A MONOCULAR SLAM METHOD TO ESTIMATE RELATIVE POSE DURING SATELLITE PROXIMITY OPERATIONS

THESIS

Scott J. Kelly, Captain, USAF

AFIT-ENY-MS-15-M-219

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENY-MS-15-M-219

A MONOCULAR SLAM METHOD TO ESTIMATE RELATIVE POSE DURING SATELLITE PROXIMITY OPERATIONS

THESIS

Presented to the Faculty

Department of Aerospace Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Astronautical Engineering

Scott J. Kelly, B.S., Mechanical Engineering

Captain, USAF

March 9, 2015

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENY-MS-15-M-219

A MONOCULAR SLAM METHOD TO ESTIMATE RELATIVE POSE DURING SATELLITE PROXIMITY OPERATIONS

THESIS

Scott J. Kelly, B.S., Mechanical Engineering
Captain, USAF

Committee Membership:

Dr. Jonathan Black
Chair

Dr. Alan Jennings
Member

LtCol Daniel Doyle, PhD
Member

Abstract

Automated satellite proximity operations is an increasingly relevant area of mission operations for the US Air Force with potential to significantly enhance space situational awareness (SSA). Simultaneous localization and mapping (SLAM) is a computer vision method of constructing and updating a 3D map while keeping track of the location and orientation of the imaging agent inside the map. The main objective of this research effort is to design a monocular SLAM method customized for the space environment. The ultimate goal is to design a complete control system for satellite proximity operations and provide the ability to perform accurate motion and structure estimation on an unknown space object based on relative motion, leveraging this SLAM method. The method developed in this research will be implemented in an indoor proximity operations simulation laboratory. Development is constrained to a monocular system to minimize the required size, weight, and power (SWAP) and allow implementation on a spacecraft as small as a CubeSat. A run-time analysis is performed, showing near real-time operation. The method is verified by comparing SLAM results to truth vertical rotation data from a CubeSat air bearing testbed. An adjustable parameter sensitivity analysis is performed as well. This work enables control and testing of simulated proximity operations hardware in a laboratory environment. Additionally, this research lays the foundation for autonomous satellite proximity operations with unknown targets and minimal additional SWAP requirements, creating opportunities for numerous mission concepts not previously available.

Acknowledgements

I would like to thank of my friends, family, and faculty for their help and support during my thesis work. Thank you Dr. Black for your guidance and motivation while I worked to create these results. Thank you Dr. Jennings for going above and beyond with your time commitment as only a committee member. Thank you mom for all of the time spent proof reading. Thank you to my peers for contributing to the productivity of brainstorming sessions. Finally, a big thank you to my girlfriend and my dog for the continued support while withstanding a moderate level of neglect.

Scott J. Kelly

Table of Contents

Page

List of Figures

List of Tables

List of Abbreviations

Abbreviation		Page
SWAP	