reconstructed image using the largest 5, 10, and 100 singular values.

3.2.2 Intelligent library sorting and nearest neighbor search

One common task in computer science is solving the following problem. A set of data points is given in an n-dimensional space. A test point is then introduced and the closest point of the original set to that point must be found. This problem is referred to as finding the "nearest neighbor". Often times the nearest neighbor search is expedited by intelligently sorting the library of existing points to speed up the search.

During both the object recognition and attitude determination steps of appearance matching, a nearest neighbor in a training set must be determined for a test point that has been projected into the multidimensional space. The number of these training points varies based

on the original training parameters and the amount of interpolation, but there could easily be thousands or tens of thousands of points which must be searched twice: once for object recognition and once for attitude determination. Unlike the training processes, the search for a nearest neighbor takes place on the spacecraft computer. Therefore it should be as computationally efficient as possible. In [1], Murase and Nayar reference an algorithm for a closest point search in higher dimensions to solve this particular problem [50].

Subsequent research presents a better alternative. Work by Weber, et al., [51] indicates that even a simple linear scan can outperform partitioning techniques like those described in [50] for applications with large dimensionality. "Large" in this context is defined as 10 dimensions or greater, which is the exact number used in [1]. Weber also references an even better method using a *vector-approximation file* (VA-file method). According to Weber and Blot [52], the VA-file method outperforms both partitioning methods like R-trees and linear scans for high-dimensionality applications.

An even more recent strategy is multiple random projection trees (MRPT) by Hyvönen et al. [53]. MRPT is a nearest-neighbor search method designed for use with data sets that have high dimensionality. According to Hyvönen et al., the method outperforms other common nearest-neighbor search methods for almost all data sets tested. In addition, the performance improvement for MRPT becomes more pronounced the greater the dimensionality of the set. For appearance matching, this fact helps offset the additional computational cost of using more eigenvalues in the compression. The relationship of this search method to appearance matching is explored further in Section 3.3.2.3.

3.2.3 Karhunen–Loève transform

The appearance matching algorithm described by Murase and Nayar [1] and referenced in this dissertation is the discrete form of the Karhunen–Loève Transform (KLT). The original idea of decomposing a continuous signal by decorrelating neighboring components was simultaneously developed by Kari Karhunen [54] in 1947 and Michel Loève [55] in 1948.

The concept of stochastic processes expressed as infinite series in this way was first proposed by Damodar Dharmananda Kosambi [56] in 1943 and the procedure is sometimes cited as the Kosambi–Karhunen–Loève Transform.

The discrete KLT is known by various names depending on the field in which it is being used: discrete KLT in signal processing, principle component analysis (PCA) in image processing, empirical orthogonal functions in meteorology, and the Hotelling transform in multivariate quality control, among others. The latter name results from its independent development by Harold Hotelling for statistical analysis in psychology in 1933 and expanded in 1936 [57, 58]. Murase and Nayar's work refers to the transform as a KLT, but in reality their algorithm is more closely related to PCA: a set of zero-mean column-wise data is combined into a data matrix. The covariance of that data matrix is computed and the eigenvalues and corresponding eigenvectors are found. This set of mutually orthogonal eigenvectors forms the basis of a multidimensional space. It should be noted that the percentage of variance in the data captured by each dimension is equal to the value of associated eigenvalue divided by the sum of all of the eigenvalues. This is the key advantage of PCA: given the standard ordering of eigenvectors according to descending value of associated eigenvalue, a set of the first k eigenvalues will capture the variance in the data better than any other set of k eigenvalues. This property also can be used to tune the number of eigenvectors used in appearance matching, which will be discussed further in Section 3.5.1.

Both KLT and and PCA are related to SVD in that they seek to determine the directions of greatest variance in a multidimensional space. Some variations of PCA, including the one presented in this research, utilize SVD in place of a simple eigenvalue decomposition of the covariance matrix, and the term SVD is often used in place of PCA, even though the two processes are not strictly the same. When PCA is performed on an image library, it has the effect of compressing the image data in a way similar to what was described in Section 3.2.1.

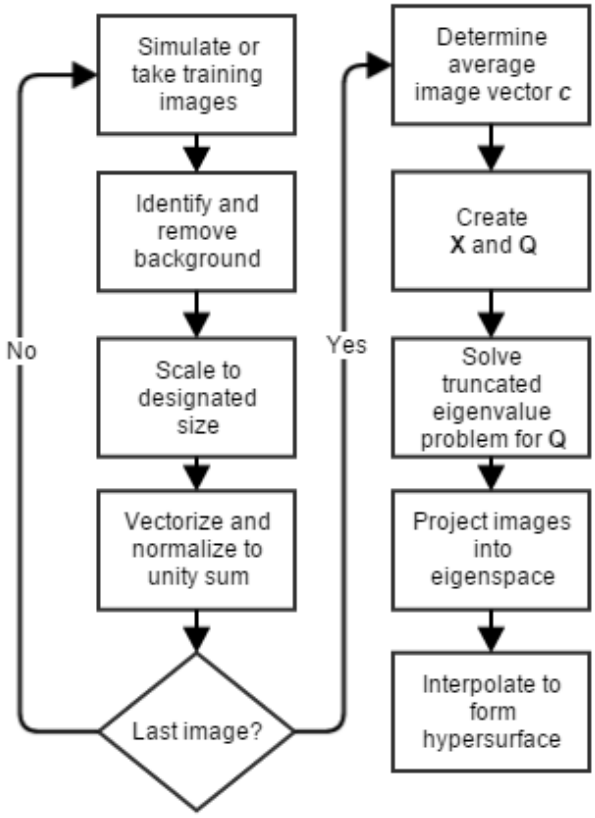

Figure 3.4: Appearance training block diagram

3.3 Algorithm

3.3.1 Training procedure

Before a test image is analyzed, the appearance matching algorithm must learn the appearance of an object using a training image set. First, the images are normalized. Extraneous background pixels are cropped out, the image is scaled to the desired size, and the overall light intensity is normalized. Next, the universal eigenspace is constructed from every image in the training set. Finally, eigenspaces are constructed for each object using only images of that object.

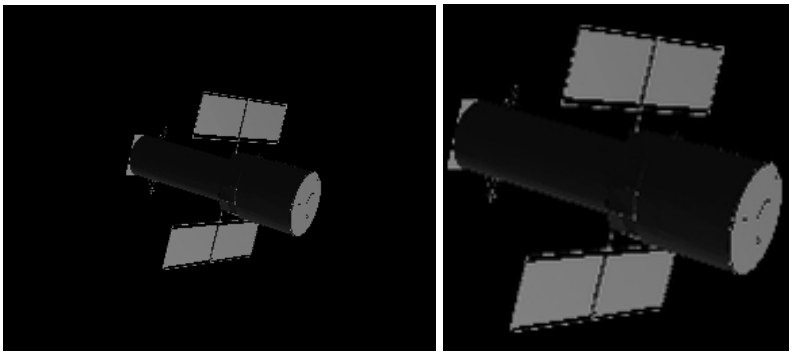

Figure 3.5: Original training image (left) and cropped and scaled image (right)

3.3.1.1 Normalization

The first step in appearance matching image set is normalization. This process begins by cropping each image to include as little of the background as possible. Typically the cropped image will have a square aspect ratio, but not necessarily. This result is then scaled to a specified size, e.g., 150 by 150 pixels. These scaled images are vectorized by reading the brightness values horizontally in a raster scan pattern to form the image vector $\hat{\mathbf{x}}$ (Equation 3.3). To account for varying lighting intensity and exposure times, each image vector is scaled so that the sum of the vector is unity, resulting in the normalized image vector $\mathbf{x}$ (Equation 3.4).

$$\hat{\mathbf{x}} = [\hat{x}_{1,}\hat{x}_2, \ldots, \hat{x}_n]^T \tag{3.3}$$

$$\mathbf{x} = \frac{\hat{\mathbf{x}}}{\sum_{i=1}^{n} \hat{x}_i} \tag{3.4}$$

3.3.1.2 Universal eigenspace and hypersurface

The normalized image vectors are combined to form image sets. The universal image set consists of every training image, while the object sets contain all the images of a single object. In the following equations, r represents the orientation index, l is the lighting index,

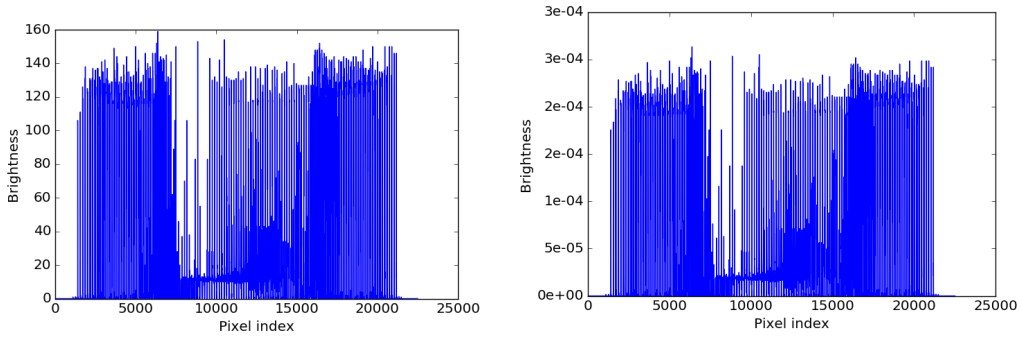

Figure 3.6: Brightness vector ($\hat{\mathbf{x}}$, left) and normalized brightness vector ($\mathbf{x}$, right)

and p is the object index. The average image $\mathbf{c}$ can be calculated by summing each column of the universal image set and dividing by the total number of images (Equation 3.5).

$$\mathbf{c} = \frac{\sum_{p=1}^{n_p} \sum_{r=1}^{n_r} \sum_{l=1}^{n_l} \mathbf{x}_{r,l}^{(p)}}{n_p \cdot n_r \cdot n_l} \tag{3.5}$$

The average image $\mathbf{c}$ of the universal image set is used along with the training data to construct the universal eigenspace. The image matrix $\mathbf{X}$ is found by subtracting the universal average image from each $\mathbf{x}$ and combining them as shown in Equation 3.6. The image matrix is then used to find the symmetric covariance matrix $\mathbf{Q} \triangleq \mathbf{X}\mathbf{X}^T$.

$$\mathbf{X} \triangleq \{\mathbf{x}_{1,1}^{(1)} - \mathbf{c}, \ldots, \mathbf{x}_{n_r,1}^{(1)} - \mathbf{c}, \ldots, \mathbf{x}_{n_r,n_l}^{(p)} - \mathbf{c}\} \tag{3.6}$$

Finding the full set of eigenpairs of $\mathbf{Q}$ would be impractical, so only the first k pairs of eigenvalues λ_i and eigenvectors $\mathbf{e}_i$ are calculated. According to Murase and Nayar [1], a value for k of 10 was sufficient for learning and recognition. However, more eigenvalues can improve the accuracy of the matching process at the expense of greater computation and storage requirements. This trade-off is explored in greater detail in Section 3.5.1. Each image in the universal set is located as a point in the k-dimensional eigenspace $\mathbf{g}_{r,l}^{(p)}$, corresponding to the pth object at the rth orientation and lth lighting condition. (Equation 3.7). If desired, these points are interpolated to form the hypersurface $\mathbf{g}(\theta_1, \theta_2)$, where θ_1 and θ_2 are the indices of the orientation and illumination conditions, respectively.

$$\mathbf{g}_{r,l}^{(p)} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k]^T (\mathbf{x}_{r,l}^{(p)} - \mathbf{c}) \tag{3.7}$$

3.3.1.3 Object eigenspaces and hypersurfaces

The images of each object p are projected into the appropriate object eigenspace by first creating the object-specific image matrix $\mathbf{X}^{(p)}$ (Equation 3.9) and calculating the resulting eigenspace from the object covariance matrix $\mathbf{Q}^{(p)} \triangleq \mathbf{X}^{(p)}(\mathbf{X}^{(p)})^T$. The average image for object p, $\mathbf{c}^{(p)}$, is found by averaging the image vectors of each object (Equation 3.8).

$$\mathbf{c}^{(p)} = \frac{\sum_{r=1}^{n_r} \sum_{l=1}^{n_l} \mathbf{x}_{r,l}^{(p)}}{n_r \cdot n_l} \tag{3.8}$$

$$\mathbf{X}^{(p)} \triangleq \{\mathbf{x}_{1,1}^{(p)} - \mathbf{c}, \ldots, \mathbf{x}_{n_r,1}^{(p)} - \mathbf{c}, \ldots, \mathbf{x}_{n_r,n_l}^{(p)} - \mathbf{c}\} \tag{3.9}$$

The first k pairs of eigenvalues $\lambda_i^{(p)}$ and eigenvectors $\mathbf{e}_i^{(p)}$ are calculated as well the hypersurface $\mathbf{f}^{(p)}(\theta_1, \theta_2)$ from the points $\mathbf{f}_{r,l}^{(p)}$ (Equation 3.10), similar to the procedure followed for the universal space and hypersurface.

$$\mathbf{f}_{r,l}^{(p)} = [\mathbf{e}_1^{(p)}, \mathbf{e}_2^{(p)}, \ldots, \mathbf{e}_k^{(p)}]^T (\mathbf{x}_{r,l}^{(p)} - \mathbf{c}^{(p)}) \tag{3.10}$$

3.3.2 Testing procedure

The preceding steps constitute the initialization of the algorithm. They are performed only once and the resulting parameters are stored. These parameters are the universal and object average images, the universal and object eigenvector sets, and the universal and object interpolated hypersurfaces.

Processing a test image is a two step process. First, the image is projected into the universal eigenspace in order to determine the object identity. Then, the image is projected into the appropriate object space in order to determine the relative attitude.

3.3.2.1 *Object identification*

For each test image received, the same normalization procedure is performed as that for the training images. A image vector $\hat{\mathbf{y}}$ (Equation 3.11) is formed from the cropped and scaled test image, which is then normalized. The algorithm maps the resulting normalized test image vector $\mathbf{y}$ (Equation 3.12) to location $\mathbf{z}$ in the universal eigenspace for object recognition (Equation 3.13). The object p that minimizes the distance between $\mathbf{z}$ (Equation 3.14) and the universal hypersurface $\mathbf{g}(\theta_1, \theta_2, p)$ is determined.

$$\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n]^T \tag{3.11}$$

$$\mathbf{y} = \frac{\hat{\mathbf{y}}}{\sum_{i=1}^{n} \hat{y}_i} \tag{3.12}$$

$$\mathbf{z} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_k]^T (\mathbf{y} - \mathbf{c}) \tag{3.13}$$

$$d_1 = \min_{\theta_1, \theta_2} ||\mathbf{z} - \mathbf{g}(\theta_1, \theta_2, p)|| \tag{3.14}$$

If the match has a high enough confidence, then the image is determined to include object p. This confidence threshold is one of the tuning parameters of the algorithm and will vary from application to application.

3.3.2.2 *Relative attitude determination*

In order to perform pose estimation, the image is then mapped into location $\mathbf{z}^{(p)}$ in the object p eigenspace (Equation 3.15). The indices θ_1 and θ_2 that result in the minimum value of $d_2^{(p)}$ determine the attitude of the object and the closest illumination condition (Equation 3.16).

$$\mathbf{z}^{(p)} = [\mathbf{e}_1^{(p)}, \mathbf{e}_2^{(p)}, \ldots, \mathbf{e}_k^{(p)}]^T(\mathbf{y} - \mathbf{c}^{(p)}) \tag{3.15}$$

$$d_2^{(p)} = \min_{\theta_1,\theta_2} ||\mathbf{z}^{(p)} - \mathbf{f}^{(p)}(\theta_1, \theta_2)|| \tag{3.16}$$

3.3.2.3 Library search and confidence

Various methods for intelligent library search were described in Section 3.2.2 with Hyvönen et al. determined to be the best option. This technique is able to search for the k_{nn} nearest neighbors in a space, and different numbers of nearest neighbors are used depending on the stage of appearance matching. For the object identification step, several nearest neighbors are found and compared to see if they matched the actual nearest neighbor. Based on this level of agreement, a confidence value c for the object identification step from 0 to 1 is determined using Equation 3.17. For this equation i is the index of the neighbor, d_i is the distance to neighbor i from the the nearest neighbor search, and $\delta_{i,1}$ is a parameter that is 1 if the identity of the ith neighbor matches the nearest neighbor and 0 if it does not. The result is a weighted match percentage of the k_{nn} nearest neighbors to the nearest neighbor.

$$c = \frac{\sum_{i=1}^{k_{nn}} \frac{\delta_{i,1}}{d_i}}{\sum_{i=1}^{k_{nn}} \frac{1}{d_i}} \tag{3.17}$$

For attitude determination, the process works somewhat differently. Nearest neighbors are useful to verify object identity since all of those neighbors should share the same identity for a good match. That is not the case for attitude determination, where neighbors have different attitudes than the attitude that needs to be verified. As such, reconstruction error is used instead as way to produce an attitude confidence.

Since appearance matching uses a form of image compression, the location $\mathbf{z}^{(p)}$ may be reconstructed into an image using the eigenvectors from the object p eigenspace according to Equation 3.18. This process reverses the steps of acquiring the eigenspace location from

the original image $\hat{\mathbf{y}}$.

$$\hat{\mathbf{y}}_r = \left(\sum_{i=1}^{n} \hat{y}_i \right) \left([\mathbf{e}_1^{(p)}, \mathbf{e}_2^{(p)}, \ldots, \mathbf{e}_k^{(p)}] \cdot \mathbf{z}^{(p)} + \mathbf{c}^{(p)} \right) \tag{3.18}$$

This reconstructed image is compared with the original image in order to find the per-pixel error, which is inverted to produce a confidence value that increases as the pixel error decreases as shown in Equation 3.19. The number of pixels is n and the resulting confidence value is ν.

$$\nu = n / \left(\sum_{i=1}^{n} |\hat{y}_i - \hat{y}_{r,i}| \right) \tag{3.19}$$

The confidence values for object identification and attitude determination are necessary for the implementation of the sensor fusion process later described in Chapter 5 as well as for the use of appearance matching in a single- or multi-spectral framework with a navigation filter.

3.4 Robustness of appearance matching

The standard appearance matching algorithm cannot account for test image backgrounds which differ from those in the training library. This limitation results from the way that the image library is projected into the universal and object eigenspaces. The PCA transform works by determining the principal directions by which images in the library are differentiated. When PCA is run on images with a black background, the image locations of greatest differentiation are those where an object is present in some orientations and not in others. These locations tend to be approximately halfway between the image center and its edge.

When the test image is also on a black background, this requirement does not present a problem. Figure 3.7 shows a Stardust satellite test image and the reconstruction of that image from its eigenspace projection. However, any non-black background disrupts the PCA. Figure 3.8 compares an image on a non-black background and its reconstruction

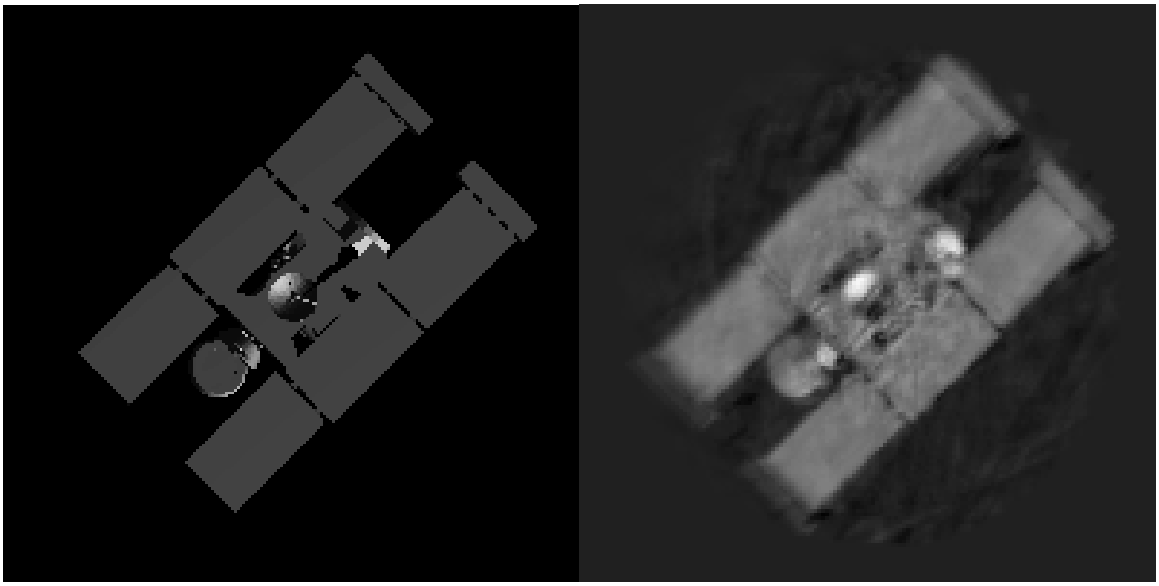

Figure 3.7: Simulated image with black background (left) and reconstruction (right) from a black-background eigenspace.

3.4.1 Fast Robust PCA

Since object recognition and attitude determination algorithms based on PCA are particularly susceptible to outlier points like noise, occlusion, and, as was shown in Section 3.4, non-black backgrounds, techniques have been developed to help improve the performance of PCA. One way to increase the performance of PCA in such situations is known as robust PCA. This technique has been proposed by multiple researchers, including Leonardis and Bischof [59]. A more modern version with better computational efficiency called fast robust PCA (FR-PCA) has since been presented by Storer, et al. [60].

The concept behind robust PCA is to break the image into multiple subimages. These images are formed by creating a series of random samples. The samples are then applied to each full image in the training set, and the resulting data is referred to as a "subimage." These subimages are typically about 1% of the total pixels in the image. The number of samplings that are created is a tunable parameter. 1000 samplings were created in tests done in [60]. Each of these samplings is applied to each image in the training library,

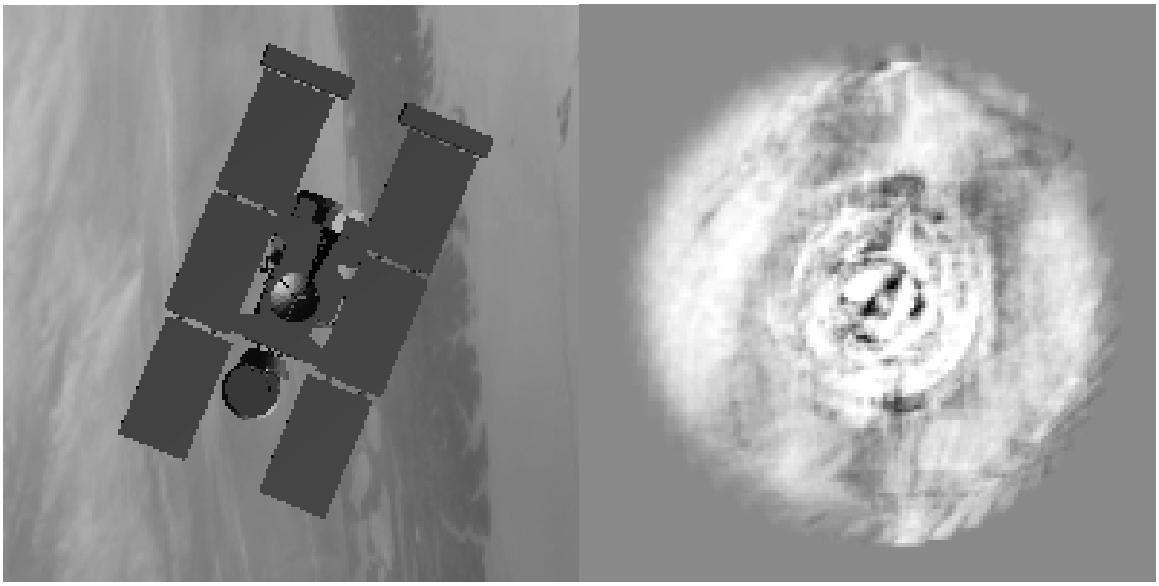

Figure 3.8: Simulated image with cloud background (left) and reconstruction (right)

creating a series of subimage sets. PCA is applied on each one of these subimage sets in the same way that it was applied to different objects in the standard PCA formulation. The result is a number of so-called "sub-subspaces", one from each sampling of the original image set. An overall subspace is created using PCA from the full image as well. Figure 3.9 outlines the FR-PCA training procedure.

Given a test sample, FR-PCA attempts to create a robust reconstruction of the image without any outlier points present. This reconstruction is created in two steps: gross outlier detection and refinement. In the first part, the test image is subsampled and each of the sample points is reprojected using the appropriate sub-subspace. If the reprojected point is within a certain tolerance of the actual test point, the point is kept. If not, it is rejected as an outlier. Once these outliers are removed, a series of reconstructions are created with the worst reconstruction errors being thrown out until the number of inliers reaches a predetermined threshold. This process is iterative, starting with the test image being reconstructed from the current set of inliers. The reconstruction is compared with the true image, and the pixels are ranked based on their reconstruction error. Then a given percentage, say 10%, are removed and the steps are repeated.

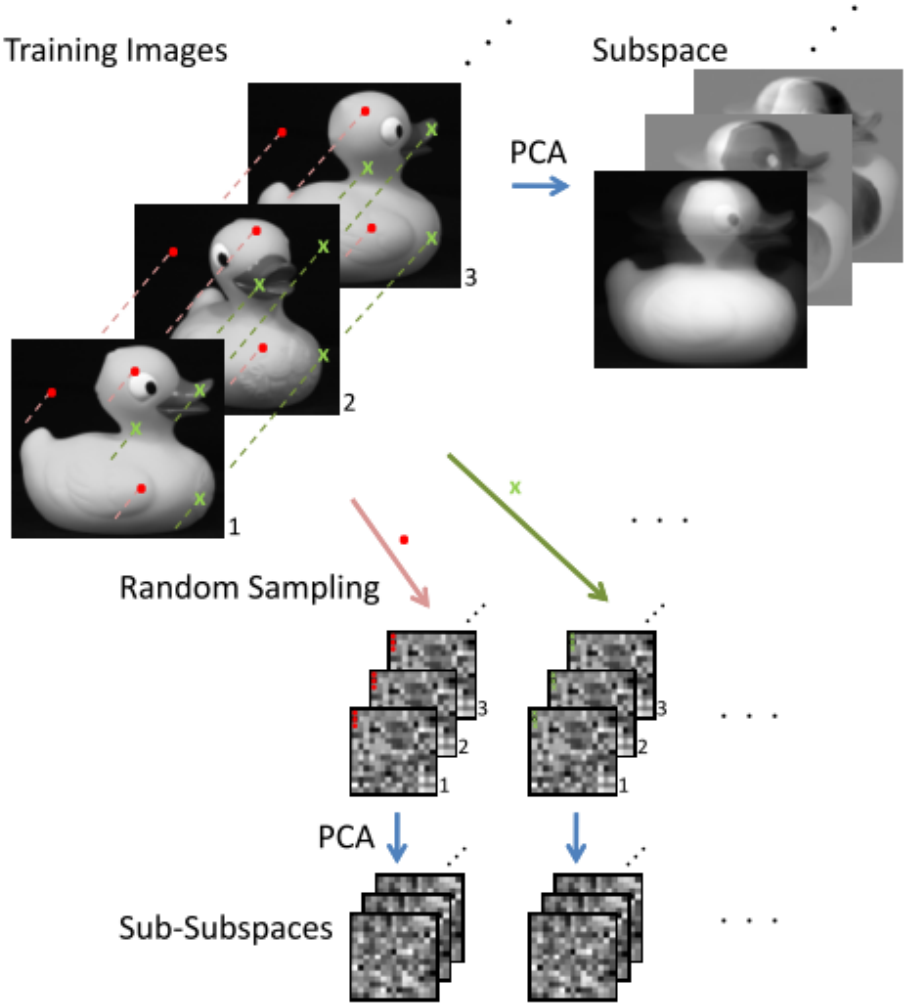

Figure 3.9: FR-PCA training procedure [60]

Figure 3.10: Occluded test image (left), reconstruction using standard PCA (center) and using FR-PCA (right) [60]

FR-PCA and robust PCA in general perform very well on outliers like occlusions and noise. Figure 3.10 shows a test case for an image of a duck with an occlusion in the upper half [60]. For the case of a non-black background, it struggles since the number of points being "rejected" is so high. For this reason, an alternate strategy for dealing with varying backgrounds was created.

3.4.2 Background randomization

In order to make the algorithm useful on an arbitrary background, PCA must be modified so that it de-emphasizes those image locations which are sometimes background and sometimes not. The result is that the object is identified and its orientation is distinguished using differences in lighting, shadow, and texture. This change is implemented via background randomization of the training images.

Since the training images are simulated via ray-tracing, the background is defined by any location where a cast ray does not intersect with the object. The ray-tracing procedure is explained in greater detail in Section 4.5.4. The background can then be replaced with anything. For this process, the background pixels are substituted with a random integer based on the possible brightness values for the image. For example, they range from 0 (black) to 255 (white), if the images are 8-bit greyscale. A sample training image with the background randomized is shown in Figure 3.11.

The random pattern is generated separately for each training image. Therefore, the image locations that differentiate between the images are changed from the black back-

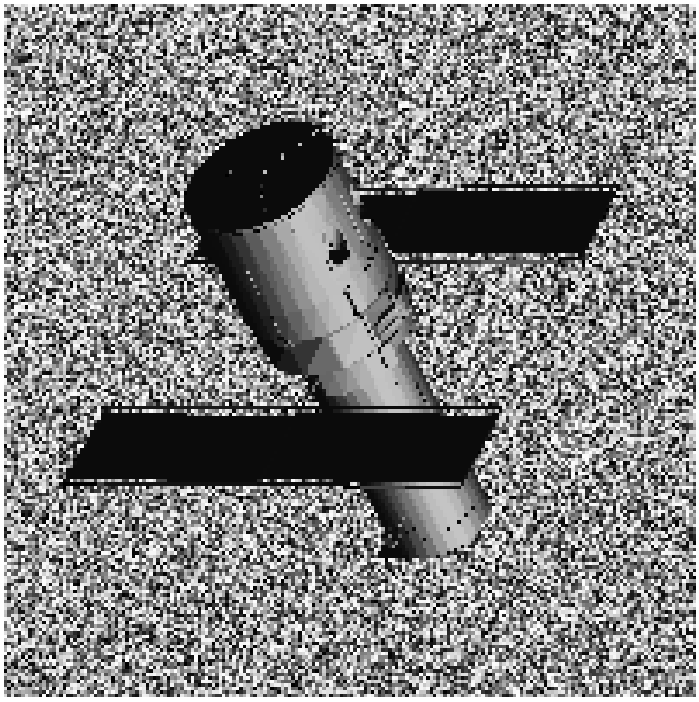

Figure 3.11: Sample random-background training image

ground case. The resulting improvement is seen by comparing the reconstruction using a random-background eigenspace (Figure 3.12) to the previous reconstruction using a black-background training library (Figure 3.8).

Unfortunately, background randomization does come with its own drawback, albeit one that may be mitigated. Because the black background with brightness zero is being replaced with non-zero brightness values, the overall light intensity of the image is raised. That, in turn, affects the value of the normalized image vector $\mathbf{x}$. In cases where the test image has a background with similar overall intensity, this change does not affect the accuracy of the object identification or attitude determination.

However, when the average background intensity varies significantly from half of the maximum range of brightness, e.g., when the test image background is black or white, then the process is affected. Specifically, the normalization of the image vector described in Section 3.3.1.1 causes a discrepancy between the black-background test image and the random-background training image. For the case when the background is black or nearly

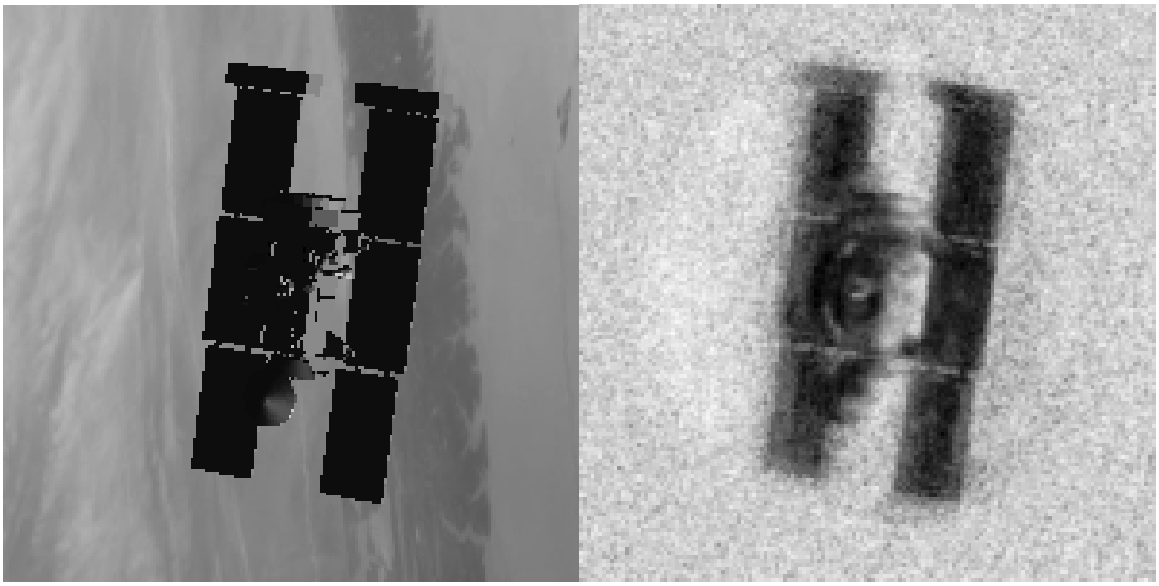

Figure 3.12: Simulated image with cloud background (left) and reconstruction from random-background eigenspace (right)

black, this problem is remedied by replacing the black background of the test image with a randomized background, which is much more straightforward than the inverse process of replacing a non-black background with a black one.

Note that this procedure is not perfect. A black or very dark pixel does not guarantee that the image location is background. It could also be an unilluminated portion of the object. When using visible spectrum images, this error is unavoidable when employing background randomization. It may be mitigated when sensor fusion is employed using a temperature threshold, a process explained in Section 5.1.1.3.

3.5 Tuning and observability

There are two main parameters to be tuned to adjust the computational performance and accuracy of the appearance matching algorithm: the quantity of the training images and the number of eigenpairs used to construct the subspaces. An argument could be made that the resolution of the image being used for training is also one of these parameters. However, the PCA process is essentially a compression of the training images. Thus the effective

image quality used for the match depends much more on the number of eigenpairs than it does on the resolution of the image.

3.5.1 Quantity of eigenpairs

Because the PCA process is a form of image compression, the more eigenpairs that are calculated in the partial SVD the better the quality of the reconstructed image. Thus more features of the target are used to find a match. The downside to having more eigenpairs is that it increases the dimensionality of the subspace and therefore the complexity of the nearest neighbor search. The memory requirement on the spacecraft is also greater.

Therefore, a balance between efficiency and performance must be found. One of the properties of PCA can assist with this determination. As previously mentioned in Section 3.2.3 the amount of variance captured by a particular principal direction is the percentage of that direction's associated eigenvalue to the sum of all eigenvalues. An example of this relationship is given in Figure 3.13 for a sample set with 162 orientations, 4 illuminations and 3 objects. Thus, if the percentage of captured variance is set as a parameter, it determines the number of eigenvalues required for a given image set. For consistency, this value is chosen for the universal set and then the same number of eigenpairs are used for the object sets.

3.5.2 Quantity of training images

The key concern when it comes to tuning the number of required training images is the amount of acceptable error in the relative attitude determination step. Assuming a correct identification, an upper bound on the attitude error is half of the difference in attitude between any two successive training images. Figure 3.14 shows the relationship between the number of orientations and the maximum error bound. This difference can be lessened somewhat by interpolating the hypersurface based on training images, but too much interpolation leads to an increase in error as well, since the interpolated points will not have

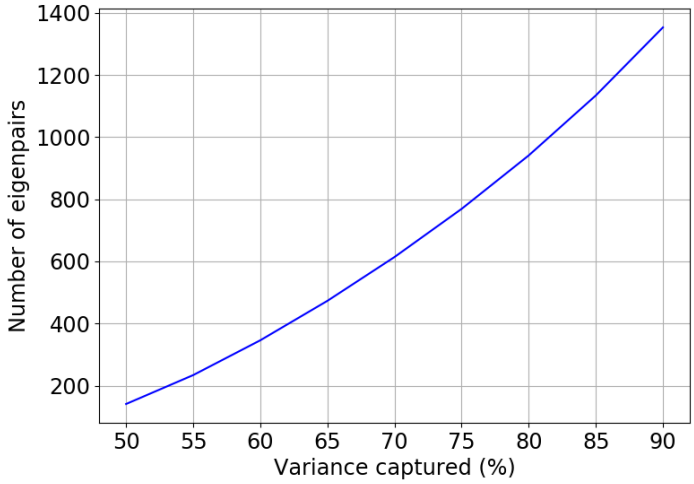

Figure 3.13: Eigenpairs versus variance captured

exactly the same parameters as a training image at that same attitude.

Additionally, increasing the number of interpolated or actual training images has a similar effect to increasing the number of eigenpairs, in that the computational time to perform the nearest neighbor search and the memory storage requirement on the spacecraft both increase with a larger number of hypersurface points. The former relates directly to how quickly a relative attitude solution can be returned to the attitude determination system of the spacecraft. As with many attitude sensors, the accuracy and speed of the solution must be balanced based on the requirements of the mission in question.

As mentioned previously in Section 3.1.2, appearance matching runs into difficulty when attempting to distinguish between attitudes with similar appearances. The first instance of this phenomenon is similar orientations producing similar appearances. For example, training images of the object which are taken at 20 and 21 degrees of roll. This case is less problematic because while additional error is introduced, it is only a multiple of the maximum error value discussed earlier. It is mitigated by increasing the number of eigenpairs or resolution of the training images so that similar orientations are more distinct from one another.

The more problematic case is two orientations with similar appearances that differ

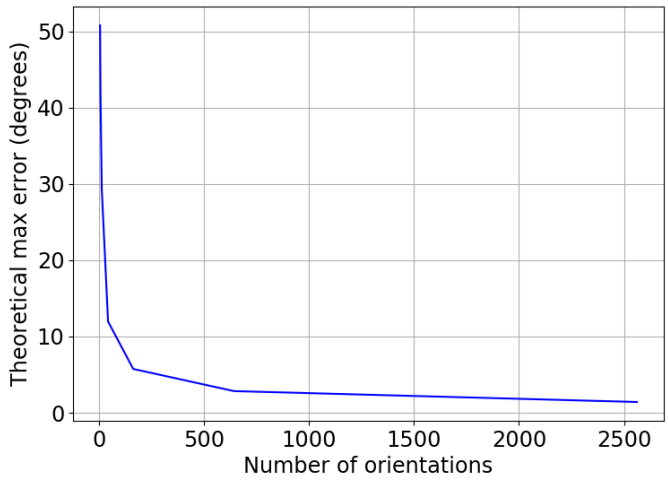

Figure 3.14: Theoretical maximum error vs number of training images

greatly in orientation. Symmetric objects may exhibit this type of error, since symmetry itself is defined as identical appearance under a transformation. The algorithm has no way of rectifying this situation on its own, although it would be able to flag a potential issue since multiple points would exist with nearly identical confidence values. Such errors due to symmetry would have to handled intelligently by a dynamic filter or similar algorithm processing the measurements to produce a unique navigation solution.

3.5.3 Observability

Of key concern to any navigation system is the notion of observability. Observability is the measure of how well the states of a system can be inferred from knowledge of its measured external outputs. In the spacecraft navigation problem described in this research, the states are the position and orientation of the target spacecraft relative to the sensing spacecraft. A related and weaker requirement is known as detectability, which is a condition that allows unobservable states as long as they are bounded. In the general problem of spacecraft navigation, no state can be guaranteed to be stable, so the stronger observability condition is necessary.

For the two vehicle spacecraft relative navigation problem, three degrees-of-freedom in

relative position are combined with three degrees-of-freedom in relative orientation for a total of six degrees-of-freedom. The dynamics of a spacecraft in orbit subject to gravity are non-linear, but the observability analysis is simplified by linearization. If the linearized system is observable at any point, then the non-linear system is observable as well (Bonnifait and Garcia [61]).

To make the linear system observable, at least 6 linearly independent measurements must be provided. Here the derivation of appearance matching is an advantage. The underlying basis vectors found in the training process are chosen to be linearly independent. As long as there are at least 6 eigenvectors resulting from a training library of at least six images, the measurement inputs to appearance matching will form part of an observable system. Interestingly, it does not matter what the images look like, as long as they are distinct. Even an all-black test image matching up to a similar looking training image will be observable, as long as that training image is unique in the training library.

CHAPTER 4

SPACECRAFT IMAGE SIMULATION ENVIRONMENT

One of the requirements of the appearance matching algorithm is the creation of a large set of training images of the various target objects at different orientations and under different lighting conditions. Certainly the option exists to use actual images of the objects taken from a camera. However, this process is time consuming and prone to error if the attitude of the target cannot be measured exactly. The target objects also may not be available to be photographed. This is frequently the case for spacecraft, which may not be available for photography under the correct environmental conditions prior to flight. Therefore, image simulation is a good alternative for the creation of the necessary training image sets.

This chapter first presents the fundamentals of image simulation, including solving the visibility problem, object modeling using a triangular mesh, and radiation simulation. Modifications to the image simulation process to accommodate the infrared spectrum are also described, including the framework for simplified thermal generation. The current state-of-the-art in image simulation is briefly described and motivation is presented for why a new tool needed to be developed.

This software tool is called the spacecraft imaging simulation environment (SISE). The SISE was developed specifically for this research for two purposes. First, it is used to generate the simulated images in both the visible and infrared spectra needed to train the appearance matching algorithm described in the preceding chapter. Second, the SISE is a useful tool in performing software-in-the-loop tests in order to both verify the intended performance of appearance matching for spacecraft, and also to assess the impact of tuning parameters like quantity of training images and number of eigenpairs. The SISE consists of four parts: initialization, target modeling, radiation simulation, and file creation and refinement. The order of these steps is given by Figure 4.1.

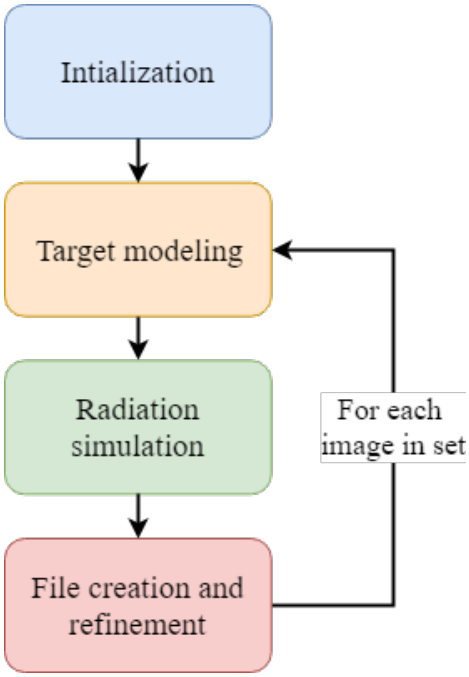

Figure 4.1: High-level SISE block diagram

Initialization loads the necessary parameters about the cameras and scene geometry as well as specifications about the camera's sensors and the target. Target modeling rotates the vertices and faces of the target model into a camera-centered frame and translates them to the appropriate location. Radiation simulation is the heart of the SISE. The reflected and emitted radiation from the target is modeled as well as the response of the visible and infrared sensors. Final refinement is added to the simulated image and the file is saved. This step involves the application of the following relevant camera errors.

- Radial and tangential distortion: the warping of the image due to physical imperfections in the camera lens
- Blur: a reduction in image sharpness due to object motion or physical defects in the camera optics
- Noise: image artifacts resulting from the electronic and digital processing of the image

Images generated by the SISE are rendered efficiently using CUDA. CUDA was introduced

in 2007 by Nvidia as a parallel computing platform and application programming interface (API). Originally standing for Compute Unified Device Architecture, CUDA allows direct access to the instruction set and parallel computational elements on a graphics processing unit (GPU). The CUDA interface utilizes "compute kernels", which are subroutines that are executed in parallel on each one of the GPU cores.

4.1 State of the art and contribution to the field

The field of image simulation is one that has been explored extensively, particularly within the last 20 years. More advanced video games and increased used of computer-generated imagery (CGI) in film and television has led to the development of numerous tools for creating more realistic images. Renderman is a program developed by Pixar for use in their 3D animated movies as well as to license to third parties [20]. Starting in March 2015, it was made available for non-commercial use. Renderman originally used an algorithm known as REYES (Renders Everything You Ever Saw) [62] developed at LucasFilm's Computer Graphics Research Group, Pixar's predecessor. REYES was designed to work on the more primitive computer architecture available in the 1980s and 90s and thus attempted to minimize ray-tracing as much as possible. Instead, it favored more efficient, less photorealistic approaches. In 2016 Renderman transitioned to a Monte Carlo-based path tracing approach. This technique allows for a faster "first-pass" render, but ultimately takes longer to converge to the same quality as REYES [63].

Mental Ray is another application used to generate photorealistic images [21]. Mental Ray was originally developed by German company Mental Images, which was bought by Nvidia in 2007. As its name suggests, Mental Ray focuses on ray tracing to generate its simulated images. While the application has been widely used for CGI in feature films, Mental Ray is also accompanied by an application programming interface (API) that allows it to interface with other programs. As such, software such as Autodesk, AutoCAD, and Solidworks use Mental Ray to create simulated images within their programs. Mental Ray's

main feature is the ability to use parallel processing to take advantage of multiprocessor machines and so-called "render farms" of multiple separate computers.

Though much of the work in the field of simulated imaging has been proprietary, an effort has been made in the computer graphics community to develop an open-source option. The result is Blender, a free and open-source graphics toolkit currently on version 2.79 [19]. Blender started out as proprietary software developed by Ton Roosendaal for Neo Geo, an animation studio based out of the Netherlands. Eventually the program was acquired by the open-source community and its code became freely available in September 2002. Blender has been primarily updated and expanded by community effort ever since. In addition to rendering simulated images, Blender has modules for visual effects, 3D printing, and an integrated game engine. Blender creates simulated images via ray tracing that can be accelerated using the GPU.

Given all of these options, including Blender that is free and open-source, it is important to demonstrate why a separate tool was developed specific to this research. Four factors went into this decision: the ability to perform infrared simulation, the ease of modifying the environment to include background randomization, the ability to automate the creation of simulated images to train the appearance matching algorithm, and a cost that is not prohibitive for academic research.

4.1.1 Infrared simulation

Since one of the main contributions of this research is the fusion of visible and infrared images, it is vital that the rendering environment has the ability to simulate images in both spectra. Based on the limited information available on Renderman and Mental Ray and the more extensive open-source documentation on Blender, none of these environments is able to generate simulated infrared images. All three have the ability to generate images that appear to be infrared, but the effect is created by modifying the color and shading of a visible image. While this effect is useful for film or television CGI, at least a basic thermal

simulation is needed for this research to appropriately demonstrate sensor fusion.

4.1.2 Background randomization

As described in the previous chapter, a randomized background is added to the appearance matching training image in order to allow the algorithm to operate on an arbitrary background. In order to do so, it is necessary to identify those pixels in the image which were not part of the target so that they may be replaced. This information must be provided by the rendering program. Both Renderman and Mental Ray utilize proprietary algorithms for which the source code is not available, making this kind of modification extremely difficult. Blender offers the best option to implement a subroutine for background randomization, so long as it satisfies the other requirements.

4.1.3 Automation

Additionally, the chosen rendering environment must have an interface that allows the automated creation of simulated images. Since all three of these environments were designed with film or video games in mind, they are good at creating a scene which progresses in space and time, like walking through a forest or approaching a spacecraft. However, appearance matching requires that multiple views of an object are taken at the same location and time but from different orientations and under different lighting conditions. The result is that in order to use one of these environments, the scenario that is used as an input must be rigged so that it creates the required set of images. In fact, the first version of the SISE [25] had this same interface and it was abandoned since the image generation needed for appearance matching was inefficient.

4.1.4 Cost

The final barrier to use one of these applications is cost, both in time and in money. Monetarily, both Renderman and Mental Ray must be licensed from their respective companies

and at the start of this research (2013), no academic discount was offered. The cost was therefore out of the budget of academic research. Blender has the advantage of being a free, open-source program. The cost associated with conducting the research was in learning and then applying the necessary coding and API. That cost had to be balanced against the time cost for the development of a program specifically for the purpose of generating and testing images for appearance matching and sensor fusion.

4.1.5 Contribution

After taking into account all of the various options, it was determined that development of an environment (SISE) specific to this research was the best option. The SISE seamlessly integrates with the necessary functions to perform appearance matching as well as those for sensor fusion. The images generated by SISE are not as photorealistic as those from Blender or one of the other aforementioned applications. However, that image quality is not necessary for adequate performance of the visual navigation algorithm and the advantages to an in-house application outweigh the drawbacks. Finally, the SISE will continue be developed with the aim of supporting similar research in both of these topics in the future. The goal is the release of the software under a free use or open source license, and discussions are underway with the appropriate legal and information technology personnel at the Georgia Institute of Technology.

4.2 Visible spectrum image simulation

4.2.1 Solving the visibility problem

In order to create simulated images of objects, a computer graphics challenge called the *visibility problem* must be overcome. As the name suggests, the visibility problem is the determination of which objects in the scene are visible by the camera and which are not, either because they fall outside of the field of view of the camera or because they are obscured by other objects. The two most popular methods for image rendering are the

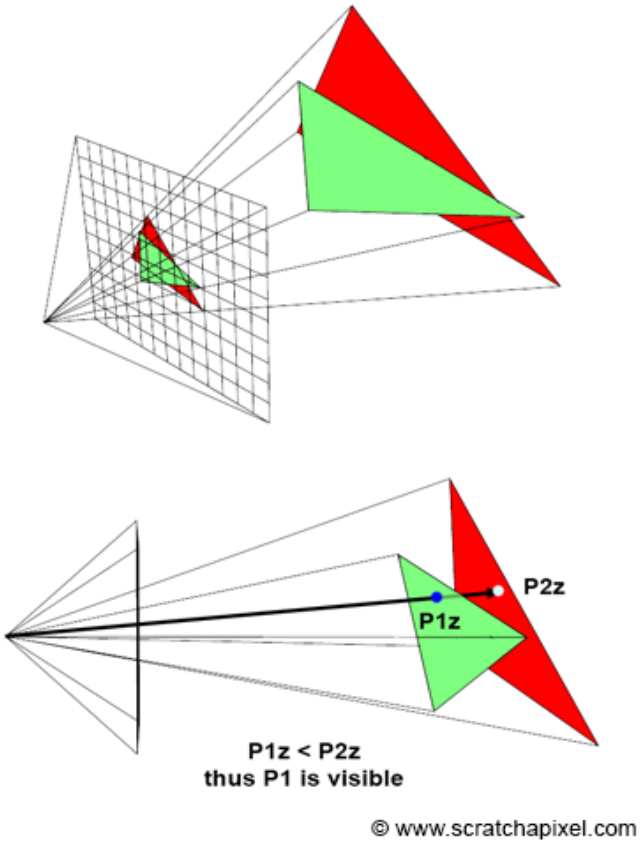

Figure 4.2: Z-buffer process with two differently colored triangles [65]

z-buffer and ray tracing.

4.2.1.1 Z-buffer

The z-buffer works by finding and tracking the depth of each part of the object [64]. It starts with the assumption that no objects are in the scene, and thus each part of the scene has a near-infinite depth from the camera. As each object is rendered, a z-matrix keeps track of the distance from the sensor to each pixel that views it. If a new object is also seen by that pixel, then the renderer compares their depth values and renders the nearer object (Figure 4.2). This process continues until all objects in the image are rendered.

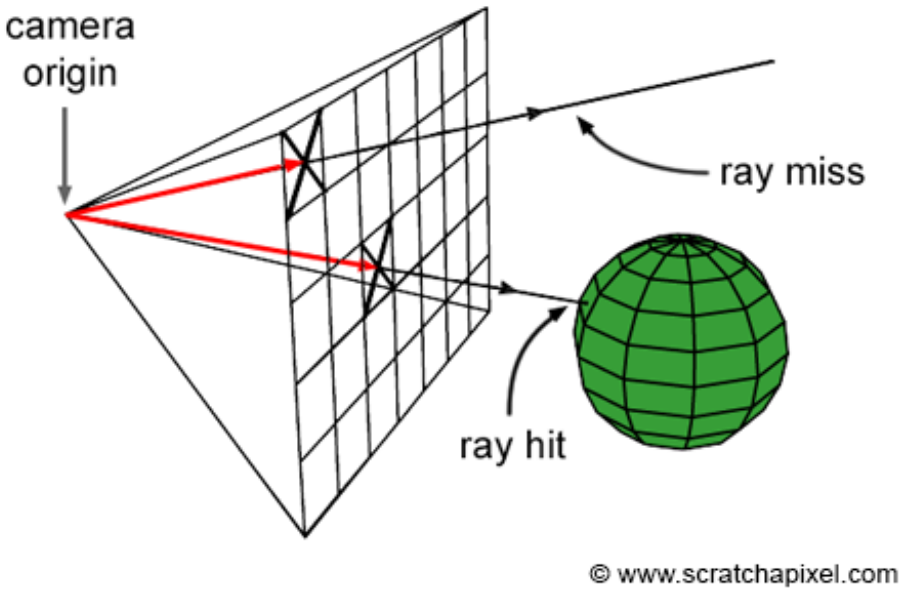

Figure 4.3: Ray-tracing for a basic scene [67]

4.2.1.2 Ray tracing

Ray-tracing works in the opposite direction from z-buffering [66]. Instead of relating the parts of each object to a particular pixel, each pixel is analyzed to see which objects it sees. A large number of rays are generated from the focal point of the camera and directed outward throughout the entire field of view. If the ray encounters an object, that event is recorded and that object is rendered at the pixel associated with that ray. (Figure 4.3)

Depending on how realistic the renderer is, more interactions between the ray and the object may be taken into account. If the object is shiny and subject to internal reflections, the ray tracer records these reflections between the object surfaces. The algorithm generates child rays for translucent objects, accounting for the fact that some light is transmitted and some is reflected.

For this particular application, the objects being imaged are convex and opaque, this additional complexity is not needed. Because ray tracing simulates what each pixel of the camera sees, ray-traced images are generally more realistic at the cost of greater compu-

tational complexity, especially when only one object is being simulated. The z-buffer was initially determined to be the better rendering strategy for this applications since it is more efficient and large numbers of simulated images must be created for training and testing. However, note that in ray tracing each ray is cast and calculated independently, whereas a single z-buffer interacts with every part of the rendered object. Therefore, ray tracing may be parallelized and accelerated with CUDA and images may be rendered more efficiently. Z-buffering does not have this advantage. As is shown in Section 4.5.6.2, CPU rendering is orders of magnitude slower with z-buffering. Because of its ability to be parallelized, ray tracing is the better choice for the SISE.

4.2.2 Triangular mesh

As a part of the ray tracing procedure, the object to be imaged must be discretized into shapes for which a ray-shape intersection can be found. One way to perform this discretization is a triangular mesh. The surface of the target object is covered with a series of vertices which are then connected by edges to form triangular faces. The more dense the vertices on the surface, the closer the resulting mesh will be to the actual image of the object. The most efficient way to create this mesh is not an even distribution of vertices, but rather concentrating them in regions which are more difficult to discretize. Thus, only a few vertices may be needed for a long flat solar panel, whereas a dense mesh will be used for a curved surface like a lens. Figure 4.4 is an example of a triangular mesh of the asteroid Geographos [68].

If this mesh does not already exist for the digital representation of an object, then it must be created. The surface of the object must be represented as a three-dimensional discrete scalar field (more commonly known as "voxels"). Then, a mesh can be found using the marching cubes algorithm. Developed by Lorenson and Cline [69], the algorithm "marches" through the scalar field, analyzing 8 neighboring voxels at a time. Based on the presence or absence of the surface at each vertex of the resulting cube, the polygons

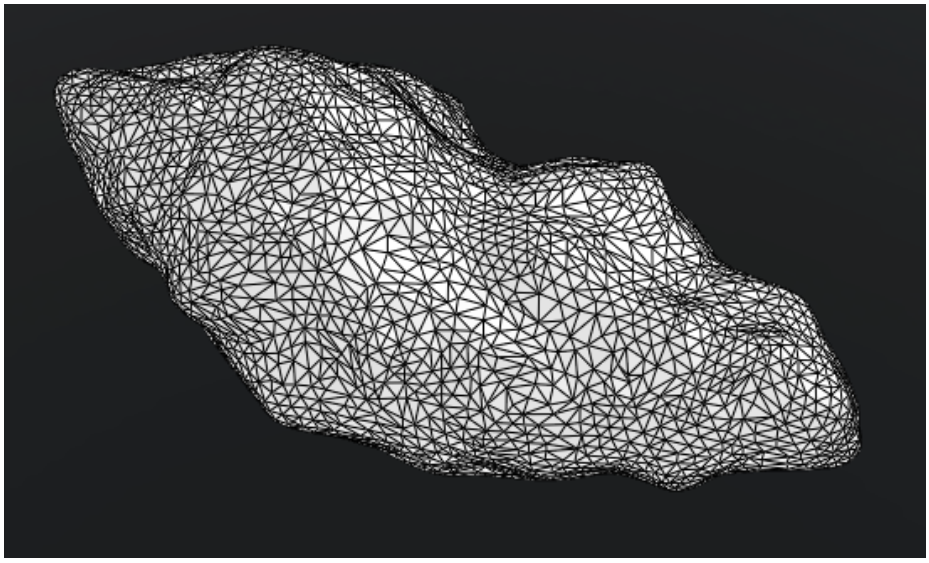

Figure 4.4: Asteroid Geographos triangular mesh [68]

necessary to represent the surface are determined. These polygons are then attached to the existing surface mesh.

Once this mesh has been constructed, the ray-triangle intersection point is calculated for each ray of the camera, if it exists. An algorithm invented by Tomas Möller and Ben Trumbore [70] finds the location of the intersection. The Möller-Trumbore method is particularly fast when compared to other ray-triangle intersection methods because it does not require pre-computation of the plane equation of the rectangle. More details on the implementation of the Möller-Trumbore method are given in Section 4.5.6.3.

4.2.3 Radiation simulation

The radiation simulation equations predict the visible light reflecting from each point on the target object. The radiation is a function of the location, normal direction and optical properties of each vertex and the relative location of the illumination sources in the scene [71, 72]. For the types of radiance that are dependent on the direction to the observer, that vector is taken into account as well. Finally, the temperature of each vertex is updated based on the energy absorbed, emitted, and conducted to neighboring vertices.

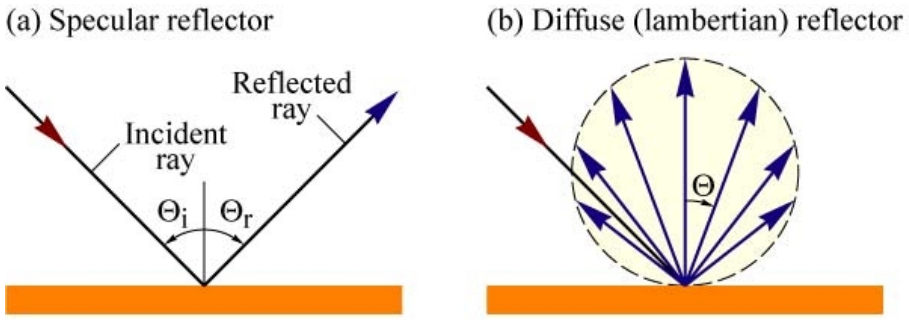

Figure 4.5: Specular (left) versus Lambertian (right) reflection [74]

4.2.3.1 Types of reflectance

Reflected light from an object is broken into three categories: ambient, diffuse, and specular [73]. Ambient reflected light is due to a light source that is sufficiently scattered so that its direction of origin is unknown. An example of this type of reflection is sunlight on a cloudy day. The surroundings are clearly being illuminated, but the clouds scatter the sunlight so that no shadows exist from which to determine the sun's direction. The next type is diffuse, or Lambertian reflection. Diffuse reflection results from a directional light source striking a rough surface. The reflected light is scattered in all directions, with a greater reflected intensity the smaller the angle is between the light source and the normal direction of the surface. An example of such a surface would be unpolished stone. The last type of reflection is specular. Specular reflection results in a "glossy" appearance and follows Snell's law: the angle of incidence equals the angle of reflection. For example, specular reflection results from smooth metal, such as the side of a spacecraft. A sketch of a comparison between Lambertian and specular reflection is given in Figure 4.5. Figure 4.6 shows the same object and lighting subject to the two types of reflectance.

4.2.3.2 Simulated radiance

Equation 4.1 gives a framework for the calculation of spectral radiance from a vertex. It was developed based on equations from Christian [76], but the model presented here was

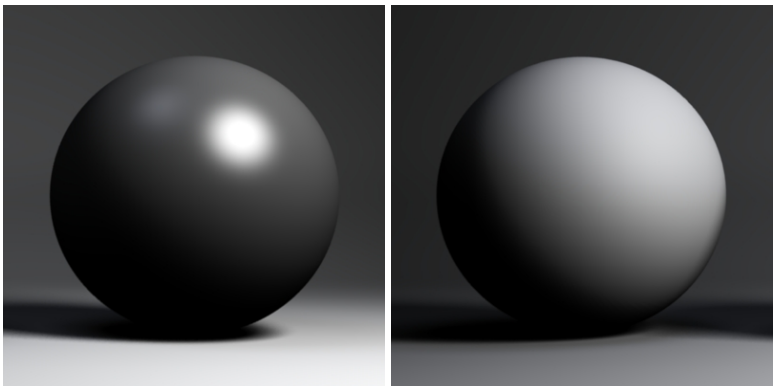

Figure 4.6: Simulated image of a sphere subject to primarily specular (left) and Lambertian (right) reflection [75]

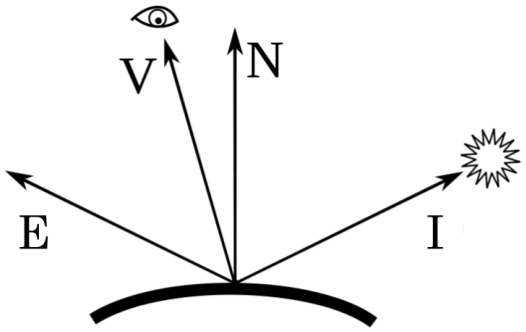

Figure 4.7: Reflection model vectors

created specifically for this research. In this equation, L represents the radiation intensity at wavelength λ, l is an index of the various light sources, and the k coefficients refer to the reflective properties of the material. The letter α is called the specular exponent. It determines the apparent smoothness of the material, with a higher α value resulting from a smoother surface. N, I, and V, are the surface normal vector, direction from the point to the light source, and direction to the observer respectively. E is an intermediate vector calculated from Equation 4.2 (shown in Figure 4.7). In terms of the various types of reflectance, an a subscript is ambient, d is diffuse or Lambertian, and s is specular.

$$L(\lambda) = k_a L_a(\lambda) + \sum_{l=1}^{n} L_l(\lambda)(k_d(N \cdot I) + k_s(E \cdot V)^{\alpha}) \tag{4.1}$$

$$E \cdot V = 2(N \cdot V)(N \cdot I) - (V \cdot I) \tag{4.2}$$

4.2.4 Measurement response

The last required step is to calculate the radiation incident on the sensors and to produce images. The sensor model subroutines traverse the field of view pixel-by-pixel using ray tracing and calculate which objects are seen by the pixel, which of any observed objects is the closest, and how much each object irradiates that pixel. This step is where lens distortion is applied to the image. Then, the subroutines translate that irradiance into a greyscale brightness value based on the internal physics of the sensors. The final result is a simulated image, used for the training portion of appearance matching or a software-in-the-loop simulation. This version of the SISE generates images in greyscale; color simulations are a topic for future work.

Equation 4.3 gives the general framework for irradiance on a pixel, while Equation 4.4 relates that irradiance to a greyscale value in the final image. These two equations were based on prior work from Christian [76]. S here refers to the irradiance on the pixel and θ is the angle between the line-of-sight to the object and the observer surface normal (Figure 4.8). A_{ap}, τ, and f are physical properties of the camera and represent the aperture area, transmittance, and focal length of the sensor. The variable s is the detector response over the integration period T. A_{det} is the area of the detector, F is the fill factor as a fraction from 0 to 1, and R is the spectral responsivity. A more detailed explanation of these terms is provided in [76].

$$S(\lambda) = L(\lambda)\tau(\lambda)\frac{A_{ap}}{f^2}\cos^4\theta \tag{4.3}$$

$$s = \int_T \int_{\lambda_{min}}^{\lambda_{max}} A_{det} S(\lambda) F R(\lambda) d\lambda dt \tag{4.4}$$

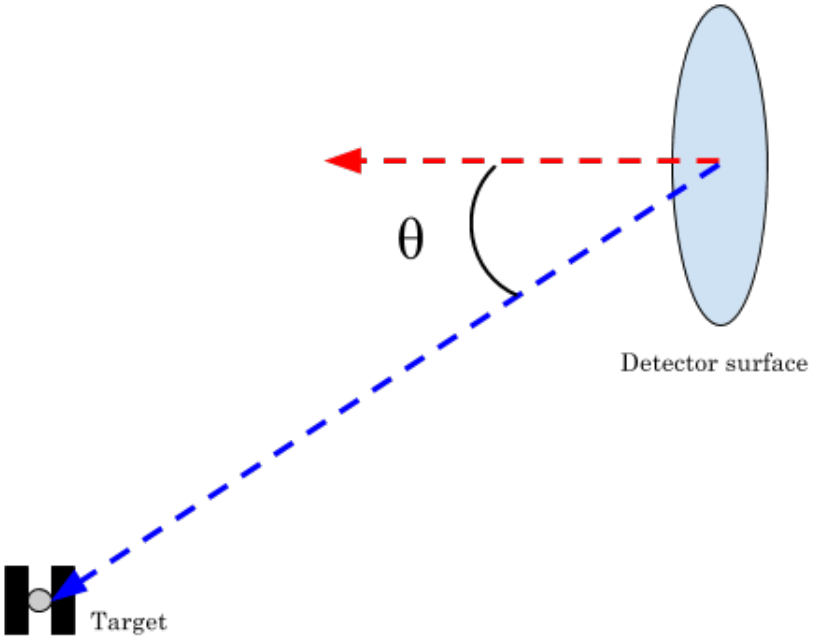

Figure 4.8: Line-of-sight angle (θ) definition for Equation 4.3

4.2.5 Error sources

After the image has been simulated, the other possible error sources may be added in addition to distortion, if needed. One potential error source is pixel blur, either due to defects in the lens or object motion. This effect can be simulated by a convolution function based on the amount of required blur, which is then iterated over each pixel in the image. Shot noise can also be introduced using a point spread function based on a normal distribution. Finally, amplifier and digitization noise can be added. These effects are functions of the dynamic range and the saturation limit, respectively. The relevant equations are described in Section 4.5.5.

4.3 Infrared spectrum image simulation

Many of the same fundamentals of image simulation apply to infrared simulation as they do to the visible spectrum. The visibility problem must be solved in the same way, and reflected infrared light behaves like visible light, albeit with a longer wavelength. The key

difference between the two spectra is the role of temperature. Except for extreme cases when the body is significantly warped due to temperature shifts or is hot enough to begin emitting visible light, the simulation of an object's image in the visible spectrum is the same regardless of its temperature. In contrast, an infrared image is quite dependent on the body's temperature. Therefore, that temperature must be simulated and accounted for in the light being radiated from the object.

4.3.1 Thermal simulation

Thermal modeling is a topic of a large amount of research and extraordinary detail. There are applications is which high-fidelity thermal modeling is vital to success. For the purposes of this research, however, a simple surface model with conduction and radiation is sufficient. The infrared images being created are relatively low quality and will be further compressed during the appearance learning process. Also, the spacecraft being simulated are representative models, so the particular internal heating characteristics are unknown. Therefore, the following simplifying assumptions are made:

1. Only surface thermal properties are modeled.

2. Only heat transfer due to conduction between faces, solar irradiation, and body radiation are considered.

3. Faces are assumed to have uniform albedo, and temperature, thermal conduction properties.

4. The target surface is assumed to be of uniform thickness.

While these assumptions clearly limit the types of effects that are included in the analysis, these limitations are considered to be acceptable for a first demonstration of the algorithm. More detailed models can be included later if necessary to improve the simulation fidelity and accuracy.

4.3.1.1 Conduction

The target models are formed from a convex triangular mesh. Each mesh panel thus has three neighboring faces. Heat is conducted through these three interfaces according to Fourier's Law for heat conduction (Equation 4.5). Q is the heat transferred in Joules, k is the thermal conductive in watts per meter-Kelvin, A is the interface area, d is the surface thickness, ΔT is temperature difference in Kelvin or degrees Celsius, and t is the time of transfer in seconds.

$$Q = \frac{kA\Delta T}{d}t \tag{4.5}$$

Using the uniform thickness assumption, this expression is simplified to Equation 4.6 for the ith side of the triangular face. The vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are the coordinates of the interfacing vertices. The subscript i has been added to Q and ΔT for clarity.

$$Q_i = kt||\boldsymbol{v}_1 - \boldsymbol{v}_2||\Delta T_i \tag{4.6}$$

4.3.1.2 Solar irradiation

The solar irradiance is assumed to be equal to 1367 Watts per meter squared near the Earth's surface. This value is used in Equation 4.7 to determine the heat transfer to the face via solar radiation. Q_s is the heat transfer, S_s is the solar irradiance, ϵ is the material's emissivity, N and I are the same vectors from Equation 4.1, and A_f is the surface area of the face. It also follows that the sun cannot illuminate the rear face of a convex body, which leads to the condition on the dot product between the face normal and the illumination direction.

$$\begin{cases} Q_s = S_s\epsilon(N \cdot I)A_f & N \cdot I > 0 \\ Q_s = 0 & N \cdot I \leq 0 \end{cases} \tag{4.7}$$

4.3.1.3 Body radiation

The radiation of heat from the face into space is found using the model for black body radiation (Equation 4.8). Q_r is the heat transfer, σ is Steffan's constant, A_f is again the surface area of the face, and T is the absolute temperature in Kelvin.

$$Q_r = \sigma \epsilon A_f T^4 \tag{4.8}$$

4.3.1.4 Temperature change

Once the heat transfer for all three neighboring faces of the triangular mesh panel has been found, the change in temperature of the mesh panel ΔT_f is calculated using the specific heat capacity c of the face's material and the various heat transfer values (Equation 4.9).

$$\Delta T_f = \left(Q_s - Q_r + \sum_{i=1}^{3} Q_i \right) / c \tag{4.9}$$

4.3.2 Simulated radiance

The model for the simulated radiance of an object in the infrared spectrum is given by Equation 4.10 and developed from Garnier [22]. This model is similar to Equation 4.1 but includes an additional term. The variable L_t represents the spectral thermal radiance of the body. This term is combined with those for reflectance that are present in the infrared spectrum. L_t is calculated using Equation 4.12, where k_b is the Boltzmann constant, c is the speed of light, h is Plank's constant, and T is the absolute temperature of body. The value b is the albedo of the object, since the preceding equation is for black body radiation. The albedo is a value between 0 and 1 and represents the actual radiation from the body as a fraction of the ideal black body radiation. Albedo values are empirically determined for various materials, so the composition of the object must be known to include the corresponding albedo.

$$L(\lambda) = k_a L_a(\lambda) + L_t(\lambda) + \sum_{l=1}^{n} L_l(\lambda)(k_d(N \cdot I) + k_s(E \cdot V)^{\alpha}) \tag{4.10}$$

$$E \cdot V = 2(N \cdot V)(N \cdot I) - (V \cdot I) \tag{4.11}$$

$$L_t(\lambda) = b\frac{2\pi^4 k_b^4}{15c^2h^3}T^4 \tag{4.12}$$

4.4 Fidelity of simulation

4.4.1 Thermal imaging

As discussed in Section 4.3.1, the inclusion of thermal image simulation into the SISE potentially adds a significant amount of complexity to the program. Visible spectrum images are functions of intrinsic properties of the imaged object: shape, size, material, etc. Thermal images, on the other hand, depend on the temperature of the object. The temperature varies over the imaging time interval as the object heats up as a result of incident radiation or cools off from radiating energy into space. The temperature may also vary within the object itself, since internal systems like computers and batteries generate heat that is seen by infrared cameras. Fully capturing these effects requires an internal model of the target object in addition to the surface model.

It is important, therefore, to determine the scope of the thermal image simulation. Enough fidelity is necessary to ensure that the results obtained using simulated images would be representative for real images. However, too much complexity would turn the focus of the research into thermal modeling instead of visual navigation. Therefore, a rudimentary thermal simulation was implemented instead of a more complex thermal model including internal components of the target. The fusion methodology described in more detail in Section 5.1 and the subsequent results in Section 6.2 confirm this scope as valid for this visual navigation application. Expanding the thermal model to include greater de-

tail is a promising area of future research (Section 7.2.6).

4.4.2 Specular effects

As described in Section 4.2.3.1, specular reflectance results from smooth surfaces and is not spread out the same way that diffuse reflectance is. As a result, specular effects can lead to a concentrated amount of reflected light in a small area, known as a highlight or "glint." Glints lead to other phenomena such as blooming and saturation in pixel detectors which complicates object identification in an image. These glints have an adverse effect on visual navigation systems, since they distort the illumination and even the perceived shape of the object. Glint presents a particular challenge for appearance matching, since the range of orientations in which a glint occurs may be small and not captured by one of the training images. Figure 4.9 shows sunlight glinting from the solar panel of the Hubble Space Telescope.

There are a few ways to simulate the effects of glints in order to test the robustness of the algorithm. The first is to simply increase the specular coefficient k_s and the specular exponent α. This has the effect of creating brighter, more concentrated highlights. For more complex specular surfaces, a Gaussian operation like the one described in Yan, et al. [78] is required. Similar to the thermal imaging simulation, the simple solution of increasing the specular exponent was deemed to have enough fidelity for the purposes of this dissertation. In this research, detector bloom and saturation are not being modeled, as these are hardware-specific effects and modern sensors are able to mitigate some of these conditions (Theuwissen [79]).

4.4.3 Background

The inclusion of the *add_background* function (Section 4.5.5.3) into the SISE allows the simulation of a non-black background, both for appearance matching training and for software-in-the-loop tests. So long as the background is defined either as a brightness

Figure 4.9: Sunlight glint off the Hubble Space Telescope [77]

function of the image location or via an image file, that background can be included in the test images. Examples of simulated images with background inclusion can be seen in Figures 3.8 and 6.11.

4.5 Software implementation

The program which generates simulated images for either appearance training or testing is known as the spacecraft imaging simulation environment (SISE). The purpose of the SISE is to generate a series of simulated images of a target spacecraft as viewed by a sensing spacecraft. The program is written in Python based on a preliminary MATLAB version developed for prior research (McBryde and Lightsey [25]).

Figure 4.10 outlines the functional flow of the SISE. Within the initialization step is are the *cam* and *load_scenario* functions which create the **Camera** object and load the scene parameters, respectively. The next three steps are completed for each image that is simulated. Target modeling takes place within the *load_mesh* function, which reads in the mesh file and applies the appropriate transformations. Radiation simulation is performed by two different functions. The *vis_raytrace* subroutine simulates visible light, while *ir_raytrace* simulates infrared radiation. The final step is file creation and refinement. The functions *vis_err* and *ir_err* apply errors in the appropriate spectrum, while *add_background* replaces the black background of the image with a different pattern, if desired.

With the exception of the target modeling functions listed in Section 4.5.3, all programs and subroutines that comprise the SISE were developed specifically for this research.

4.5.1 Function *sise*

The function *sise* is the top-level function for generating simulated images. It is called by either the appearance training routine or the software-in-the-loop test. That program could be training image generation or software-in-the-loop testing, for example. The *sise* function takes as input the location and orientation of the sensor and the target in inertial space. It

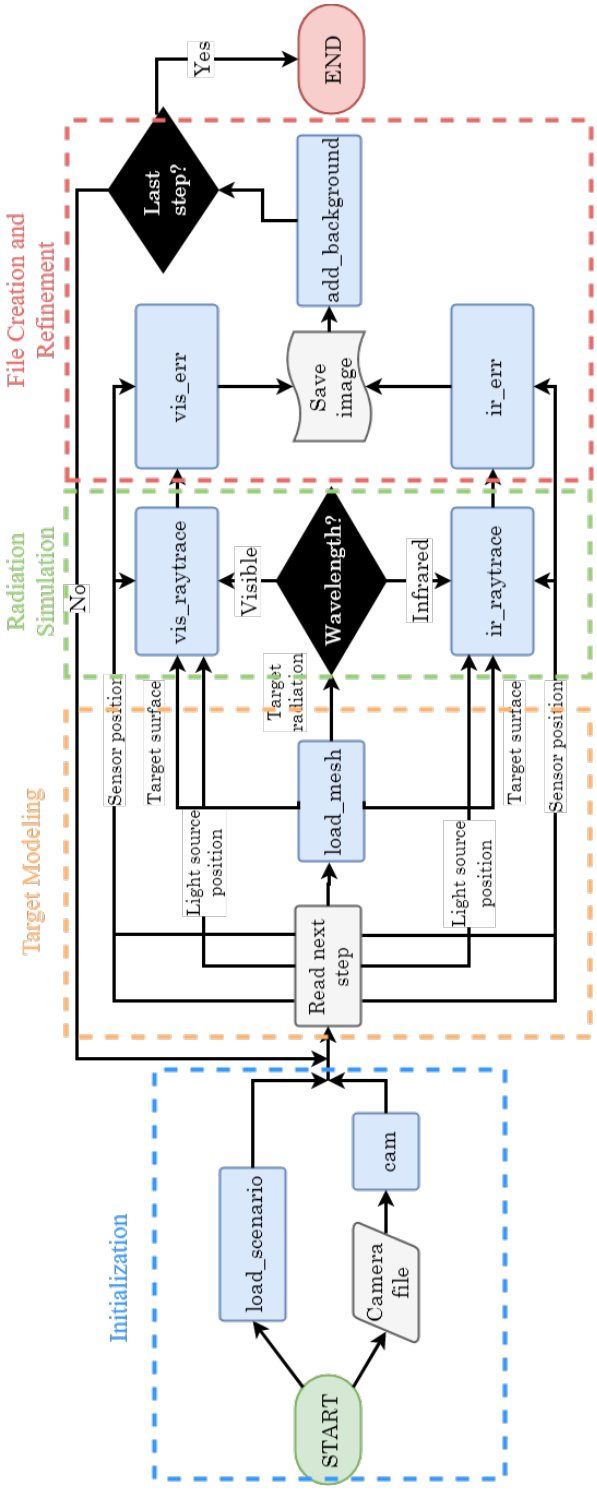

Figure 4.10: SISE function flowchart

also requires the position of the lighting source or sources in the scene. Additionally, *sise* needs some image generation parameters: the file name of the CAD model for the target and whether or not a background is being added to the image. The purpose of these two parameters is addressed in Sections 4.5.3 and 4.5.5.3, respectively. Lastly, the main function requires a **Camera** object. This object is generated from the *cam* function and is addressed in greater detail in Section 4.5.2.1. Given these inputs, the end result is a matrix of light intensity values which can be saved into an image file, if desired.

The design of the *sise* function is fairly straightforward. A matrix is initialized of the appropriate size based on parameters from the **Camera** object. Functions from the *trimesh* library are used to initialize a **Mesh** object which is then translated or rotated if necessary to the given target position and attitude. This mesh, along with the camera position and attributes, is passed to the appropriate ray-tracing function. Finally, the background of the image is either replaced with the desired background or left black.

4.5.2 Initialization

4.5.2.1 Function cam

The *cam* function is responsible for initializing parameters for the camera, for the sensor, and for designating any errors that are applied. These values are given in a configuration file that is called by *cam* to create the **Camera** object. That object also contains the functions for training an appearance matching library. This encapsulation of the sensor's characteristics and appearance training parameters is part of the multi-sensor framework, the details of which are addressed in Section 5.2. The following parameters are initialized when *cam* calls a camera configuration file:

- Camera
  - Image size, also called resolution (pixels)
  - Focal length (millimeters)

  - Aperture (f-stop)
  - Transmittance (percentage)
  - Dynamic range (decibels)
- Sensor
  - Spectrum
  - Pixel size (micrometers)
  - Quantum efficiency (percentage)
  - Saturation limit (photons)
  - Fill area (percentage)
  - Dark fraction (percentage, visible) or wavelength (nanometers, infrared)
- Error
  - Radial distortion (coefficients, 1-5)
  - Blur (pixels)
  - Shot noise (number of electrons)

These parameters are stored in the **Camera** object and are used to generate the simulated images.

4.5.3 Target modeling

Unlike the other aspects of the SISE, the triangular mesh handling was performed using a freely available library called *trimesh* developed by Dawson-Haggerty [80]. According to the author, the package "is a Python library for loading and using triangular meshes." This research mainly utilized three functions from this library. The function *load_mesh* creates the **Mesh** object from a CAD file. This object contains information about the geometry

of the mesh: the location of the vertices in space, the vertices which make up each face, and the normal direction of each face. The pre-computation of this last value significantly speeds up ray tracing later in the program. The other two functions are part of the **Mesh** object: *apply_translation* and *apply_transform*. The first function translates the vertex locations in space and the second rotates the vertex locations and normal directions of each triangular face using a direction cosine matrix.

Note that an earlier version of the SISE utilized the library's built-in ray tracing function as well. It was later replaced by a faster version using CUDA acceleration, addressed in greater detail in section 4.5.6.

4.5.4 Radiation simulation

The *ray_trace* function is the last major component in the image rendering process. It takes the rotated and translated mesh as well as the camera parameters and returns a light intensity matrix of the mesh on a black background. There are two versions of the *ray_trace* function, one each for the visible and infrared spectra. The main purpose of the function is to set up the CUDA acceleration which performs the actual ray casting and tracing. Several arrays are initialized on the GPU to contain the aforementioned data after which a kernel function is called to cast a ray from each pixel. Further details on that procedure are given in Section 4.5.6.

4.5.5 File creation and refinement

4.5.5.1 Distortion

Distortion is the first error source to be implemented. Unlike the subsequent error types described in this section, the calculation and application of distortion takes place in the *kernel* function, the particulars of which are discussed in Section 4.5.6.2.

Distortion is defined as the failure of a lens to preserve straight lines in an image. Distortion comes in two main types: radial distortion, which is radially symmetric, and

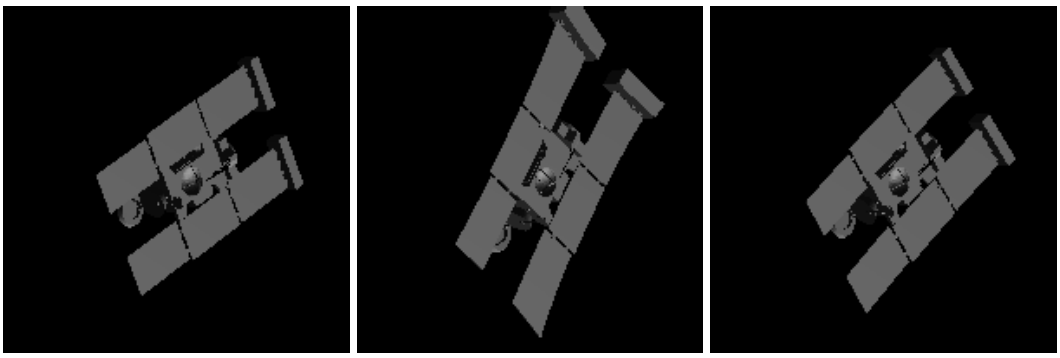

Figure 4.11: Undistorted image (left), radially distorted image (center), tangentially distorted (right)

tangential distortion, which is not. Since this error is caused by the nonuniform bending of light that passes through the lens, the part of the object seen by each ray is different than what would be seen by an undistorted lens. Brown [81] models distortion using five coefficients, three for radial distortion and two for tangential distortion (Equation 4.13). The values x_n and y_n are the first two coordinates of the normalized image location, k_i are the various coefficients, r_n is radial image coordinate defined as $r_n = \sqrt{x_n^2 + y_n^2}$, and x_d and y_d are the resulting distorted image coordinates. Figure 4.11 shows a simulated image of the Stardust spacecraft which is transformed by each type of distortion.

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = (1 + k_1 r_n^2 + k_4 r_n^4 + k_5 r_n^6) \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} 2k_3 x_n y_n + k_4(r_n^2 + 2x_n^2) \\ k_3(r_n^2 + 2y_n^2) + 2k_4 x_n y_n \end{bmatrix} \quad (4.13)$$

4.5.5.2 Functions vis_err and ir_err

In order to improve the fidelity of a software-in-the-loop test and to verify the robustness of an algorithm, the functions *vis_err* and *ir_err* can be used to model error effects in visible and infrared images, respectively. Two categories of error are modeled by these functions: blur and noise.

Blur in an image is accomplished by convolving the light intensity matrix with a Gaus-

sian function (Equation 4.14, Stockman and Shapiro [82]). The standard deviation σ tunes how much blur is applied to the image and x and y are pixel coordinates.

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.14}$$

Noise can be added from a number of different sources, as described by Kodak [83]. Shot noise results from thermally generated electrons in the photoelectric surface of a visible spectrum sensor. Amplifier noise applies to both spectra and is a byproduct of amplifying the photoelectric signal coming from the sensor.

The pixel by pixel noise is given by Equation 4.15. The variable g is a normal random variable, f_d is the fraction of the sensor area that is not exposed to light, l_{sat} is the saturation limit of the pixel in photons, and r_d is the dynamic range in decibels.

$$e_{pix} = g\left(f_d l_{sat} + \left(\frac{l_{sat}}{r_d/20}\right)^2\right) \tag{4.15}$$

Finally, digitization noise is inherent in both spectra due to the conversion of the analog electrical signal into a digital form. It is modeled by dividing the light intensity matrix by the number of photons per intensity bit a, taking the floor, and then multiplying by the photons per bit again (Equation 4.16).

$$I = a\left\lfloor \frac{I}{a} \right\rfloor \tag{4.16}$$

4.5.5.3 Function add_background

The *add_background* function is an auxiliary function that was added to the SISE to aid in the background randomization procedure (Section 3.4.2). As part of the *ray_trace* function, rays that do not encounter the mesh are assigned a "brightness" of -1. Thus, the *add_background* function identifies these pixels for replacement with the desired background. Any value is possible between 0 and 255, but the options utilized in this research

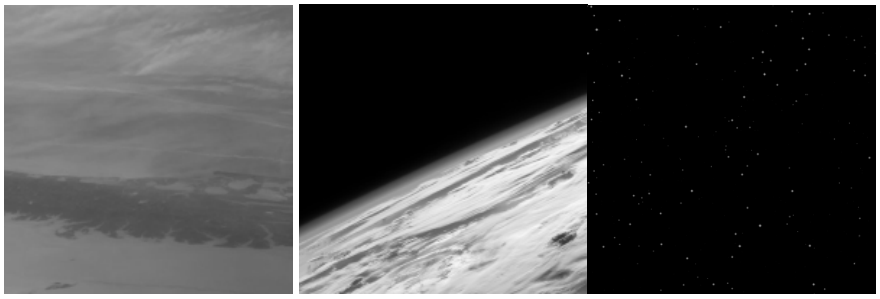

Figure 4.12: Cloud (left), horizon (center) and star (right) background image files

are all zero (black), random, star field, cloud, and horizon. Note that the random values are determined from a built-in Python function for generating random numbers while the cloud, horizon, and star backgrounds were copied from an appropriate image file (See Figure 4.12). Examples of the image with cloud, horizon and star backgrounds of these are given in Section 6.1. The *add_background* function searches the light intensity matrix for negative values and replaces them, returning the modified matrix as an output.

4.5.6 GPU acceleration

Utilizing the multiple processing cores on a graphics processing unit vastly speeds up the repetitive execution of a simple function. Applying GPU acceleration to the SISE using CUDA improves the image simulation performance by orders of magnitude.

4.5.6.1 CUDA Overview

CUDA is designed to interface with C, C++, and Fortran by default, but a Python package known as Numba enables CUDA programs to be written and executed in Python [84]. The *ray_trace* functions described in Section 4.5.4 load the Numba module, which in turn allows the creation of the necessary variables and functions on the GPU. Every value or array called as an input to *ray_trace* needs to be copied to the GPU so it can be seen by the kernel. An additional array for the light intensity matrix output is created on the GPU as well. Once these are set up, the kernel function is executed with two additional parameters: number of grids and number of threads. These parameters are responsible for telling the

GPU how many concurrent processes need to be run. For ray tracing, selection of these parameters is straightforward: one thread per ray, executed on a 2D array of cores of the same size as the resulting image.

4.5.6.2 Function kernel

The compute kernel is the function that is executed on each one of the cores. It takes as input the triangular mesh parameters, the size of the image and the pixels on the sensor, the focal length of the camera, the sensor position and attitude, and the location of the lighting source. Based on a management function, the kernel is assigned a particular pixel which in turn allows it to calculate the unit vector for that ray. Then, a ray-triangle intersection is calculated for each face of the object. A *kernel* function was developed specifically for this research to simulate images as part of the SISE.

If a hit is detected, the distance to that hit is saved in case a face with a closer intersection exists, in which case that face is replaced. If at least one face of the object is intersected, the incident radiation on the sensor is calculated using one of the functions from Section 4.5.6.4. If not, the pixel is assigned a -1 brightness value to indicate it is a background pixel. This process is run in parallel for each pixel in the camera, greatly increasing the image rendering efficiency.

To show this, a series of images were rendered using the CUDA interface and with only CPU. 5 images were rendered of the same object at 10 different orientations. The only difference between the algorithms was that the CPU version used optimized linear algebra operations in place of the hand-coded versions required by CUDA (Section 4.5.6.5). The results are given in Table 4.1.

4.5.6.3 Function hit

The *hit* function is on-board the GPU and calculates the intersection between the cast ray and a triangular face of the object. It utilizes the intersection algorithm developed by Möller

Table 4.1: Rendering time for an image with GPU vs CPU

Image size	# of pixels	Avg. CPU rendering time (s)	Avg. CUDA rendering time (s)
40x32	1280	126.91	0.0483
80x64	5120	493.907	0.0848
160x128	20480	2068.52	0.249
320x256	81920	8249.39	0.982

and Trumbore, the details of which are found in [70]. This process determines whether the intersection exists and if so, what the distance is to the intersected point. The algorithm works by taking the origin and direction of the ray and the vertices of the face and then running a series of checks to determine if the ray, in fact, hits the triangle.

First, a determinant is calculated to make sure that the ray does not lie in the same plane as the triangle, precluding an intersection. If the ray intersects the plane, the barycentric coordinates u and v are calculated. Barycentric coordinates, or areal coordinates, define the location of a point relative to the vertices of a triangle. They are subject to Equations 4.17-4.19. N is a point in Cartesian coordinates; A, B, and C are the vertices of the triangle; and u, v, and w are the barycentric coordinates.

$$N = uA + vB + wC \tag{4.17}$$

$$1 = u + v + w \tag{4.18}$$

$$0 \leq u, v, w \leq 1 \tag{4.19}$$

Because of Equation 4.18, only two of the coordinates need to be calculated in order to find the third, which in practice is usually w. This relationship leads to an additional restriction on u and v: $u + v \leq 1$. These coordinates are used to make sure that the intersection lies within the bounds of the triangular face, which means that the barycentric coordinates satisfy the derived restriction as well as Equation 4.19.

Finally, the algorithm needs to determine the presence of a ray intersection, not just a

line intersection. This final check is made to ensure that the intersection point lies along the correct direction from the origin. If all of these checks are satisfied, a distance from the origin to the intersection is returned. If at least one fails, a predetermined large distance is returned, indicating that no intersection exists. Note that this algorithm does not distinguish between an intersection on the "front" or "back" of a face. That check is made later by the irradiance functions.

4.5.6.4 Functions light and therm

The two functions *light* and *therm* are copied to the GPU and calculate the irradiance from a particular face onto the pixel from which the ray was cast. For the visual *ray_trace* function, *light* uses Equations 4.1 through 4.4, to find the sensor response of that ray in the visible spectrum. Similarly, *therm* uses Equations 4.10 through 4.12 to calculate the sensor response in the infrared range. These functions are called only once per ray that is incident on the object.

4.5.6.5 Helper functions

Functions on the GPU have a limited instruction set that does not include any functions that take arrays as inputs. In order to execute the preceding routines, a series of helper functions were created and transferred to the GPU:

- Array cross product - $\mathbf{z} = \mathbf{x} \times \mathbf{y}$
- Array dot product - $z = \mathbf{x} \cdot \mathbf{y}$
- Array subtraction (also used for array addition) - $\mathbf{z} = \mathbf{x} - \mathbf{y}$
- Array division by a scalar (also used for array multiplication by a scalar) - $\mathbf{z} = \mathbf{x}/y$
- Array Euclidean norm - $c = \sqrt{\mathbf{x} \cdot \mathbf{x}}$

CHAPTER 5
SENSOR FUSION AND FILTERING

5.1 Sensor fusion strategies

As described in Section 2.1.3, there are numerous strategies for visible and infrared sensor fusion. Based on the needs and limitations of appearance matching, a two-step approach is proposed for combining multiple visible and infrared cameras. First, a co-located infrared camera is used to observe the target in the visible image and crop the excess background, if it exists, from both the infrared and visible camera test images. Then one of two options is chosen for the actual data fusion. One is to combine the images into a hybrid image vector and run the PCA on those images. The second is to perform the PCA on the visible image set and the infrared image set separately and then reconcile the results from the two spectra.

5.1.1 Target identification and image cropping

While the background randomization technique allows the target images to be located on an arbitrary background, the appearance matching implementation given in this research requires that the size of the object within the image frame remains consistent from the training library to the test image. Thus, the target must be located within the image so that the "excess" image can be removed and the remaining part of the image scaled to the appropriate size.

5.1.1.1 Optical flow

Multiple techniques exist to perform this kind of object extraction. One method is optical flow segmentation. Optical flow refers to the movement of pixels from image to image, usually between frames of a video. If certain parts of an image move in a similar way,

an algorithm can be trained to separate those parts into segments based on that motion. One common method within the field of computer vision is the Lucas-Kanade method [85]. This method assumes that within the neighborhood of a point under consideration, the displacement of the contents of the image are small and constant from frame to frame. Thus, each pixel in a window surrounding the point must satisfy a system of equations (Equations 5.1 to 5.3) where (V_x, V_y) is the local velocity vector; q_i are the pixels within the window; and D_x, D_y, and D_t are the partial derivatives of the image with respect to the image location (x, y) and the time t. This overdetermined system is solved using the least squares principle and the image is segmented into regions with a similar local velocity vector.

$$D_x(q_1)V_x + D_y(q_1)V_y = -D_t(q_1) \tag{5.1}$$

$$D_x(q_2)V_x + D_y(q_2)V_y = -D_t(q_2) \tag{5.2}$$

$$\vdots$$

$$D_x(q_n)V_x + D_y(q_n)V_y = -D_t(q_n) \tag{5.3}$$

The partial derivatives are solved for numerically, using the central difference formula for the spatial derivatives and backward difference for the temporal derivative. Equations 5.4 to 5.6 give the expressions for these derivatives, with t being the current frame, $t-1$ being the previous frame, and Δt the time between frames.

$$D_x(q_i) = \frac{\partial q_i}{\partial x} = \frac{q_i(x+1, y, t) - q_i(x-1, y, t)}{2} \tag{5.4}$$

$$D_y(q_i) = \frac{\partial q}{\partial y} = \frac{q_i(x, y+1, t) - q_i(x, y-1, t)}{2} \tag{5.5}$$

$$D_t(q_i) = \frac{\partial q}{\partial t} = \frac{q_i(x, y, t) - q_i(x, y, t-1)}{\Delta t} \tag{5.6}$$

Typically these segments are going to be foreground objects that are moving relatively

Figure 5.1: Dense optical flow using the Lucas-Kanade method [86]

rapidly and background objects which are moving slowly or not moving at all. Thus, optical flow is often used to separate foreground from background. Figure 5.1 shows an example from the software library OpenCV, in which pedestrians in a video are separated from the roadway on which they are walking.

The obvious drawback to using optical flow for segmentation is that multiple images are needed before optical flow can be established. Additionally it requires the foreground and the background to be moving at a different enough rate such that they can be separated. Neither of these conditions can be guaranteed for a satellite tracking a target. Figure 5.2 shows the optical flow segmentation based on video of the International Space Station jet-

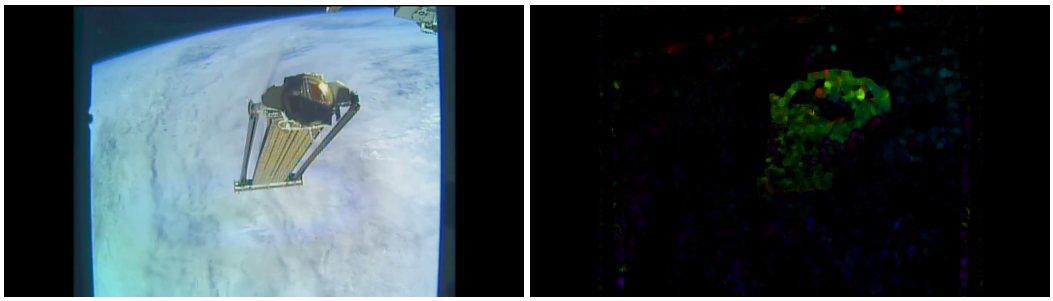

Figure 5.2: Dense optical flow segmentation (right) of ROSA jettison (left)

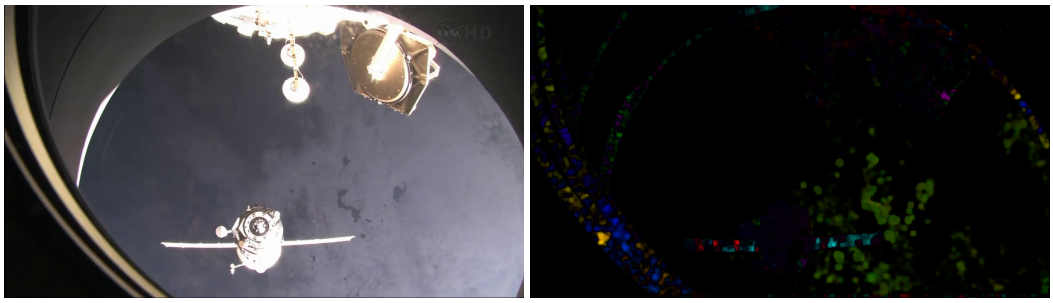

Figure 5.3: Dense optical flow segmentation (right) of Dragon docking (left)

tisoning the Roll-Out Solar Array (ROSA) [87]. Conditions for optical flow segmentation are favorable and the right hand image clearly shows the array. Figure 5.3 is video of the Dragon module docking with the ISS [88]. The cloud motion is greater in the background here and module does not show a consistent enough optical flow to be segmented. Both of these examples were generated for this research using the OpenCV dense optical flow library.

5.1.1.2 Iterative processing

Background randomization has already been shown to be effective at identifying an object on a non-black background. One possibility, therefore, is to process portions of the image in an iterative fashion. Assuming that an object exists within the image frame and that enough sub-images are processed, one will return a high-confidence object identity and thus locate the object within the image. Since the scale of the object within the image is unknown as well, this process would have to be repeated for multiple window sizes. A

Figure 5.4: Test objects on a cluttered background [59]

similar procedure was performed using robust PCA in Leonardis and Bischof [59] on the image shown in Figure 5.4.

This procedure, too, has an obvious drawback: computation time. Multiple PCA matches have to be performed for each image. Even assuming the process stops once a high-confidence match is found, the computation effort is untenable, especially for an embedded system.

5.1.1.3 Infrared masking

Given that an infrared signal is already being used later in the process, the simplest and most logical choice is to use the information from the infrared camera. A good assumption given the space environment is that the target will have a significantly different temperature from the background. Even if the background is the Earth, the average radiating temperature of the Earth's surface is 288 K or 15 degrees Celsius [89]. An operable satellite will have a temperature greater than this due to waste heat from external electronics. An inoperable target will heat and cool from the presence or absence of solar radiation. This temperature will differ from the Earth's, which is regulated by atmospheric processes.

A threshold is applied to the infrared image to separate the foreground of the target from the background of the image. The location and extent of the target within the image frame may then be determined. These parameters are represented using two 2D pixel locations:

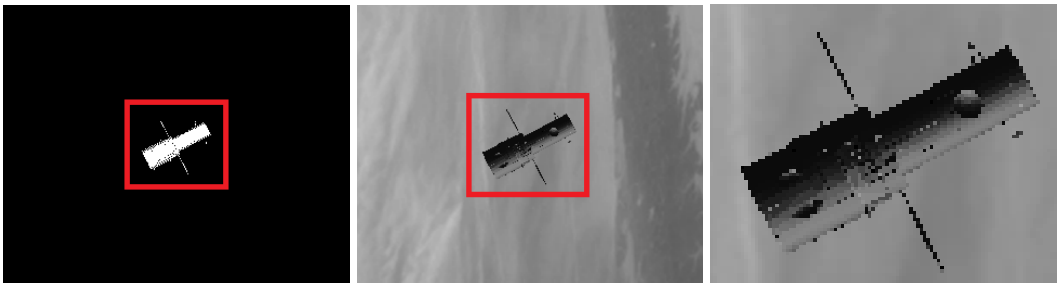

Figure 5.5: Example of infrared masking with HST on a cloud background

the top-right of the image and the bottom-left. The infrared image is then cropped to remove as much as the background as possible. Finally, the resulting intensity array is scaled to the same size as the training library. A example of this process is given in Figure 5.5.

For the images from every other sensor, the object is located using the extent information from the first infrared image. First, unit vectors $\mathbf{u}^{TR}$ and $\mathbf{u}^{BL}$ to the object extent points are found (Equations 5.7 and 5.8). Let (μ_1^R, ν_1^T) and (μ_1^L, ν_1^B) be the pixel locations of the target extent in the infrared image. The values w_1 and h_1 are the width and height of the infrared sensor, in pixels, p_1 is the infrared pixel size, and f_1 is the infrared camera focal length. Typically these final two values will have units of micrometers and millimeters, respectively.

Equations 5.9 to 5.12 locate the target extent (μ_2^R, ν_2^T) and (μ_2^L, ν_2^B) on the second sensor. The values w_2 and h_2 are the width and height of the second sensor in pixels, p_2 is the second sensor pixel size, and f_2 is the second camera focal length. The $\lceil \rfloor$ operator rounds toward the nearest integer, with half-integers rounding down. Note that an assumption is made in this case that the two cameras are close enough together that the location of the object does not vary significantly within the two image frames. If the two sensors are far apart, a correction factor would need to be applied to $\mathbf{u}^{TR}$ and $\mathbf{u}^{BL}$ to account for the discrepancy.

$$\mathbf{u}^{TR} = \begin{bmatrix} \left(\mu_1^R - \frac{w_1}{2}\right) p_1 \\ \left(\nu_1^T - \frac{h_1}{2}\right) p_1 \\ f_1 \end{bmatrix} \tag{5.7}$$

$$\mathbf{u}^{BL} = \begin{bmatrix} \left(\mu_1^L - \frac{w_1}{2}\right) p_1 \\ \left(\nu_1^B - \frac{h_1}{2}\right) p_1 \\ f_1 \end{bmatrix} \tag{5.8}$$

$$\mu_2^R = \left\lceil \frac{u_x^{TR}}{u_z^{TR}} \frac{f_2}{p_2} \right\rfloor + \frac{w_2}{2} \tag{5.9}$$

$$\mu_2^L = \left\lceil \frac{u_x^{BL}}{u_z^{BL}} \frac{f_2}{p_2} \right\rfloor + \frac{w_2}{2} \tag{5.10}$$

$$\nu_2^T = \left\lceil \frac{u_y^{TR}}{u_z^{TR}} \frac{f_2}{p_2} \right\rfloor + \frac{h_2}{2} \tag{5.11}$$

$$\nu_2^L = \left\lceil \frac{u_y^{BL}}{u_z^{BL}} \frac{f_2}{p_2} \right\rfloor + \frac{h_2}{2} \tag{5.12}$$

5.1.1.4 Relative position

Using the infrared image to crop and scale the target serves an additional purpose besides transforming the test image into a usable form by the PCA algorithm. Based on the pixel locations of the extent, the location of the centroid of the target within the image frame, along with the amount the image must be scaled, can be calculated. This same information is available for the training image library, though in that case the cropping and scaling will be minimal if the library is well constructed.

By comparing the image extent in a test image to its corresponding image in the image library, relative position to the target is found along with the relative orientation determined using PCA. The location of the centroid in the image gives the unit vector $\hat{\mathbf{r}}$ (Equation

5.13) to the object while the scaling factor σ, along with the distance to the object from the training library d_{train}, gives scalar range. These two parameters combine to give a relative position solution $\mathbf{r}$ (Equation 5.14). That solution cannot be found until after the object identification step of appearance matching, since the geometry of the object is also required.

$$\hat{\mathbf{r}} = \frac{\mathbf{u}^{TR} + \mathbf{u}^{BL}}{2} \tag{5.13}$$

$$\mathbf{r} = \sigma d_{train} \hat{\mathbf{r}} \tag{5.14}$$

5.1.2 Hybrid PCA

The first method for fused PCA is performed using a hybrid image. During single-spectrum PCA training, an image is rasterized into a vector. That vector is then normalized to a total brightness of 1 and then combined with the rest of the image set of different objects, illuminations, and orientations to form the matrix $\mathbf{X}$. In order to do hybrid image PCA training, images at a single orientation from different sensors are vectorized and normalized separately (Figure 5.6), and then concatenated in order to form a hybrid image vector (Figure 5.7). That vector is then combined as before to form a new, longer vector $\mathbf{X}_h$. The same PCA process is then followed as before to create eigenvectors and surfaces for the universal as well as the object image sets.

The advantage to fusing the signals in this way is that the exact same framework can be used as before to identify the object and find the closest matching orientation. There are a couple of drawbacks, though. First, since the image is now larger, more eigenvalues will have to be used in order to get the same amount of variance. In addition, this method relies on the exact same alignment between the various cameras during training as for the test images. Even a slight misalignment could render the hybrid PCA process inconclusive. Finally, if one or more sensor becomes unavailable, separate PCA parameters would be

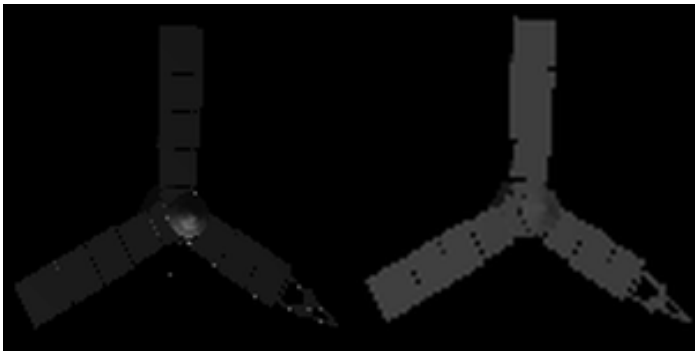

Figure 5.6: Visible (left) and infrared (right) spectrum cropped images

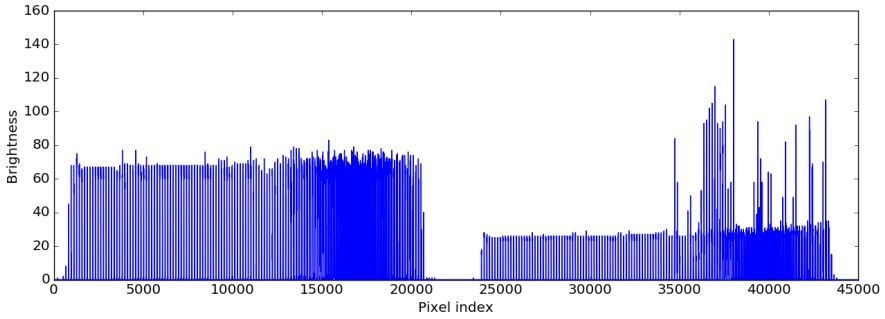

Figure 5.7: Hybrid image vector

needed for each possible configuration of sensors, including each sensor on its own. It is apparent that such an arrangement would have a high memory and computation cost for anything beyond just two sensors.

5.1.3 Reconciled PCA

The second approach to PCA sensor fusion is to reconcile the results of the various sensors. In this method, the image sets are kept separate and the PCA process is performed on each of them in turn. This results in as many universal image sets as sensors, and as many object image sets as the product of the size of the object library and the number of sensors. When a set of test images is processed, the images from each sensor are matched with its particular group of eigenspaces. The result is a series of object identities and attitudes and their corresponding confidence values. These confidence values are defined for a test image in Section 3.3.2.3. Those values are then usually reconciled into a single result before being

passed along to whatever process is performing the visual navigation. The exception to this approach is the case in which a multiple hypothesis filter is implemented downstream. That possibility is discussed in greater detail in Section 7.2.1. For the purposes of this dissertation, it is assumed that a single result is desired.

The advantage to this fusion strategy is that the contribution of each sensor is known and is balanced according to the user. The downside is that a new, more complicated, fusion-inclusive subroutine has to be written to combine the results from each sensor. There is also an increased computational burden on-board the spacecraft, since the projection and nearest neighbor search must be performed for every sensor instead of just once.

5.2 Multi-sensor framework

In order for this system to be multi-spectral instead of just dual-spectral, the framework must be able to accommodate more than two sensors, multiple sensors of the same type, even sensors that are not visible or infrared. In order to accomplish this, the **Camera** object described in Section 4.5.2.1 additionally contains the appearance matching parameters associated with each sensor.

Nothing about the training or learning procedure must be modified for reconciled PCA. The PCA procedure is simply run once for each sensor and a list of **Camera** objects is carried through the process instead of a single object, one for each sensor. The modification comes in the recognition step. Appearance recognition is performed for each sensor, resulting in an object identity with accompanying confidence value and an attitude with accompanying confidence as well. The images together are used with infrared masking (Section 5.1.1.4) to find the relative position and complete the pose estimate. If these values differ among the sensors, then a single answer has to be found from the different values, if one can be found. This reconciliation procedure is described in the following sections.

It should be noted that in some ways, the single-spectrum PCA already does a form of reconciliation in calculating confidence values. Each of the nearest neighbors in the

object identification and attitude determination step is analogous to reading from a different sensor with a confidence determined by its distance from the test point. Relative position determination is dependent on the object's identity and is considered to be part of the object identification step. Therefore, reconciling results from different sensors can be done in much the same way as for different nearest neighbors. One key difference is that with different sensors there is the possibility of two or more high confidence results. Reconciled PCA must be able to handle this possibility in an logical manner.

5.2.1 Object identification

Object identification is the result of a multi-step process. The first step is reconciling the object's identity among the various sensors. This step is performed using the algorithm specified in Section 3.3.2.1. Object identities are returned from each sensor along with their associated confidence values.

The first parameter to examine is the confidence value associated with each sensor's object identification. If that value for one of the sensors is much higher than the others, then that identity is selected as the best guess. Similarly if more than one sensor agrees on the object's identity with high confidence, then the attitude determination proceeds with reconciling the attitudes of those sensors.

The worst-case scenario is that multiple sensors disagree with similar confidence values. It does not matter if they have high or low confidence in their disagreement. One final piece of information may be used to resolve this disagreement. If the history of the object identity is available and agrees with one of the sensors, then that can be used to "break the tie." However, if that information does not exist, then there is no way to reconcile the two object identities within the current framework. The addition of multiple-model filtering or a similar approach could keep both possibilities active until one collapses. More detail on this potential addition is provided in Section 7.2.1. Figure 5.8 outlines the object identification fusion procedure.

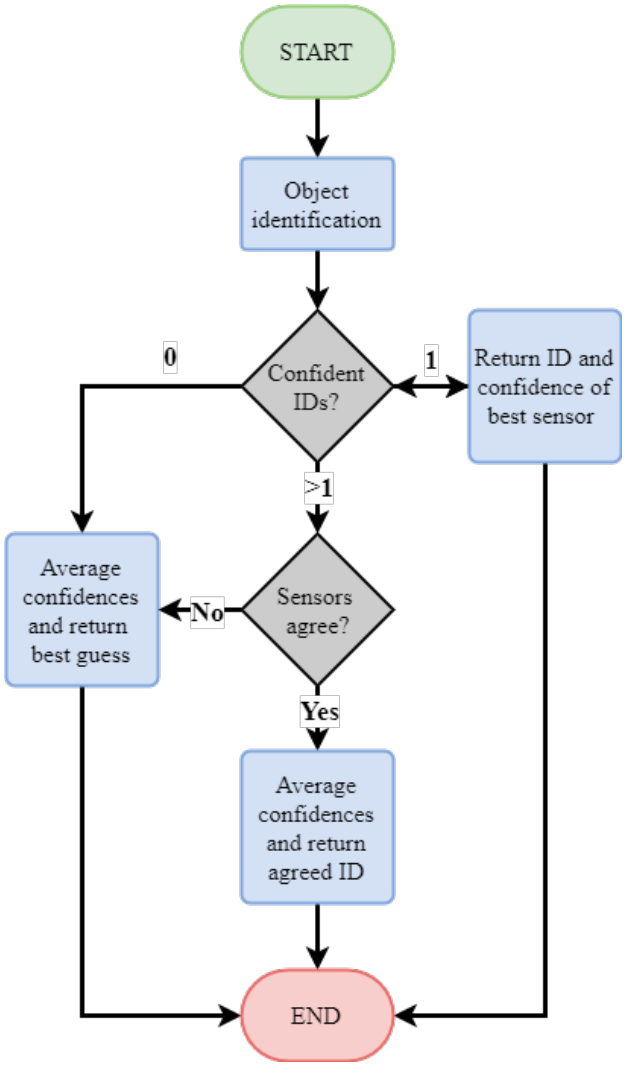

Figure 5.8: Object identification fusion procedure

5.2.2 Attitude determination

Based on the outcome of the object identity reconciliation, the relative attitude is then determined. This step is performed using the algorithm specified in Section 3.3.2.2. If no sensor confidently identifies the object, then the attitude is returned as "no solution." Assuming the solution is the measurement portion of an attitude filter, then the filter would propagate for this time step and then wait for the next measurement. If only one sensor identifies the target with confidence, then the attitude and confidence from that sensor alone is used.

The reason for using only one sensor instead of some combination of all the attitude solutions derives from the way that appearance matching performs attitude determination. Attitude solutions are determined from the object-specific eigenspace. Therefore, if attitude determination is performed using the wrong object identity, the resulted attitude solution would be useless. In other words, it does not help to "corrupt" the attitude measurement by including sensors with low object identification confidence.

For the rest of this section it is assumed that multiple sensors have agreed on an object identity. The attitude determination solutions must then be reconciled. The ideal scenario is that the sensors agree on the exact same attitude and both with high confidence. If that is the case then that attitude is returned by the algorithm with high confidence. This possibility is extremely rare, but it is included for completeness. The next best option is that the sensors agree on the attitude within a small margin for error. That possibility is reconciled using a weighted average of the confidences in each orientation direction, but the overall confidence for this case is lower than for the case where the sensors fully agree. Equation 5.15 shows how this averaging is performed. The value v_r is the reconciled confidence, and v_i are the individual sensor confidences ordered from highest to lowest. The last case is that in which more than one sensor is confident in its solution but the sensors do not agree within a margin for error. In that case, the confidences are combined according to Equation 5.16 and the attitude with the highest confidence is returned.

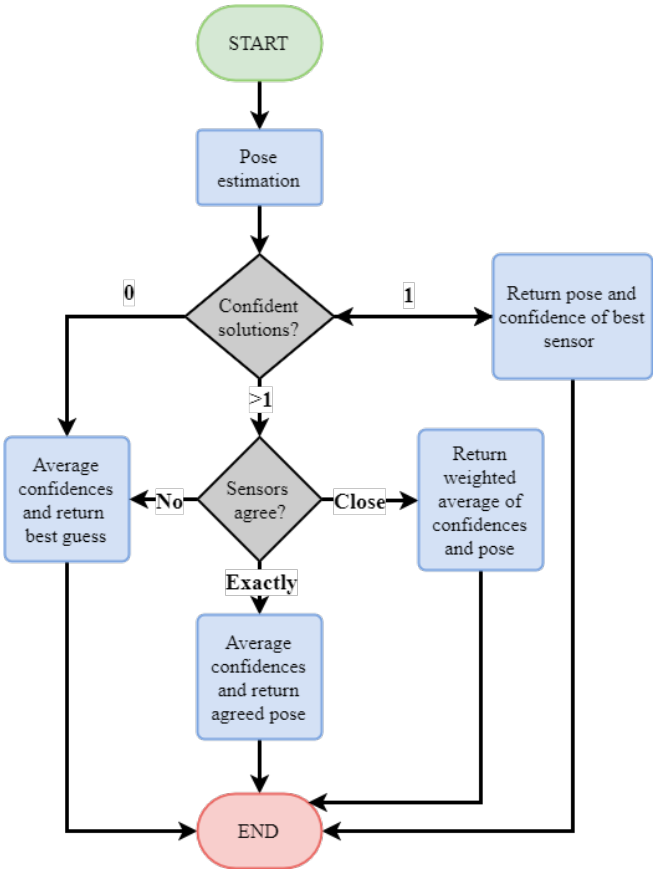

Figure 5.9: Pose estimation fusion procedure

It should be noted that the reconciliation described above applies to relative attitude. Relative position is determined from the single infrared sensor with the highest object confidence as described in Section 5.1.1.3. Figure 5.9 outlines the attitude determination fusion procedure.

$$v_r = \left(1 - \frac{v_1 - \sum_{i=2}^{n_c} v_i}{v_1}\right) v_1 \tag{5.15}$$

$$v_r = v_1 - \sum_{i=2}^{n_c} v_i \tag{5.16}$$

5.3 *A priori* and time history inclusion

The process presented here exists on the measurement side of the navigation problem. Given an input of an image or images, appearance matching and sensor fusion returns an object identity and an pose estimate (using the object identity combined with attitude determination) along with a confidence value for each of these. As presented here, the algorithm finds this information using only the single test image and the training library.

Thus, the identity and pose are estimated independently of any stored information or knowledge from additional sources other than the test image. It follows that such information could be tracked and utilized in a filter downstream of this measurement. However, the sensor fusion framework also allows for the introduction of *a priori* or time history information using a virtual sensor if desired.

5.3.1 Virtual sensor framework

The inclusion of a virtual sensor works in much the same way as an actual sensor. For a particular time step, an object identity and pose estimate is returned along with the accompanying confidence values. The difference is that instead of this information resulting from processing an image, it is derived from a source external to the current appearance matching process. That could be previous information about the identity of the object or its pose, or it could be calculated using the time history of the object or an attitude model of the given scenario.

5.3.2 Object identification

Providing object identity via a virtual sensor is a matter of calculating the probability that some previously recorded object identity changed. This is analogous to the addition of process noise to a time update in estimation theory. A simple way to do this is to reduce the confidence in the object identity proportionally to how much time has passed since the

identity was calculated. The time history confidence may also be reduced if there is a series of low confidence solutions generated by the appearance matching method.

5.3.3 Pose estimation

The best source of pose estimation values and confidences for a virtual sensor is an extended Kalman filter using a position and attitude model. Previous data can be used to estimate the pose and how well it is known. The advantage to implementing a virtual sensor in this way is that the sensor fusion appearance matching protocol is somewhat "self-correcting" without an outside estimator. Should such an estimator be implemented, however, then the virtual sensor should not be used as the estimated data would be counted twice: once by the filter and once by the virtual sensor. Implementation of such a virtual sensor is reserved for future work and described in more detail in Section 7.2.4.

CHAPTER 6
RESULTS

As part of this research, a series of tests were performed in order to prove the utility and effectiveness of appearance matching and visible-infrared sensor fusion for spacecraft visual navigation. In particular, the tests themselves fell into two categories: supporting statements made about the algorithms' performance and showing the algorithms' effectiveness in situations representative of those that would be encountered by a spacecraft on-orbit.

Two key additions to the state of the art were made by this research. The first was the addition of background matching to the appearance matching training procedure in order to allow object identification and pose estimation on an arbitrary background. To test this improvement, two libraries of training images were simulated in the visible spectrum, one using background randomization and one using a black background. A set of test images was also simulated with four different background cases to compare the appearance matching performance using the two different training libraries. Only the visible spectrum was tested in this case to simplify the presentation of the results and highlight the improvement made by background randomization.

The other addition was the application of visible and infrared sensor fusion to appearance matching. To test this procedure, training libraries of the same orientations were simulated in the visible and infrared spectra. A set of test images was then processed by the algorithm using only the visible spectrum, only the infrared spectrum, and the fusion procedure. The results from the three cases were compared in object identification accuracy and pose estimation error.

To determine the performance of the algorithms in representative conditions, the appearance matching and sensor fusion were challenged in different ways. First, image artifacts like noise and glint were introduced in order to test the robustness of visual navi-

gation measurement to these errors. Next, object identification and pose estimation were performed at increasing distances to determine the effect of distance to the sensor on appearance matching.

The final test was to create a realistic mission scenario that might be performed using this system. The scenario that was chosen had a target and chasing spacecraft in a stable Clohessy-Wiltshire-Hill relative orbit. The chasing spacecraft has simulated visible and infrared cameras to take navigation measurements of the target throughout the orbit.

6.1 Background randomization performance

Background randomization was tested using 100 test images of 4 different spacecraft using one set constructed from black-background training images and again with a set of random-background training images. The additional targets serve to test the performance of the object identification step of the appearance matching algorithm. Representative images of these objects are given in Figure 6.1. Both universal training sets consisted of a total of 10272 training images. There were 642 orientations evenly distributed over pitch and yaw, 4 lighting conditions evenly distributed over pitch and yaw, and 4 spacecraft. The distributions for lighting and orientations are given in Figure 6.2. The object sets each contained 2568 images.

The simulated camera had parameters given in Table 6.1. The mvBlueFOX3-1013G is a commercial-off-the-shelf (COTS) machine vision camera made by German company Matrix Vision. The 1013G variant uses an e2v EV76C560 sensor [90]. The lens was selected to be a Fujinon 16 millimeter lens as it is offered by Matrix Vision with the camera. The sensor-lens system has specifications given in Table 6.1. This hardware has a form factor and power specifications that could be flown realistically on a small satellite mission requiring a visible spectrum camera. A corresponding infrared camera is presented later (Section 6.2).

The PCA was performed using eigenvalues that represented 50% of the variance, ac-

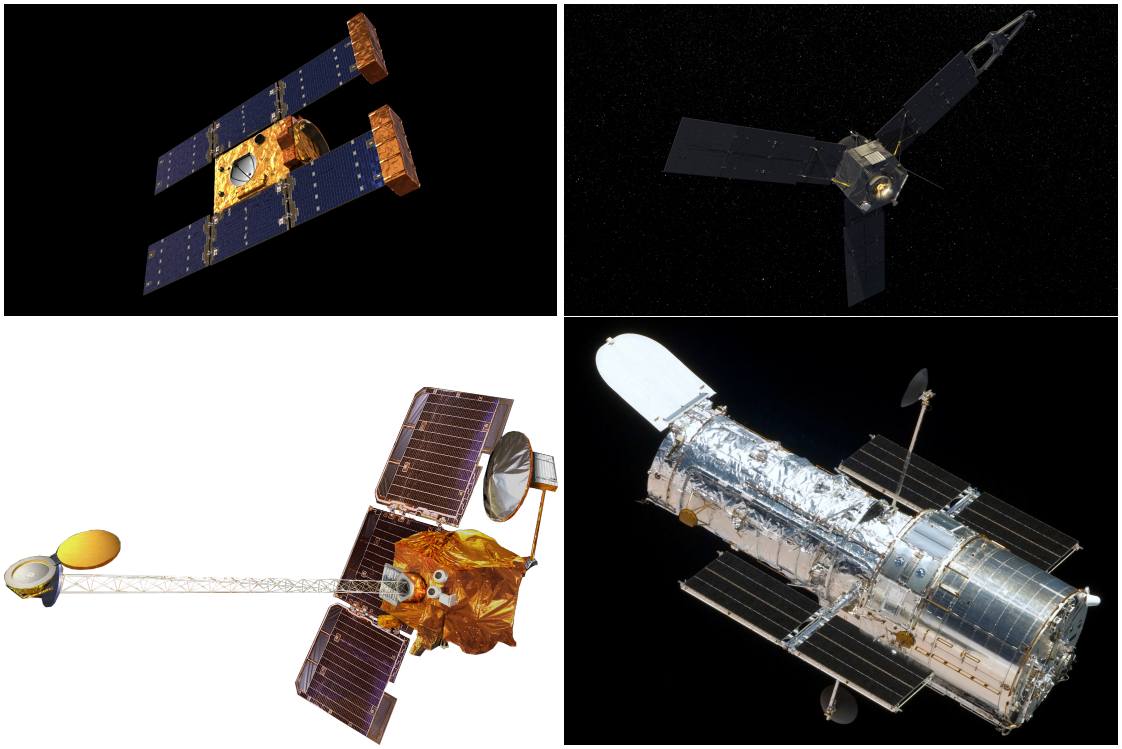

Figure 6.1: Representative images of test objects: Stardust (top left), Juno (top right), Odyssey (bottom left), and HST (bottom right) (Images credit: NASA)

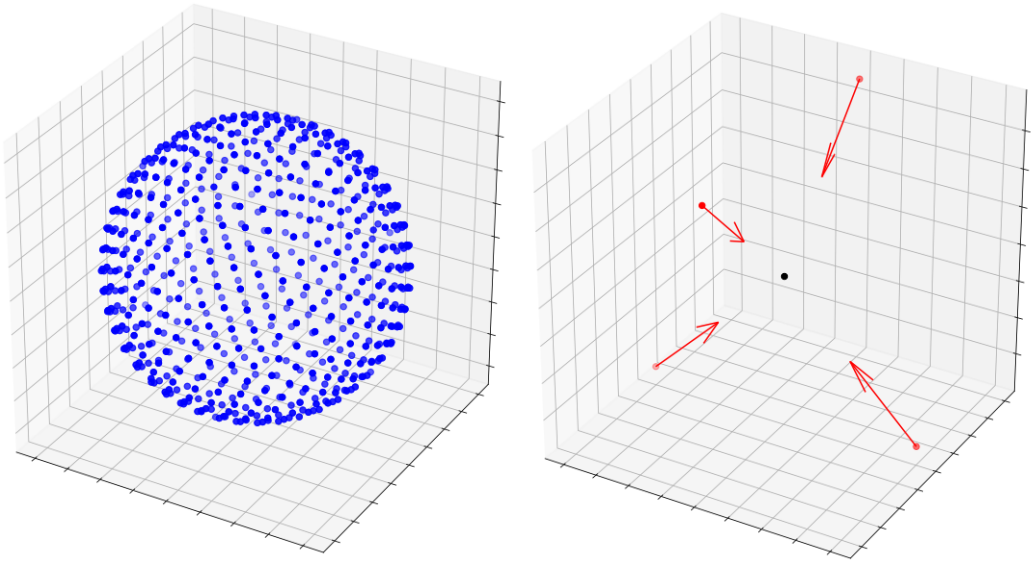

Figure 6.2: Orientation (left) and lighting (right) conditions

Table 6.1: mvBlueFOX3-1013G camera parameters

Image size (pixels)	1280×1024
Focal length (mm)	16
Aperture	f/1.4
Sensor type	CMOS
Dynamic Range	51.5 dB
Pixel size (μm)	5.3×5.3
Power	< 2.5W

Table 6.2: Object relative distances

Object	Relative distance (m)
Stardust	400
Juno	425
Odyssey	500
Hubble	2300

cording to Section 3.5.1. That led to 26 eigenvalues for the black-background set and 253 for the random-background set. The greater variance in the randomized background means that significantly more eigenvalues are needed for that set than for the black-background set. Each object was simulated at a distance that had it occupying approximately 80% of the image frame. These distances are given in Table 6.2. The effect of distance to the target on appearance matching is presented in Section 6.4.

The following sections compare the performance of the random-background training set to the one with black-background images. First, a baseline test is presented with black-background images. This test serves as a best-case scenario for the black-background set as well as showing that the addition of a random-background training does not degrade the appearance matching performance for test images with a black background. A series of tests is then presented with different test backgrounds representative of the space environment.

6.1.1 Performance on black-background test images

In testing the utility of background randomization, it was first important to show that the background randomization process did not significantly affect the accuracy for test images

Table 6.3: Black background object identification accuracy

Object	Random-background training	Black-background training
Stardust	100.0%	100.0%
Juno	100.0%	100.0%
Odyssey	100.0%	100.0%
Hubble	100.0%	100.0%
Overall	**100.0%**	**100.0%**

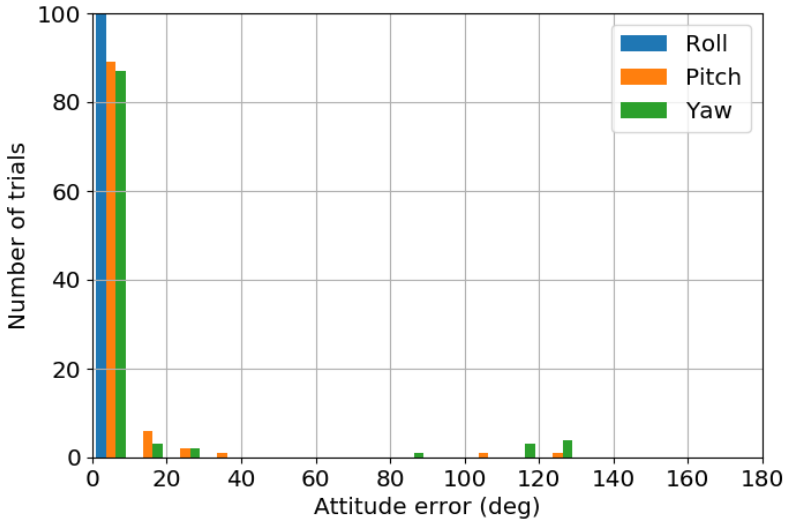

Figure 6.3: Random background attitude error by axis

that did happen to have a black or nearly black background. For the latter case, random pixels were added to account for the varying light intensity as described in Section 3.4.2. The results for object identification for each object as well as overall are given in Table 6.3. The attitude determination results are shown in Figures 6.3 and 6.4. The full pose estimate was not required for these tests since the objects were simulated a constant relative distance for each test.

These results show comparable performance between the black-background library and the random-background library using the random noise injection. In both cases, the high-attitude error cases stem from symmetricity. This fact is evident from the clustering of errors around 90, 120, and 180 degrees.

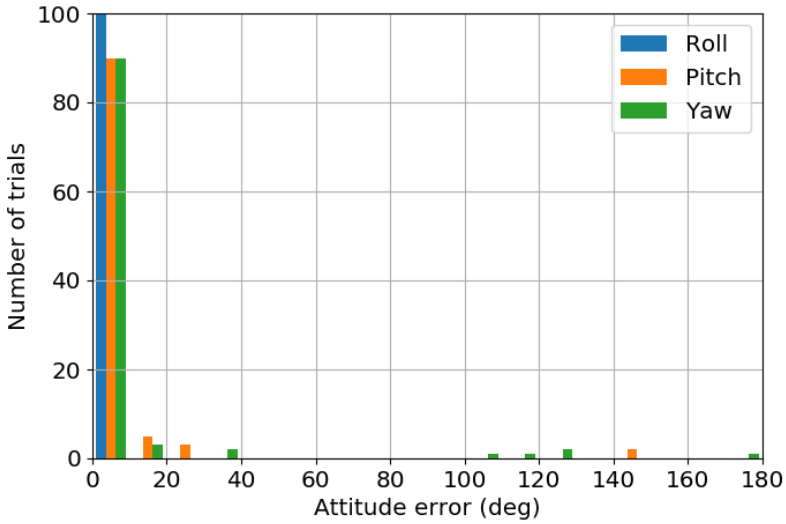

Figure 6.4: Black background attitude error by axis

Table 6.4: Star background object identification accuracy

Object	Random-background training	Black-background training
Stardust	100.0%	80.0%
Juno	100.0%	100.0%
Odyssey	100.0%	0.0%
Hubble	100.0%	100.0%
Overall	**100.0%**	**73.0%**

6.1.2 Performance on star test background

The next series of tests were performed on simulated images with a star background inserted. In theory, the black-background library should perform the best on these images, as they are the closest to pure black of the three test backgrounds. Figure 6.5 shows a sample star background test image.

Table 6.4 gives the object identification accuracy for the random-background tests similar to Table 6.3 above. Figures 6.6 and 6.7 compare the number of trials with a given attitude error per axis between the two training libraries. Note that the attitude error is only calculated when the object is correctly identified, so the two cases do not have the same total number of trials.

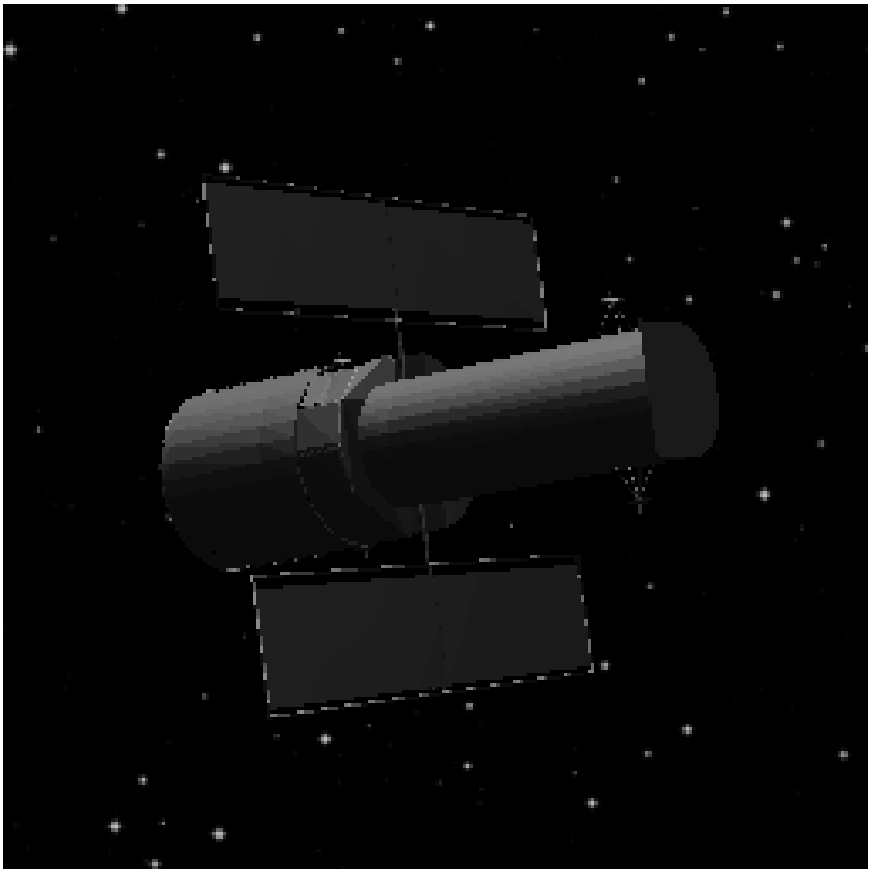

Figure 6.5: Sample random-background test image

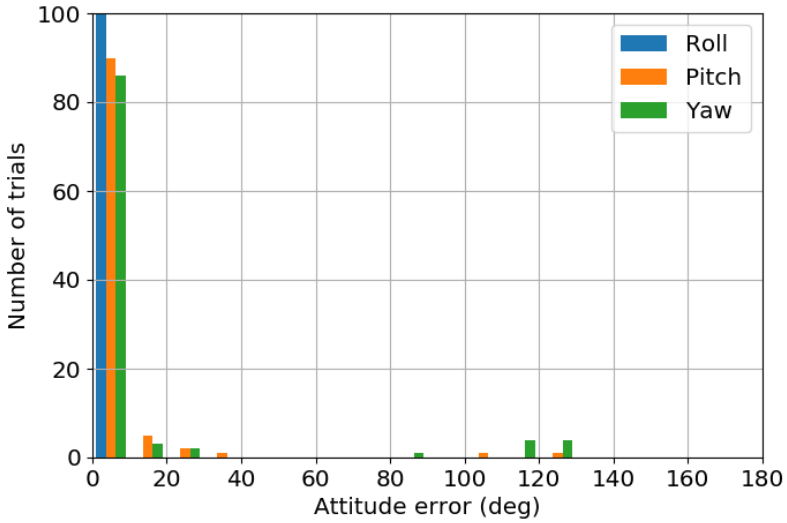

Figure 6.6: Random background attitude error by axis

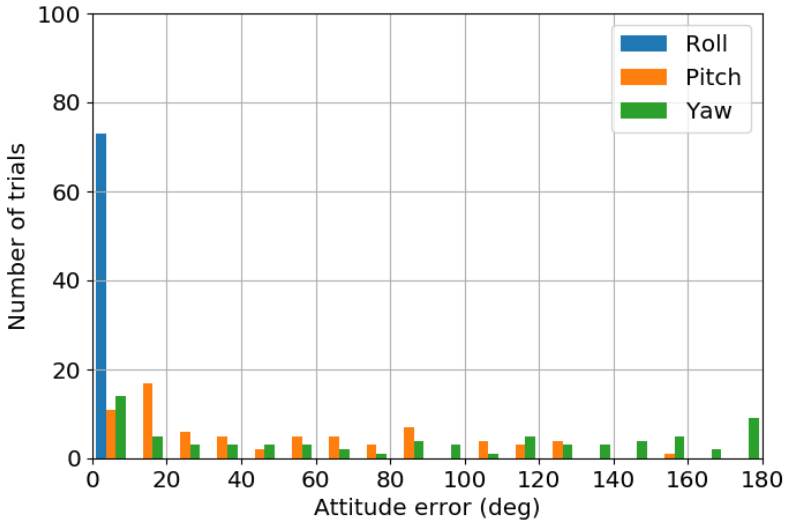

Figure 6.7: Black background attitude error by axis

Table 6.5: Cloud background object identification accuracy

Object	Random-background training	Black-background training
Stardust	100.0%	100.0%
Juno	100.0%	0.0%
Odyssey	100.0%	0.0%
Hubble	100.0%	0.0%
Overall	**100.0%**	**25.0%**

In both object identification and attitude error, the random-background library clearly outperforms the black-background library. The black-background training handles the object identification relatively well, but performs worse at attitude determination.

6.1.3 Performance on cloud test background

The next series of tests were conducted on simulated images with a background of clouds. This series was designed to emulate a spacecraft being imaged from above in low Earth orbit. Figure 6.8 shows a sample test image.

Table 6.5 gives the object identification accuracy for this case. Figure 6.9 again compares the number of trials with a given attitude error per axis between the two training libraries.

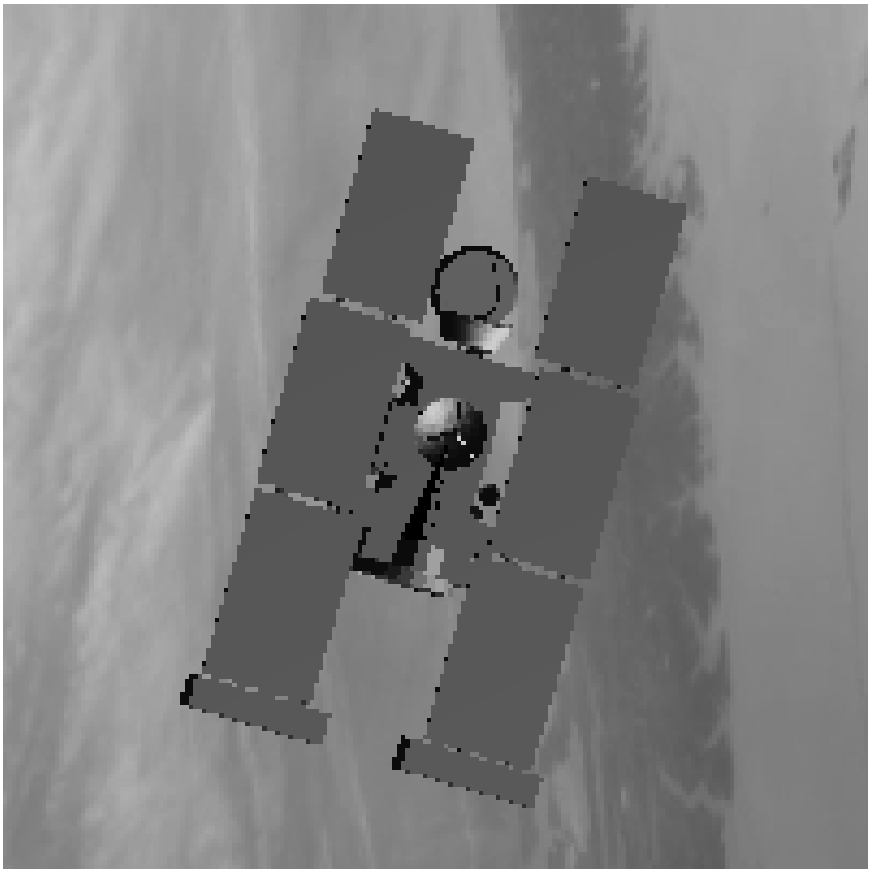

Figure 6.8: Sample cloud-background test image

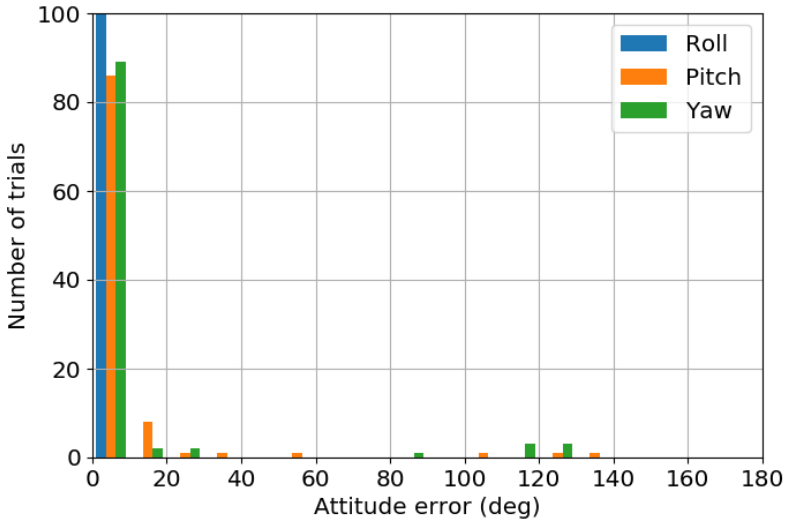

Figure 6.9: Random background attitude error by axis

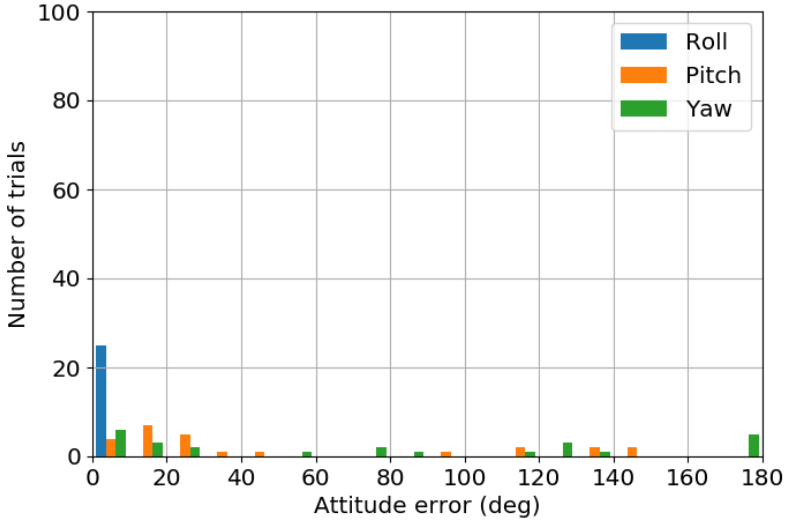

Figure 6.10: Black background attitude error by axis

Table 6.6: Horizon background object identification accuracy

Object	Random-background training	Black-background training
Stardust	100.0%	0.0%
Juno	100.0%	0.0%
Odyssey	96.0%	100.0%
Hubble	100.0%	0.0%
Overall	**99.0%**	**25.0%**

The random-background library again outperforms the black-background library. The only object which was identified correctly at all was Stardust, though that object identification was perfect.

6.1.4 Performance on horizon background

The final comparison tests were designed to provide a more difficult situation for the random-background training. The simulated background is the limb of the earth, resulting in a half black and half cloud background. Figure 6.11 shows a sample test image.

As in the tests in Section 6.1.1, any black pixels in the test image were replaced with random values. Table 6.6 gives the object identification accuracy. Figure 6.12 compares the number of trials with a given attitude error per axis between the two training libraries.

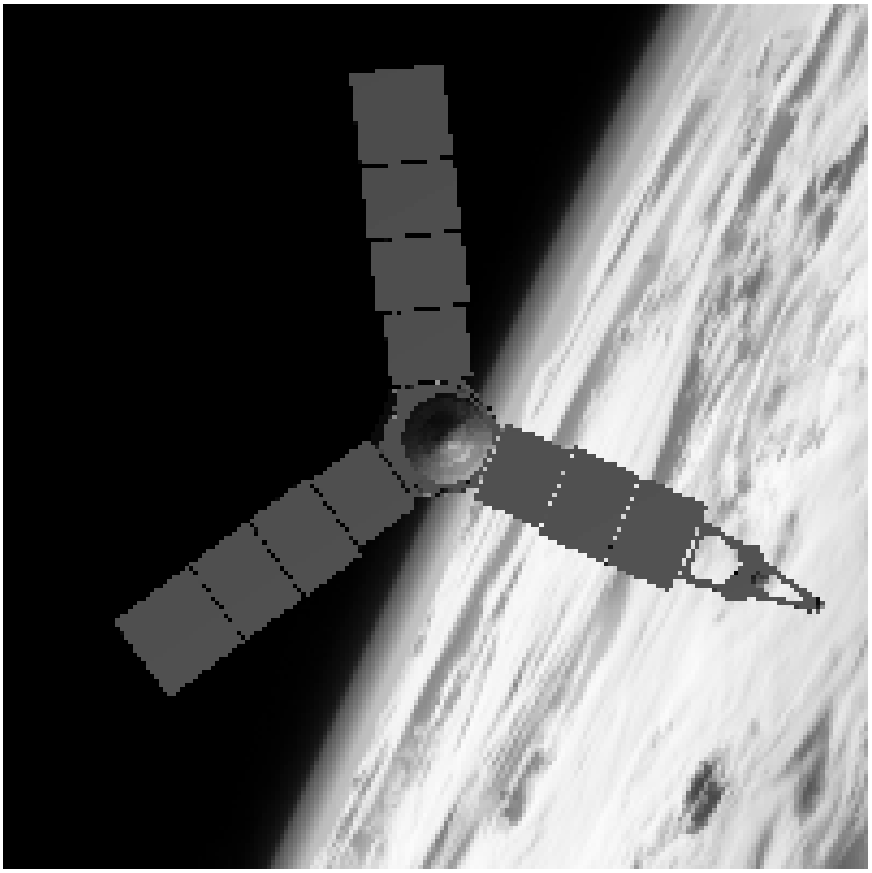

Figure 6.11: Sample horizon-background test image

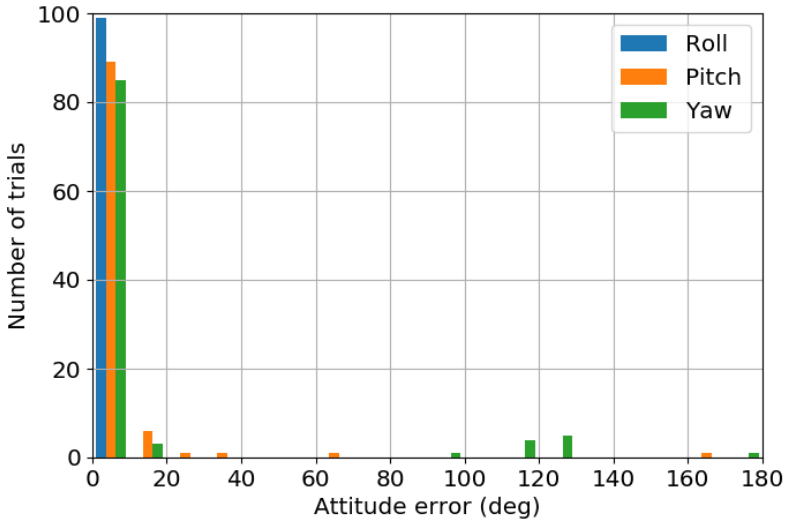

Figure 6.12: Random background attitude error by axis

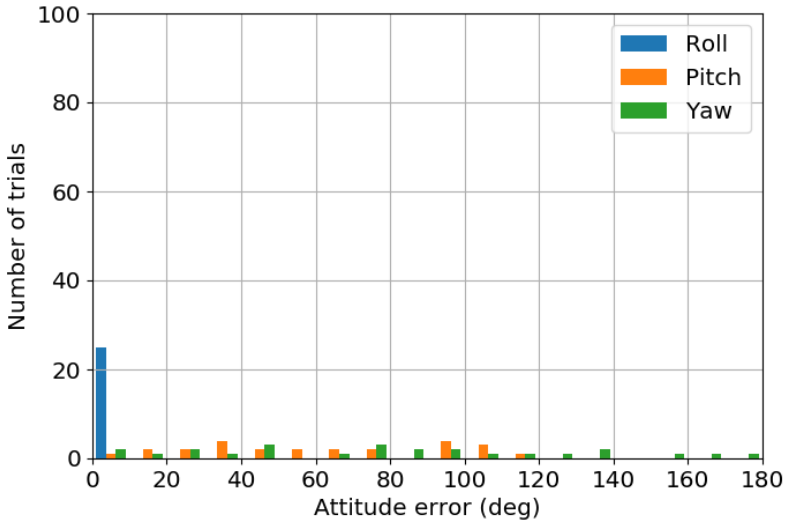

Figure 6.13: Black background attitude error by axis

As expected, this test was more challenging for the random background library, particularly for the Odyssey spacecraft. However, it still accommodates the horizon background as well than the standard black-background algorithm. Conversely, Odyssey is the only spacecraft correctly identified by the black-background library.

6.2 Sensor fusion versus single-spectrum performance

The sensor fusion protocol was tested using a software-in-the-loop test similar to that performed in Section 6.1. The same set of random attitudes and light directions were tested three times: once with the visible spectrum only, once with the infrared camera only, and once with the fusion protocol using both sensors. The visible spectrum and infrared spectrum virtual camera reflect real-world hardware that are available for use on spacecraft right now.

6.2.1 Simulated hardware

For both scenarios, representative hardware was chosen to provide realistic simulated image parameters. The assumption was made that the combined hardware system should be

Table 6.7: FLIR Tau 2 640 bolometer parameters

Image size (pixels)	640×512
Focal length (mm)	19
Aperture	f/1.25
Sensor type	Uncooled VOx
Spectral band (μm)	7.5 - 13.5
High-gain scene range (degrees C)	-25 to 135
Low-gain scene range (degrees C)	-40 to 550
Pixel size (μm)	17×17
Power	< 1.2W

of volume, weight, and power profile that could potentially fly aboard a moderately-sized CubeSat (e.g., 6U to 12U). Using this restriction, a visible spectrum camera and an infrared spectrum camera were chosen. These are the Matrix Vision mvBlueFOX3-1013G [91] mentioned previously and the FLIR Tau 2 640 [92].

The Tau 2 640 is a long-wavelength infrared (LWIR) bolometer made by American company FLIR. The Tau 2 640 uses an uncooled vanadium oxide (VOx) core with a spectral band of 7.5 to 13.5 μm which allows it to sense temperatures from -25 to 135 degrees Celsius in high-gain mode and from -40 to 550 degrees Celsius in low-gain mode. This range adequately spans the usual operating temperature range of spacecraft as well as temperatures of passively heated objects in low Earth orbit. FLIR offers lenses of various focal lengths; the 19 mm lens was selected to give the bolometer a similar field of view as the visible spectrum camera. Detailed specifications are given in Table 6.7.

6.2.2 Results

The following are the results of the software-in-the-loop tests for the three sensor measurement cases: visible only, infrared only, and fusion. The test images were the same for each case, and the results of the single sensor cases were used for the fusion case. The same four objects were used as in the previous tests, and a number eigenvalues sufficient to cover 50% of the variance for each sensor were used: 12 for the infrared camera and 82 for the visible spectrum camera. Table 6.8 compares the object identification accuracy of each case. Table

Table 6.8: Object identification accuracy for different spectra

Object	Visible only	Infrared only	Fusion
Stardust	96.0%	100.0%	100.0%
Juno	100.0%	100.0%	100.0%
Odyssey	100.0%	100.0%	100.0%
Hubble	100.0%	100.0%	100.0%
Overall	**99.0%**	**100.0%**	**100.0%**

Table 6.9: Sensor fusion attitude error statistics

Spectrum	Median pitch error (deg)	% large error	Median yaw error (deg)	% large error
Infrared	4.18	29.0	2.14	22.0
Visible	3.265	16.0	1.14	15.0
Fusion	3.17	16.0	1.14	15.0

6.9 gives the median attitude error in each axis for each case as well as the percentage of attitude errors in each axis which fall outside the theoretical maximum error based on the number of training orientations, called "large error".

The results here are what one would reasonably expect. The effect of varying lighting is less apparent for the infrared spectrum, which leads to better object identification accuracy. However, the detail and lighting variation in the visible spectrum images means that the attitude determination is better for those images. Finally, the fusion results combine the best results of both spectra, with as good or better results in both categories.

The more granular results from these three figures back up the overarching results presented in Table 6.8. Figure 6.16 shows fewer high-error axes than Figure 6.14 and a similar number to Figure 6.15. This comparison is also evident in Figure 6.17, which shows more high-error red bars, representing the infrared performance.

6.3 Robustness of appearance matching to imaging error

In order for appearance matching to be a viable technique for spacecraft navigation, it must be able to work in the presence of common types of errors which may distort the test image. These errors derive from mechanical imperfections in the camera, like distortion and blur,

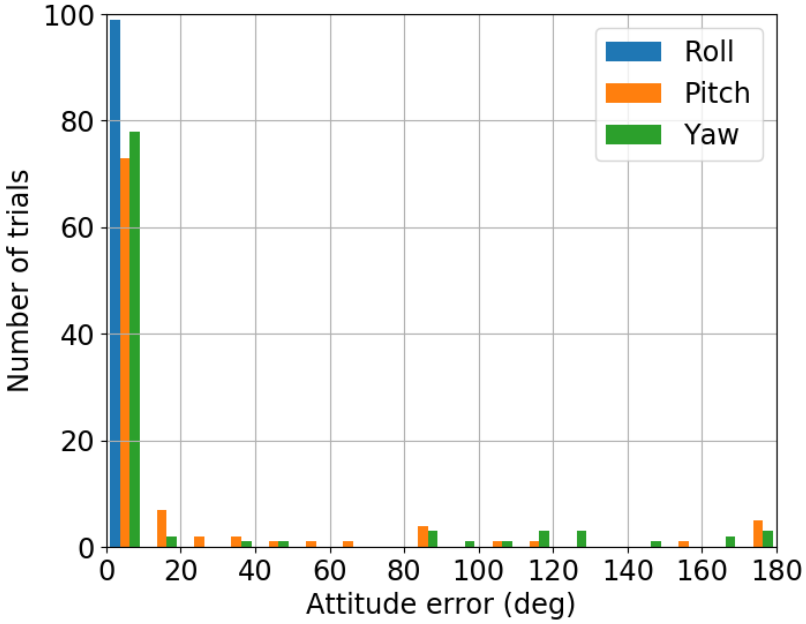

Figure 6.14: Infrared spectrum attitude error by axis

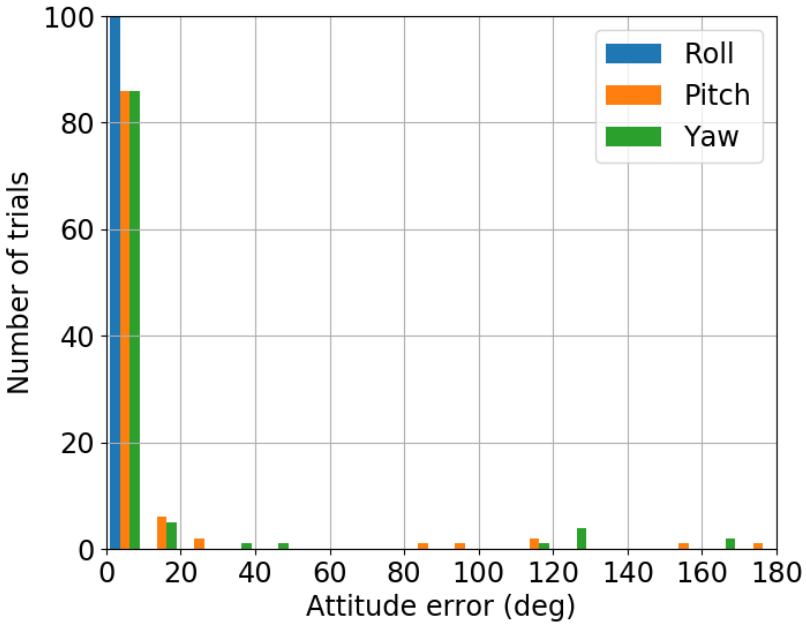

Figure 6.15: Visible spectrum attitude error by axis

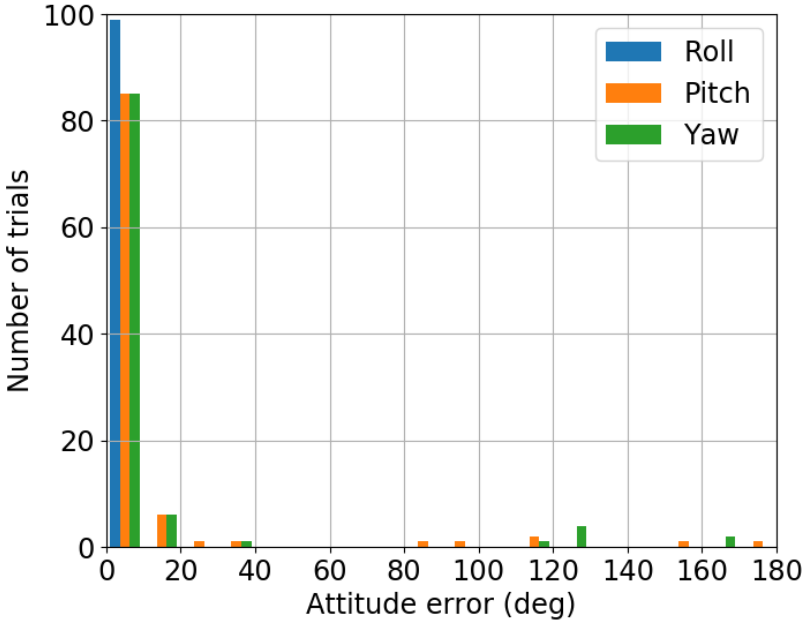

Figure 6.16: Fusion spectrum attitude error by axis

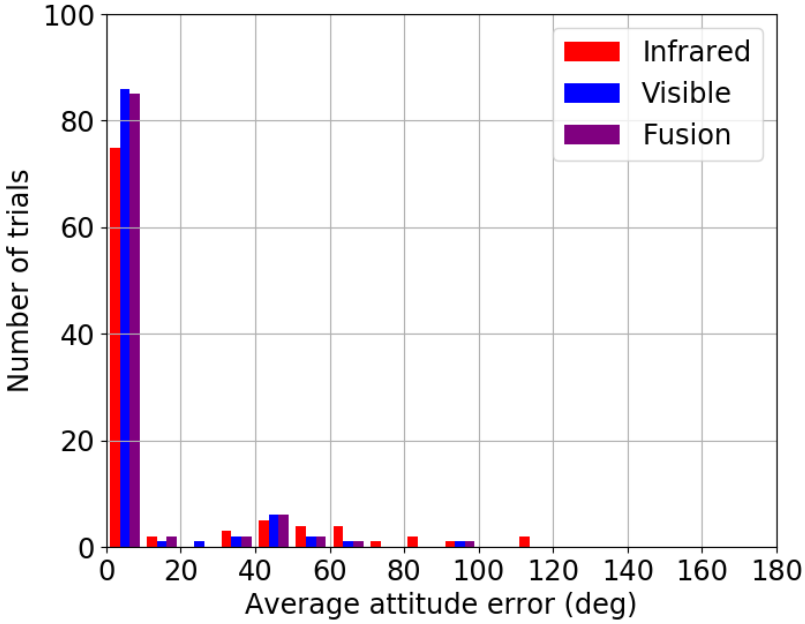

Figure 6.17: Comparison of average attitude error by spectrum

from noise in the analog-to-digital processing of the image, or from effects due to the nature of the target like glint. Each one of these error sources was introduced one by one into the simulated test images to see how they affected the object identification and pose estimation accuracy. The same simulated hardware was used as in Section 6.2, though the cameras were not tested individually, only in the fusion configuration.

The training configuration includes 642 orientations under 4 lighting conditions of the 4 objects used in previous software-in-the-loop tests. A number of eigenpairs was used for each spectrum to account for 60% of image set variance according to Section 3.5.1, resulting in 18 eigenpairs for the infrared camera and 412 eigenpairs for the visible camera. This training configuration was used for every one of the tests in this section.

The test scenario consisted of 8 images each of the four objects. These images were generated using random orientations, lighting, and at two distances from the sensing spacecraft: the relative distances listed in Table 6.2 and 4 times those relative distances. According to the results in Section 6.4, no appreciable degradation in performance occurs over that distance range for the error-less case. Thus, any impact on performance is due to the introduced errors. Object identification, attitude determination, and range errors are presented for each error type.

6.3.1 Distortion

As previously discussed in Section 4.5.5.1, distortion is the result of imperfections in the camera lens. Typically, these distortions are modeled on the ground before a camera is launched. Standard procedures exist for this kind of modeling [93]. These effects would be considered when the training library is simulated. However, additional distortion is sometimes introduced in vibrations during launch or thermal effects. Therefore, showing that appearance matching is robust to a reasonable amount of distortion is important. For these tests, a first distortion coefficient of 1^{-8} was used as the test case applied using Equation 4.13. The hardware being simulated in this test was calibrated using the MATLAB calibra-

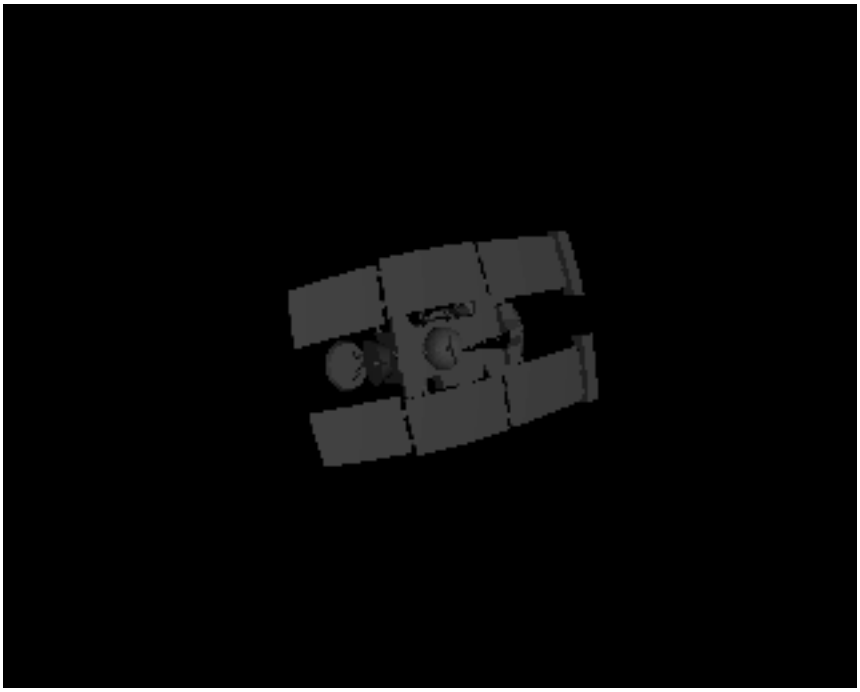

Figure 6.18: Example distorted image

tion procedure referenced earlier [93]. The test value is an order of magnitude worse than the worst distortion value produced by that process, representing distortion from a lack of calibration compounded by an additional effect. An example distorted image is shown in Figure 6.18.

6.3.2 Blur

Blur in a test image may result from two main sources. First, if the shutter time of the camera is too long, the target may be blurred as is passes through the image frame. Blur resulting from incorrect shutter speed may be compensated for within the imaging software routine. Second, the focus point for the camera may differ from the distance from the camera to the target. If the camera has a fixed focal length, then it is not possible to fully account for this error on-orbit. Thus, appearance matching must be able to accept a reasonable amount of blur. For these tests, a Gaussian blur was applied with a standard deviation of 2 pixels using Equation 4.14. The amount of blur in an image may vary

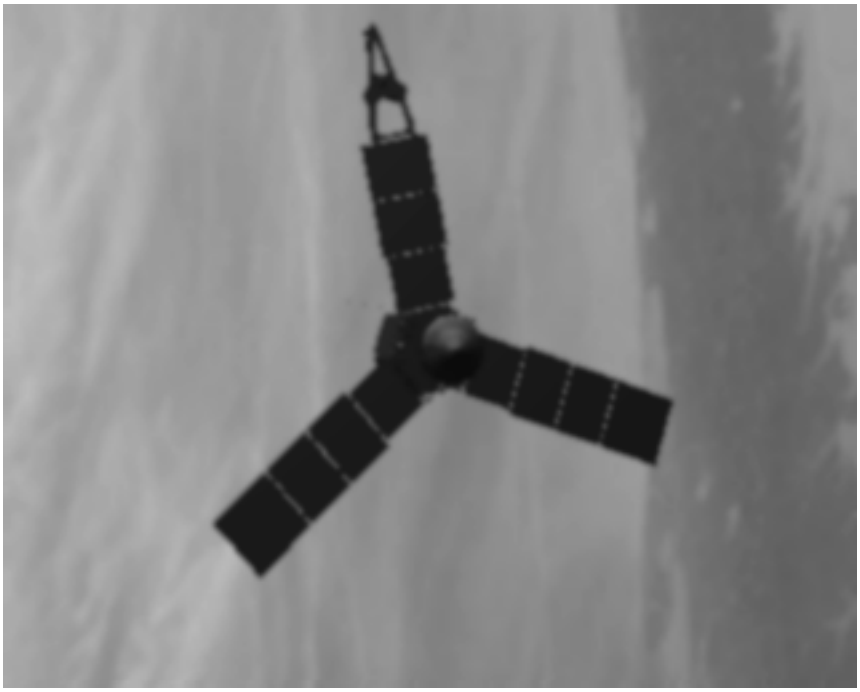

Figure 6.19: Example blurred image

significantly based on the factors previously described. The amount of blur was selected as it blurred the image enough to be noticeable but not so much that the object was no longer recognizable. An example blurred image is shown in Figure 6.19.

6.3.3 Noise

Noise in a digital image is introduced as an artifact of analog to digital conversion, as described in greater detail in Section 4.5.5.2. Noise was introduced to to the test images according to Equation 4.15 and the cameras' digital parameters. An example noisy image is shown in Figure 6.20. The amount of noise in the image has an upper limit set by the specifications of the camera. The amount simulated for this test is that worst-case scenario.

6.3.4 Glints

Glints in an image are the result of localized regions of high specular reflection. By skewing the illumination or even overall shape of a target object, glints have the potential to have an

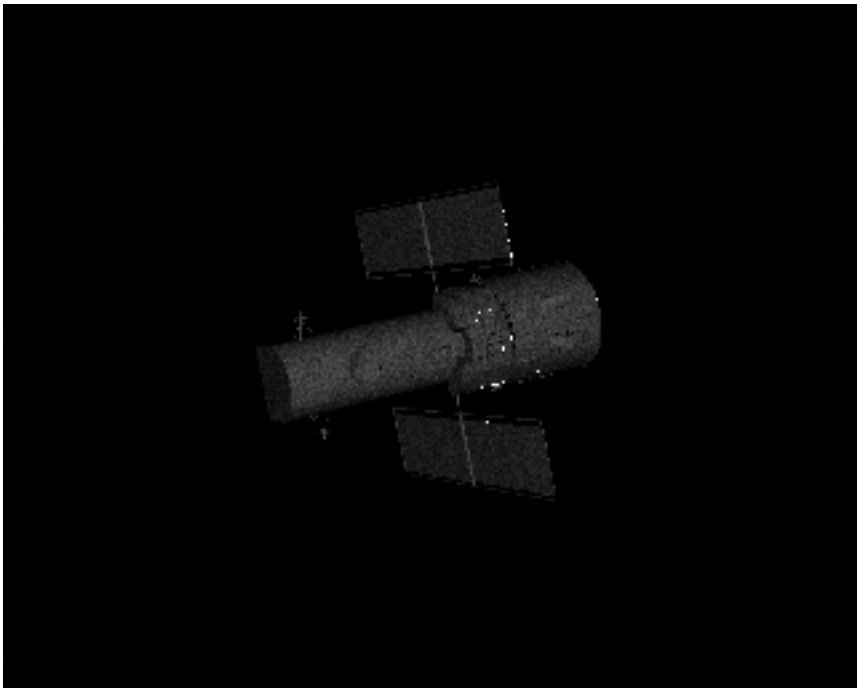

Figure 6.20: Example noisy image

adverse effect on appearance matching procedure. Glints for this test were simulated by an increase in the specular coefficient and exponent of Equations 4.1 and 4.10. These values were selected to produce a glint effect as similar as possible to actual images of spacecraft subject to glint. An example image with a glint is shown in Figure 6.21.

6.3.5 Other effects

As discussed in Section 4.4.2, blooming and saturation were not modeled for this research. Modern sensors can compensate for these effects, so the appearance matching procedure does not need to be robust to them. If the algorithm is required to perform on older hardware, the effect of sensor effects should be examined.

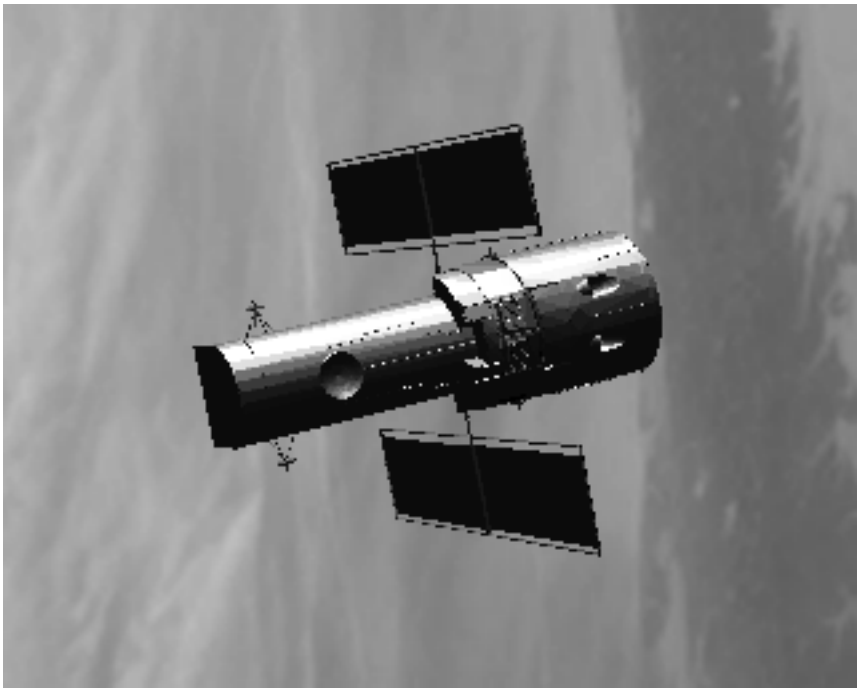

Figure 6.21: Example image with glint

Table 6.10: Error effect on object identification accuracy

Error type	Reference distance (RD)	4×RD
No error	100.0	100.0
Distortion	100.0	100.0
Blur	100.0	100.0
Noise	100.0	96.9
Glint	96.9	100.0

6.3.6 Results

6.3.6.1 Object identification

The object identification accuracy was very robust to the various errors, with only a few failed identification cases for noise and for glint (Table 6.10). The accuracy was good at both the near and farther relative distances.

Table 6.11: Error effect on pitch error

Error type	% large pitch error at RD	% large pitch error at 4×RD
No error	6.25	25
Distortion	6.25	18.75
Blur	3.125	62.5
Noise	6.25	28.125
Glint	15.625	28.125

Table 6.12: Error effect on yaw error

Error type	% large yaw error at RD	% large yaw error at 4×RD
No error	6.250	18.750
Distortion	12.500	15.625
Blur	9.375	40.625
Noise	6.25	28.125
Glint	15.625	21.875

6.3.6.2 Attitude determination

In general the fused appearance matching algorithm achieved good results given the errors with respect to attitude determination. Tables 6.11 and 6.12 show the percentage of large error cases as defined in Section 6.2.2. All of the imaging error cases resulted in a slight increase in large attitude error cases, with the exception of blur, which had significantly greater numbers of large attitude error cases at the further distance. The blurred image along with the smaller number of pixels imaged had a large effect on the shape and illumination necessary for accurate attitude determination.

6.3.6.3 Relative position

In terms of relative position, the various imaging errors had a slight effect on the relative position error as measured by object range. Table 6.13 gives this error as a percentage of the reference distance. This effect was increased at the farther distance, with the relative position performance suffering more for the imaging error cases than for the case with no imaging error, with the blurry case again performing the worst of the error cases. As is discussed later in Section 6.6, a relatively small amount of error is acceptable in relative

Table 6.13: Error effect on relative position error

Error type	Reference distance (RD)	4×RD
No error	1.58%	3.55%
Distortion	4.52%	3.30%
Blur	2.90%	9.41%
Noise	1.66%	6.02%
Glint	2.67%	6.47%

position because pose estimation with appearance matching would not be used for docking. Once the image exceeds the image frame the object cannot be positively identified, a step that is a prerequisite to pose estimation.

6.3.6.4 Summary

Based on these results, it is clear that distortion has the greatest effect on performance at the reference distance and blur has the greatest effect on performance at the farther distance. This effect is seen in all three phases of appearance matching: object identification, attitude determination, and relative location. As such, it would be important in a real-world application to make sure the focal lengths of the cameras are selected carefully, and likely with an emphasis on a farther focal point than a nearer one. Additionally, the cameras should be calibrated as well as possible on the ground before going into space to reduce distortion error.

6.4 Effect of distance on appearance matching

In addition to testing the performance of background randomization versus black-background training, it is also important to show that the improved appearance matching algorithm has utility as a spacecraft relative navigation technique. Part of that verification is whether or not objects can be identified and their attitudes determined at various distances. An analogous question is how few pixels are needed for the object to be correctly analyzed by the algorithm.

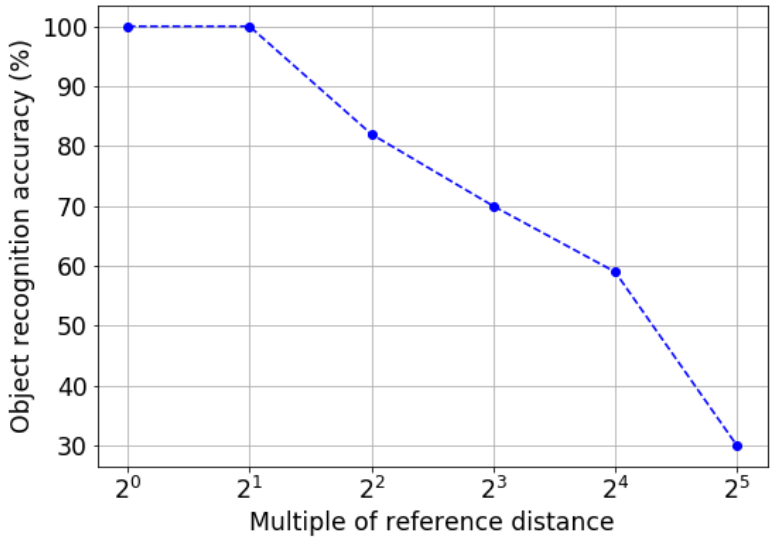

Figure 6.22: Recognition accuracy versus object distance

For this test, the same library of 4 spacecraft was used as before. Using cloud background test images, a set of 100 random orientations (25 per object) was simulated at increasing distances from the virtual sensor. Since the spacecraft are different sizes, the distance from the camera is given in terms of multiples of a reference distance. That value is the closest distance from the camera for which the entire object is still visible in the field of view. Those reference distances are given in Table 6.2. It should be noted that for this test, it is assumed that the object was located on the background using infrared masking (Section 5.1.1.3).

The object identification accuracy and median attitude error average over the three axes was calculated at each distance. Those values are given versus object distance in Figures 6.22 and 6.23, respectively.

The object identification accuracy certainly degrades as the object recedes from the camera, maintaining 100% accuracy at the first two steps and then declining steadily as the object recedes. However, an accuracy of at least 80% is maintained even when the object takes up only 1/16 of the frame, which translates to 64 by 64 pixels. The attitude error significantly degrades starting at 8 times the reference distance. However it should be noted

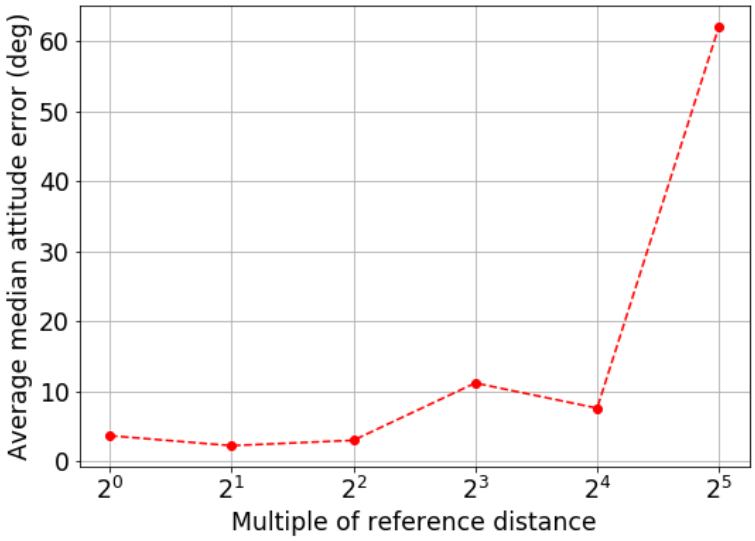

Figure 6.23: Average median attitude error versus object distance

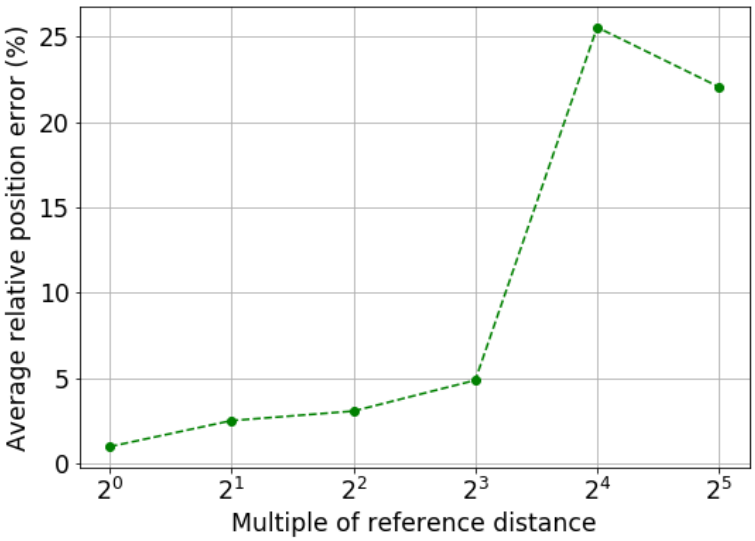

Figure 6.24: Relative position error versus object distance

that the further away the target is, the less critical it is to have exact attitude knowledge, especially for a rendezvous or intercept scenario. It is also apparent that the relative position error increases with distance and as object identification accuracy becomes poorer. This result is logical since the relative distance is found by comparing the normalization scale between the test image and the training library.

6.5 Hardware-in-the-loop test

Running software-in-the-loop tests for visual navigation is useful for algorithm tuning as well as verifying the performance of the technique under ideal conditions. However, it is important for appearance matching to perform well on real images as well. The most representative test would be to image a spacecraft with known geometry on-orbit using a camera with known characteristics. Such a test was not feasible in the time frame of this research. Instead, hardware images were taken of 3D printed analog. This analog was created using the same computer-aided design (CAD) file that was used to create the simulated images for appearance training. Thus, the appearances of the real and simulated images should match closely.

6.5.1 Analog model

The 3D printed analog used for these tests was created using the CAD file based on the file used for the Juno spacecraft from the earlier software-in-the-loop tests. Due to the way the solar panels were modeled in the earlier file, it was difficult to create an accurate model. The model used for test fills in holes in the panel to create a solid face on each side available on the website Thingiverse [94]. The analogs were printed using a plastic called polylactic acid (PLA) on an Ultimaker 3 printer [95].

Figure 6.25: Original Juno model (left), modified model (center), and printed analog (right)

6.5.2 Test setup

Three images are given in Figure 6.25. The first is the Juno model used for the previous tests, the second is the modified model for 3D printing, and the third is an image of the printed analog. The analog was imaged on a pure black background and a bright lamp was used to simulate the sun's illumination of the objects the only illumination source for this test. The training library also used a black background with 252 total orientations, 6 in the roll direction and 42 in the pitch-yaw directions. The same four illumination directions were simulated as in the previous tests. In order to test the object identification performance, images of the Odyssey and Hubble models were also simulated and added to the training library. Thus, the universal training set had 3024 images and each object set had 1008 images. The PCA was performed with 240 eigenpairs accounting for 70 percent of the image set variance.

There were 25 images of the analog taken with the mvBlueFOX3 camera (Table 6.1). Rather than try to measure the orientation of the analog to determine a truth value, simulated images at random attitudes were generated and the analog was positioned to match the appearance of those images. The attitude of the simulated image was taken as the truth value for the test image. In addition, the light source was placed at different angles for each image. Since the light source direction was not being measured in the test, and exact direction was not needed.

6.5.3 Results

6.5.3.1 Object identification

The object identification accuracy of the image set was 92%. Twice the analog was misidentified as the Hubble model in a similar orientation. In both cases the object confidence was less than .7, lower than the confidence values for any of the correct identifications.

6.5.3.2 Attitude determination

The attitude determination performance was not as strong as in the software in the loop tests. Of the 23 correctly identified images, 16 or 69.5% returned an attitude solution within the minimum error for the library. There were 5 images, or 21.7%, that returned an attitude within the minimum error except for a symmetry error of 120 degrees, reflecting the three-way nearly axisymmetric nature of the Juno model. The remaining 2 images, or 8.70%, returned a non-symmetric error, averaging 23.2 degrees in each axis.

6.6 Mission scenario

The previous results in this chapter show that appearance matching with background randomization is an accurate object identification and attitude determination technique on an arbitrary background that can be extended to return a full pose estimate. In addition, sensor fusion between visible and infrared spectrum cameras improves the operational capacity of a visual navigation system. However, it is also important to demonstrate that this system also will work with a more realistic visual navigation scenario. One such scenario was constructed and tested using simulated images: in a Clohessy-Wiltshire-Hill stable relative orbit. The hardware simulated in these scenarios is the same as those in previous sections.

6.6.1 Scenario setup

The formation flying scenario used for this test uses two satellites in orbit around the Earth. The target spacecraft is in a circular orbit with a semi-major axis of 6730 km and an inclination of 51.5 degrees, approximating the orbit of the ISS. The model used was the Stardust spacecraft. The sensing spacecraft is given similar parameters but modified so that it is in a stable CWH "football orbit" in the frame of the target spacecraft. That has relative dimensions of 500×1000 meters. The orbits of both spacecraft are simulated using two-body orbital mechanics.

The attitude of the target spacecraft is held constant, while the sensing spacecraft is assumed to be given an input that maintains the target in the center of the image frame. That attitude is calculated for each time step using the spacecraft's relative position and the triad method [96]. The triad method was an early solution to the satellite attitude determination problem, producing the direction cosine matrix (DCM) relating two linearly independent reference vectors $\mathbf{R}_1$ and $\mathbf{R}_2$ in one reference frame to the same vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ in the body-fixed frame. The DCM $\mathbf{C}$ is found using Equations 6.1 through 6.5. To align the vector between the spacecraft with the camera boresight in the z-direction, the vectors $\mathbf{R}_2$ and $\mathbf{r}_2$ are identically the y-unit vector, $\mathbf{r}_1$is the z-unit vector, and $\mathbf{R}_1$ is the unit vector to the target spacecraft from the sensing spacecraft.

$$\hat{\mathbf{s}} = \frac{\mathbf{r}_1}{||\mathbf{r}_1||} \tag{6.1}$$

$$\hat{\mathbf{S}} = \frac{\mathbf{R}_1}{||\mathbf{R}_1||} \tag{6.2}$$

$$\hat{\mathbf{m}} = \frac{\mathbf{r}_1 \times \mathbf{r}_2}{||\mathbf{r}_1 \times \mathbf{r}_2||} \tag{6.3}$$

$$\hat{\mathbf{M}} = \frac{\mathbf{R}_1 \times \mathbf{R}_2}{||\mathbf{R}_1 \times \mathbf{R}_2||} \tag{6.4}$$

$$\mathbf{C} = \begin{bmatrix} \hat{\mathbf{S}} & \hat{\mathbf{M}} & \hat{\mathbf{S}} \times \hat{\mathbf{M}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{s}} & \hat{\mathbf{m}} & \hat{\mathbf{s}} \times \hat{\mathbf{m}} \end{bmatrix}^T \tag{6.5}$$

The simulation is run for a complete orbit of the spacecraft about the Earth. In order to simulate the eclipse of the satellite by the earth, the illumination from the sun is turned off for 50% of the orbit, during which only minimal ambient light is available in the visible spectrum. The full relative pose is determined at a 120 second time step using the full fusion system.

6.6.2 Results

The results for this scenario are given in Figures 6.26 and 6.27. The target object entered eclipse at 1376 seconds and exited eclipse at 4204 seconds, meaning that was eclipsed over the full range of relative distances twice. This portion is represented in figures by a dashed line. Object identification was verified at each time step and maintained throughout the scenario, so those results are not presented here. Also, since the position of the spacecraft in the image frame stayed consistent and the relative attitude varied only slightly, the range was the key component of this scenario. That observation is backed up in Figure 6.27, which shows that the relative error varies from approximately 0 to 4 percent but does not have any relationship to the range value itself. It should also be noted that the range is overestimated the worst at the greatest distance, which is preferable to being underestimated and risking a collision. It has already been discussed in this research that for docking, another algorithm would have to be used since appearance matching cannot produce a solution if the object exceeds the image frame.

Additionally, one might expect this error to reflected in Figure 6.27, but that plot displays relative error instead of absolute error. For example, the absolute error at 4920 seconds is 31.87 meters and at 4200 seconds is 37.15 meters, but the former time step has slightly higher relative error.

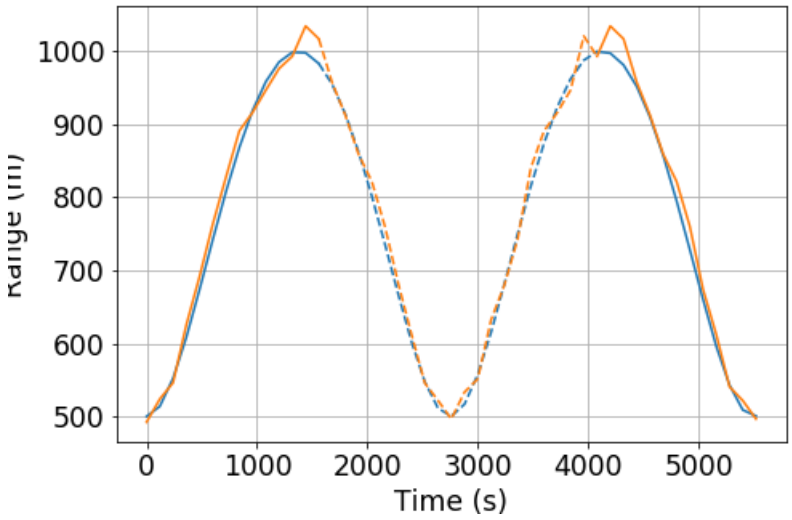

Figure 6.26: Recorded range (green) versus actual range (blue), eclipse portion dashed

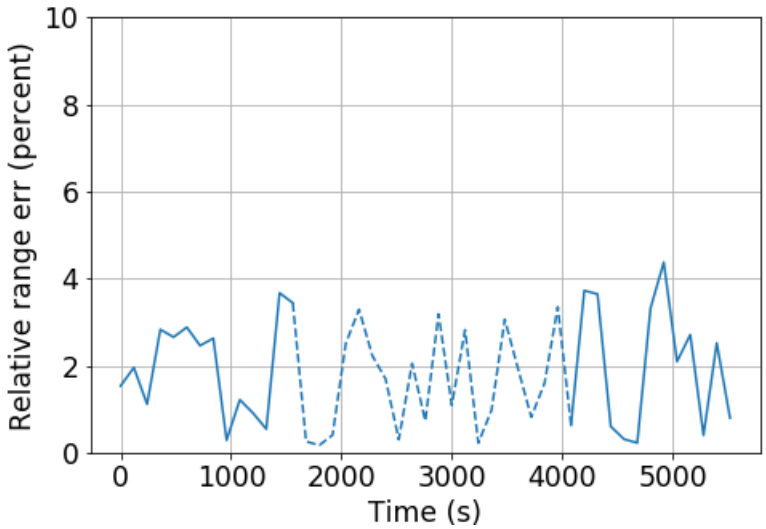

Figure 6.27: Percent range error over time, eclipse portion dashed

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Summary of results

This dissertation uses the terrestrial techniques of appearance matching and vision and infrared sensor fusion that have not been previously applied to spacecraft. As described in Chapter 1, contributions are made in three areas:

1. Appearance matching for spacecraft is enhanced using background randomization;
2. A spacecraft imaging simulation environment software tool is created for analysis; and
3. Object identification and pose estimation is performed with sensor fusion using reconciled PCA with simulated spacecraft images.

In Chapter 3, the technique of appearance matching is applied to spacecraft object identification and attitude determination. The theoretical basis of appearance matching was explained, including image compression and the Karhunen–Loève Transform. The fundamental results from Murase and Nayar [1] were discussed, including the basic procedure for object identification and attitude determination through the construction of the universal and object-specific eigenspaces. Robustness was added to the appearance matching method through the introduction of background randomization. Finally, the effect of the two tuning parameters of training library size and number of eigenvalues was discussed.

Chapter 4 introduced the spacecraft imaging simulation environment (SISE). First, the SISE was compared with respect to other image simulation tools. The fundamentals of image simulation were explained, including ray tracing, the formation of a triangular mesh, and the simulation of the two types of radiation: visible and infrared. The necessary fidelity

for such an environment was discussed, particularly with regard to thermal simulation and different error sources. The computer architecture for the SISE was described, including the computational enhancement provided using the CUDA protocol.

In Chapter 5 visible and infrared spectrum sensor fusion was introduced in the context of appearance matching. It was shown how masking using the infrared camera is able to isolate a target on a background in the visible spectrum image. This process also combines with object identification to produce a pose estimate. Hybrid PCA was briefly described in order to show how reconciled PCA is the better fusion technique. The fusion process for both object identification and attitude determination was shown. Finally, the virtual sensor framework was introduced as a way to allow the fusion algorithm to take input from *a priori* conditions and external sources.

Finally, Chapter 6 presented results verifying the concepts discussed in the previous chapters. Software-in-the-loop tests demonstrated how background randomization allows PCA to operate on multiple different backgrounds and how sensor fusion improved object identification and relative attitude determination versus each sensor alone. Sensor fusion appearance matching was also challenged with different types of errors and relative distances and performed successfully. A test case was presented to demonstrate this performance in a realistic context using a target and sensing spacecraft in a stable CHW relative orbit.

7.2 Future work

While the new methods demonstrate promise for spacecraft vision-based relative navigation, more work can be done to improve the algorithm's performance. Some suggested improvements are presented as potential topics for future research.

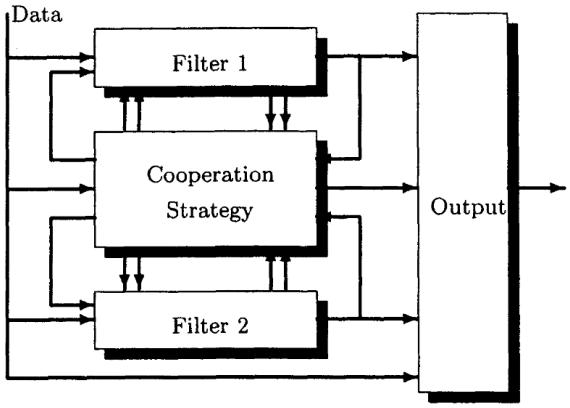

Figure 7.1: General structure of a multiple model estimation algorithm with two filters [97]

7.2.1 Multiple model filtering

In Chapter 5, the assumption was made that a single object identity and pose estimate was desired at each time step. However, one area of potential follow-on work would remove that assumption. In the case of two or more high confidence results, a filter could be implemented to "track" both cases. It would maintain each hypothetical case until additional information eventually collapsed them into a single solution. Implementing such a system was beyond the scope of this dissertation, which focused primarily on producing measurements and confidence values. However, adding a multiple model extended Kalman filter (MMEKF) or a type of particle filter is a logical next step.

Li and Jilkov [97] give an overview of the multiple model framework as well as several state of the art approaches to constructing an MMEKF. The key fact to note about this type of filter is that it is a hybrid filter, containing both a discrete component and a continuous component. When applied to appearance matching, the discrete element is the identity of the object and the continuous element is the pose of the object. MMEKFs vary in what is referred to in [97] as the cooperation strategy. It determines how the different models interact to find a single output. The cooperation strategy either selects the output of one or the other model or combines them in a kind of average. For appearance matching, the former strategy would be applicable to a library of several different objects while the latter

would be preferable in the case where the library is different configurations of the same object. An example of this case would be a spacecraft with solar panels at different angles of deployment.

7.2.2 Color-aided appearance matching

All of the investigation and results for this research used greyscale images. Appearance matching has been used with color images for object identification. Morioka and Hashimoto [98] used color histograms to train and appearance matching algorithm. Their algorithm was able to track and label objects across multiple cameras. Since histograms were used, this approach would not be able to perform attitude determination.

However, the multi-sensor framework could be used with a color camera, where each of the red, green, and blue channels represents a different sensor. Training the appearance matching with each channel and potentially an infrared camera as well would facilitate both object identification and attitude determination, the combination of which produces a full pose estimate as described above.

7.2.3 Quantity of eigenvalues

The relationship between the number of eigenpairs and percentage of image set variance captured was demonstrated in Section 3.5.1. However, the deeper question of how many eigenpairs are "enough" is more complex. Several factors affect the amount of variance in a training library, including numbers of orientations, lighting conditions, and objects; whether background randomization is implemented, and even the physical characteristics of the target objects. A comprehensive study of the relationship between all of these parameters to the amount of variance and performance of appearance matching in object identification and attitude determination would be a useful tool for the implementation of this method.

7.2.4 Virtual sensing

Accurately incorporating *a priori* information allows the improvement of both the object identification and pose estimation steps of the appearance matching algorithm. In particular, many of the high-error attitude determination results are due to the symmetric nature of the target object. Such cases would be identified and mitigated if previous state information is available. For example, if an attitude is returned that is 89 degrees away from the result one second ago, it is much more likely that such a result is due symmetricity, especially if the target was not viewed to have a high angular rate before.

One way to utilize previous state information is through a dynamic filter, but another option is presented by the multi-sensor framework described in this research. A "virtual" sensor may be constructed, which uses *a priori* object identification and state information to generate an estimate for the current object identity and state along with associated confidence values. Studies would need to be performance to compare the effectiveness of this approach versus a dynamic filter of some kind.

7.2.5 Adaptive learning

One logical extension of this research is its application to scenarios where the object is not well known beforehand. As presented the spacecraft visual navigation algorithm requires knowledge of the potential targets in order to generate the simulated images for appearance matching training. A hypothetical system could take a generic model for the target and then improve the image library as it approached, including more detail over time as knowledge is gained about the target. Ross et al. [99] propose a method of robust visual tracking that uses a sequential PCA method to update the eigenspace with additional data.

This feature is attractive, particularly for travel to interplanetary bodies which have not been well measured or mapped. However, including an adaptive algorithm presents some challenges. The first is that in order to train the appearance matching algorithm, the attitude of the object must be associated with each image. This corroboration cannot be done

very well if the knowledge of the object is imperfect. Additionally, the processing power required to solve the eigenanalysis is far greater than to do the projection and matching. The adaptive update would either take a long time or have to be offloaded to the ground for processing.

7.2.6 Better simulation fidelity and efficiency

Another potential avenue of future work is to improve the fidelity of the SISE. The current version is sufficient for the needs of this research, but better ray tracing or improved thermal simulation is something that could benefit future appearance matching algorithms which use more eigenvalues or greater resolution. As addressed in Section 4.3.1, a number of simplifying assumptions were made for the current version of the thermal simulation. If infrared appearance matching were being implemented for a real-world mission, more information would be known about the internal thermal properties of the target. Therefore, the fidelity of the infrared images could be improved.

In a similar way, the ray tracing implementation for this research did not account for reflection or refraction, which would have required the generation and handling of "child" rays for each ray cast. For the targets that were simulated for Chapter 6 this simplification was acceptable, but a different application could require a higher fidelity image simulation.

In addition, the adaptive learning method described in the previous section may need to simulate images of its own in order to update the eigenspaces. For this reason, another potential avenue of future research would be improving the efficiency of the image simulation. In particular, allowing it to generate simulated images rapidly on platforms not enabled with GPUs. RayChip, developed by SiliconArts, is a commercial chip designed to perform real-time ray tracing on an embedded system [100]. If it could be integrated with a flight or attitude computer on a spacecraft, then simulated images could potentially be created on the fly.

Figure 7.2: RayChip ray tracing chip [100]

7.3 Reflection

The results presented in this research show the advantages and potential of applying new techniques from different fields to spacecraft visual navigation. As on-board computational power and memory capacity improve for spacecraft, the work presented here serves as a possible model for the development of novel navigation techniques. Appearance matching in particular shows promise as an option for spacecraft missions to perform accurate object identification and pose estimation. The addition of background randomization and sensor fusion make this version of the technique more flexible and robust than previous implementations, an expansion that could be applied to terrestrial applications as well.

There are also several topics of future work in algorithm improvement, software development, and hardware implementation. Further research will be able to build on and utilize this research's contribution to spacecraft navigation in the same way that past work has benefited this dissertation.

REFERENCES

[1] H. Murase and S. K. Nayar, "Learning and recognition of 3d objects from appearance", in Proceedings of IEEE Workshop on Qualitative Vision, IEEE, 1993, pp. 39 –50.

[2] X. He, S. Yan, Y. Hu, P Niyogi, and H. J. Zhang, "Face recognition using laplacianfaces", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 3, pp. 328–340, 2005.

[3] H. Murase, "Online recognition of free-format japanese handwritings", in Pattern Recognition, 1988., 9th International Conference on, IEEE, 1988, pp. 1143–1147.

[4] S. Clark, Ten commercial earth-observing satellites launched aboard minotaur-c rocket, https://spaceflightnow.com/2017/10/31/ten-commercial-earth-observing-satellites-launched-aboard-minotaur-c-rocket/, Online. Accessed 11/05/17., Oct. 2017.

[5] Cyclone global navigation satellite system (cygnss), http://clasp-research.engin.umich.edu/missions/cygnss/docs/CYGNSS_FactSheet_October2014.pdf, Online. Accessed 11/04/2017., Oct. 2014.

[6] S. Bandyopadhyay, G. P. Subramanian, R. Foust, D. Morgan, S.-J. Chung, and F. Hadaegh, "A review of impending small satellite formation flying missions", in 53rd AIAA Aerospace Sciences Meeting, 2015, pp. 2015–1623.

[7] G. Roesler, Robotic servicing of geosynchronous satellites (rsgs), https://www.darpa.mil/program/robotic-servicing-of-geosynchronous-satellites, Online. Accessed 10/30/17.

[8] Darpa selects ssl as commercial partner for revolutionary goal of servicing satellites in geo, https://www.darpa.mil/news-events/2017-02-09, Online. Accessed 11/14/2017, Feb. 2017.

[9] Elsa-d | astroscale, http://astroscale.com/services/elsa-d, Online. Accessed 11/14/2017., 2016.

[10] J.-C. Liou and N. L. Johnson, "Risks in space from orbiting debris", in Science, 5759, vol. 311, Jan. 2006, pp. 340–341.

[11] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs, "A search engine for 3d models", ACM Transactions on Graphics, vol. 22, no. 1, pp. 83–105, 2003.

[12] D. G. Lowe, "Object recognition from local scale-invariant features", in Computer vision, 1999. The proceedings of the seventh IEEE international conference on, Ieee, vol. 2, 1999, pp. 1150–1157.

[13] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 24, pp. 509–522, 2002.

[14] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot", in IEEE/RSJ International Workshop on Intelligent Robots and Systems, IEEE, 1991, pp. 1442–1447.

[15] L. Walker and D. Spencer, "Automated proximity operations using image-based relative navigation", in 26th Annual USU/AIAA Conference on Small Satellites, vol. 8, 2012, pp. 1–12.

[16] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem", International journal of computer vision, vol. 81, no. 2, pp. 155–166, 2009.

[17] C. R. McBryde and E. G. Lightsey, "Spacecraft relative navigation using appearance matching and sensor fusion", in Proceedings of the 2017 International Technical Meeting of The Institute of Navigation, Jan. 2017, pp. 301–312.

[18] C. R. McBryde, A. E. Johnson, and E. G. Lightsey, "Spacecraft relative navigation using appearance matching", Acta Astronautica. Submitted, 2018.

[19] Blender.org - home of the blender project - free and open 3d creation software, `https://www.blender.org/`, Accessed: 11/4/2014.

[20] Pixar's renderman, `https : / / renderman . pixar . com / view / renderman`, Accessed: 11/4/2016.

[21] Nvidia advanced rendering: Nvidia mental ray, urlhttps://www.nvidia.com/en-us/design-visualization/solutions/rendering/product-updates/, Online. Accessed: 11/4/2016.

[22] C. Garnier, R. Collorec, J. Flifla, C. Mouclier, and F. Rousï¿œe, "Infrared sensor modeling for realistic thermal image synthesis", in

IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 6, IEEE, 1999, pp. 3513–3516.

[23] S. Bhatia and G. Lacy, "Infra-red sensor simulation", in I/ITSEC, 1999, pp. 1–8.

[24] J.-F. Shi, S. Ulrich, S. Ruel, and M. Anctil, "Uncooperative spacecraft estimation using an infrared camera during proximity operations", in AIAA SPACE, 2015, pp. 149–162.

[25] C. R. McBryde and E. G. Lightsey, "End-to-end testing of a dual use imaging sensor for small satellites", Journal of Small Satellites, vol. 5, no. 1, pp. 435–448, 2016.

[26] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales", Journal of applied statistics, vol. 21, no. 1-2, pp. 225–270, 1994.

[27] ——, "Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention", International Journal of Computer Vision, vol. 11, no. 3, pp. 283–318, 1993.

[28] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems", Computing, vol. 38, no. 4, pp. 325–340, 1987.

[29] J. Duchon, "Splines minimizing rotation-invariant semi-norms in sobolev spaces", in Constructive Theory of Functions of Several Variables, W. Schempp and K. Zeller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1977, pp. 85–100, ISBN: 978-3-540-37496-1.

[30] S. A. Nene, S. K. Nayar, and H. Murase, Columbia object image library (coil-20), `http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php`, Online. Accessed 11/1/2017.

[31] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves", IEEE Transactions on computers, vol. 100, no. 3, pp. 269–281, 1972.

[32] J. A. Christian and S. Cryan, "A survey of lidar technology and its use in spacecraft relative navigation", in Proceedings of the AIAA Guidance, Navigation, and Control Conference, 2013, pp. 19–22.

[33] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: A survey from 2010 to 2016", IPSJ Transactions on Computer Vision and Applications, vol. 9, no. 1, pp. 9–16, 2017.

[34] J. Han and B. Bhanu, "Fusion of color and infrared video for moving human dectection", Pattern Recognition, vol. 40, no. 6, pp. 1771–1784, 2007.

[35] J. Saeedi and K. Faez, "Infrared and visible image fusion using fuzzy logic and population-based optimization", Applied Soft, vol. 12, no. 3, pp. 1041–1054, 2012.

[36] J. Zhao and S. S. Cheung, "Human segmentation by geometrically fusing visible-light and thermal imageries", Multimedia Tools and Applications, vol. 73, no. 1, pp. 61–89, 2014.

[37] R. T. Howard, A. F. Heaton, R. M. Pinson, and C. K. Carrington, "Orbital express advanced video guidance sensor", in IEEE Aerospace Conference, IEEE, 2008, pp. 1–10.

[38] K. Williams, A. Taylor, B. Page, P. Wolff, B. Williams, D. Stanbridge, and J. McAdams, "Navigation for the messenger mission's first mercury encounter", in AIAA/AAS Astrodynamics Specialist Conference and Exhibit, pp. 6761–80.

[39] B. Naasz, J. V. Eepoel, S. Queen, C. M. Southward, and J. Hannah, "Flight results of the hst sm4 relative navigation sensor system", in 33rd ANNUAL AAS GUIDANCE AND CONTROL CONFERENCE, 2010, pp. 1–23.

[40] C. Harris, "Tracking with rigid models", in Active Vision, A. Blake and A. Yuille, Eds., Cambridge, MA, USA: MIT Press, 1993, ch. Tracking with Rigid Models, pp. 59–73, ISBN: 0-262-02351-2.

[41] Gamut 1080p hd tvi cctv bullet camera with 30m ir, 4-in-1, ahd: Spycameracctv.com, https://www.spycameracctv.com/spycamera/gamut-1080p-tvi-hd-cctv-camera-30m-night-vision-varifocal-lens, Online. Accessed 2017-10-09.

[42] Optical identification, http://www.balluff.com/en/de/products/machine-vision-and-optical-identification/optical-identification/, Online. Accessed 2017-10-9.

[43] Thermal infrared camera - artray.inc - usb camera, usb3.0 camera, ingaas camera, http://www.artray.us/thermo.html, Online. Accessed 2017-10-9.

[44] H. Murase, F Kimura, M. Yoshimura, and Y. Miyake, "An improvement of the auto-correlation matrix in pattern matching method and its application to handprinted hiragana", Trans. IECE, vol. 64, no. 3, pp. 276–283, 1981.

[45] M. Oren and S. K. Nayar, "Generalization of lambert's reflectance model", in Proc. of the 21st conference on computer graphics and interactive techniques, ACM, 1994, pp. 239–246.

[46] H. Murase and S. K. Nayar, "Illumination planning for object recognition using parametric eigenspaces", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 12, pp. 1219–1227, 1994.

[47] C. Eckart and G. Young, "A principal axis transformation for non-hermitian matrices", Bulletin of the American Mathematical Society, vol. 45, pp. 118–121, Feb. 1939.

[48] L. Mirsky, "Symmetric gauge functions and unitarily invariant norms", Quarterly Journal of Mathematics, vol. 11, pp. 50–59, Mar. 1960.

[49] B. Novak, Tech tower, `https : / / www . flickr . com / photos / brookenovak/22840887/`, O, Jun. 2005.

[50] S. A. Nene and S. K. Nayar, "Closest point search in high dimensions", in Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition, ser. CVPR '96, Washington, DC, USA: IEEE Computer Society, 1996, pp. 859–865, ISBN: 0-8186-7258-7.

[51] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces", in VLDB, vol. 98, 1998, pp. 194–205.

[52] R. Weber and S. Blott, "An approximation-based data structure for similarity-search", ESPRIT project HERMES (no. 9141), Tech. Rep. 24, Oct. 1997.

[53] V. Hyvönen, T. Pitkänen, S. Tasoulis, E. Jääsaari, R. Tuomainen, L. Wang, J. Corander, and T. Roos, "Fast nearest neighbor search through sparse random projections and voting", in Big Data (Big Data), 2016 IEEE International Conference on, IEEE, 2016, pp. 881–888.

[54] K. Karhunen, Über lineare methoden in der wahrscheinlichkeitsrechnung, ser. Annales Academiae scientiarum Fennicae. Series A. 1, Mathematica-physica. 1947.

[55] M. Loeve, "Fonctions alï¿œatoires de second order", in Processus Stochastiques et Movement Brownien, P. Levy, Ed., Paris: Hermann, 1948.

[56] D. D. Kosambi, "Statistics in function space", in D.D. Kosambi: Selected Works in Mathematics and Statistics, R. Ramaswamy, Ed. New Delhi: Springer India, 2016, pp. 115–123, ISBN: 978-81-322-3676-4.

[57] H. Hotelling, "Analysis of a complex of statistical variables into principal components.", Journal of educational psychology, vol. 24, no. 6, p. 417, 1933.

[58] ——, "Relations between two sets of variates", Biometrika, vol. 28, no. 3/4, pp. 321–377, 1936.

[59] A. Leonardis and H. Bischof, "Robust recognition using eigenimages", Computer Vision and Image Understanding, vol. 78, no. 1, pp. 99–118, 2000.

[60] M. Storer, P. M. Roth, M. Urschler, and H. Bischof, "Fast-robust pca", in Scandinavian Conference on Image Analysis, Springer, 2009, pp. 430–439.

[61] P. Bonnifait and G. Garcia, "Design and experimental validation of an odometric and goniometric localization system for outdoor robot vehicles", IEEE Transactions on robotics and automation, vol. 14, no. 4, pp. 541–548, 1998.

[62] R. L. Cook, L. Carpenter, and E. Catmull, "The reyes image rendering architecture", in ACM SIGGRAPH Computer Graphics, ACM, vol. 21, 1987, pp. 95–102.

[63] M. Seymour, Renderman: Under the (new) varnish, https://www.fxguide.com/featured/renderman-under-the-new-varnish/, Online. Accessed 2017-10-15., May 2015.

[64] N. Greene, M. Kass, and G. Miller, "Hierarchical z-buffer visibility", in Proc. of the 20th Ann. Conference on Computer Graphics and Interactive Techniques, ser. SIGGRAPH '93, New York, New York: ACM, 1993, pp. 231–238.

[65] Rasterization: A practical implementation (an overview of the rasterization algorithm), https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation, Online. Accessed 2017-10-9.

[66] A. S. Glassner, Ed., Ray tracing. Acadmeic Press, 1989.

[67] An overview of the ray-tracing rendering technique, https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-overview, Online. Accessed 2017-10-9.

[68] A. SpaceShop, Geographos | 3d resources, https://nasa3d.arc.nasa.gov/detail/geographos, Accessed: 10/27/2017, Nov. 2015.

[69] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm", in Computer graphics, ACM, vol. 21, 1987, pp. 163–169.

[70] T. Möller and B. Trumbore, "Fast, minimum storage ray/triangle intersection", in ACM SIGGRAPH 2005 Courses, ACM, 2005, pp. 21–28.

[71] D. Edwards, Radiation heat transfer. New York: Hemisphere Publishing Corporation, 1981.

[72] S. T. Hsu, Engineering heat transfer. Blacksburg, Virginia: D. Van Nostrand Company, Inc., 1962.

[73] J. Lekner, Theory of reflection, of electromagnetic and particle waves. Dordrecht, Netherlands: Springer, 1987.

[74] E. Schubert, Light Emitting Diodes.org, Available at: http://www.ecse.rpi.edu/~schubert/Light-Emitting-Diodes-dot-org (accessed October 7, 2015), 2006.

[75] B. Simonds, Physics and creating plausible materials, https://bensimonds.com/2010/08/27/plausiblematerials/, Online. Accessed 11/20/2017., Aug. 2010.

[76] J. Christian, "Optical navigation for a spacecraft in a planetary system", PhD thesis, The University of Texas at Austin, 2011.

[77] Sun glints off hubble's solar panels during sm3a, http://commons.wikimedia.org/wiki/File:Sun_glint_from_Hubble's_solar_panels.jpg, Online. Accessed 11/20/2017.

[78] L.-Q. Yan, M. Hašan, W. Jakob, J. Lawrence, S. Marschner, and R. Ramamoorthi, "Rendering glints on high-resolution normal-mapped specular surfaces", ACM Transactions on Graphics (TOG), vol. 33, no. 4, pp. 1–9, 2014.

[79] A. J. P. Theuwissen, Solid-state imaging with charge-coupled devices. Springer, 1995.

[80] M. Dawson-Haggerty, Trimesh, https://pypi.python.org/pypi/trimesh, Accessed: 10/15/2017, Sep. 2017.

[81] D. C. Brown, "Close-range camera calibration", PHOTOGRAMMETRIC ENGINEERING, vol. 37, no. 8, pp. 855–866, 1971.

[82] G. Stockman and L. G. Shapiro, Computer vision, 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001, ISBN: 0130307963.

[83] Ccd image sensor noise sources, https://www.uni-muenster.de/imperia/md/content/ziv/multimedia/downloads/kodak___noise_sources.pdf, Online. Accessed 3/8/2018, Aug. 2001.

[84] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler", in Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, ser. LLVM '15, Austin, Texas: ACM, 2015, 7:1–7:6, ISBN: 978-1-4503-4005-2.

[85] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", in Proceedings of Imaging Understanding Workshop, 1981, pp. 674–679.

[86] Opencv: Optical flow, https://docs.opencv.org/3.2.0/d7/d8b/tutorial_py_lucas_kanade.html, Online. Accessed 11/2/2017, Dec. 2016.

[87] N. Johnson, Roll-out solar array (rosa) jettisoned from space station, https://www.youtube.com/watch?v=jATBbjU4IyA, Online. Accessed 1/18/2018., Jun. 2017.

[88] S. Videos, [iss] hd video of progress 45 (m-13m) docking to iss, https://www.youtube.com/watch?v=_NPghRm48ok, Online. Accessed 1/18/2018., Nov. 2011.

[89] Atmospheres and planetary temperatures, https://www.acs.org/content/acs/en/climatescience/energybalance/planetarytemperatures.html, Online. Accessed 3/10/2018, 2018.

[90] Ruby 1.3m - ev76c661 - cmos image sensor - teledyne e2v, https://www.e2v.com/products/imaging/cmos-image-sensors/ev76c661/, Online. Accessed 11/20/2017., 2017.

[91] Usb3 vision camera with e2v / aptina sensors - mvbluefox3 - industrial image processing, https://www.matrix-vision.com/USB3-vision-camera-mvbluefox3.html, Online. Accessed 10/13/2017, Aug. 2017.

[92] Tau 2 lwir camera cores | flir systems, http://www.flir.com/cores/display/?id=54717, Online. Accessed 11/2/2017, 2017.

[93] J.-Y. Bouget, Camera calibration toolbox for matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/, Online. Accessed 4/15/2018, 2018.

[94] Juno spacecraft 1:180 for 3d printing by dustin970 - thingiverse, https://www.thingiverse.com/thing:1515076, Online. Accessed 4/10/2018, Apr. 2016.

[95] Ultimaker 3: Unrivaled print quality, https://ultimaker.com/en/products/ultimaker-3, 2018.

[96] H. D. Black, "A passive system for determining the attitude of a satellite", AIAA Journal, vol. 2, no. 7, pp. 1350–1351, 1964.

[97] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. part v: Multiple-model methods", IEEE Transactions on Aerospace and Electronic Systems, vol. 41, no. 4, pp. 1255–1321, Oct. 2005.

[98] K. Morioka and H. Hashimoto, "Color appearance based object identification in intelligent space", in Advanced Motion Control, 2004. AMC '04. The 8th IEEE International Workshop on, 2004, pp. 505–510.

[99] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking", International Journal of Computer Vision, vol. 77, no. 1, pp. 125–141, 2008.

[100] W.-C. Park, H.-J. Shin, B. Lee, H. Yoon, and T.-D. Han, "Raychip: Real-time ray-tracing chip for embedded applications", in Hot Chips 26 Symposium, Cupertino, CA: IEEE, Aug. 2014.

VITA

Christopher Ryan McBryde was born in Winter Park, Florida, on July 20, 1988, the 19th anniversary of the landing of Apollo 11. He attended elementary and middle school in Maitland, Florida, and graduated from Winter Park High School in 2006 as a valedictorian with his International Baccalaureate diploma. Christopher graduated *summa cum laude* from the University of Florida in 2010 with a Bachelor of Science degree in aerospace engineering. He began graduate school at the University of Texas at Austin under the advisement of Dr. Glenn Lightsey where he received his Master of Science in Engineering in aerospace in 2012. Christopher began his doctoral research at UT Austin with the support of the NASA Space Technology Research Fellowship during which he interned at Ames Research Center and the Jet Propulsion Laboratory in the summer. He transferred to the Georgia Institute of Technology in 2015 where he completed his PhD.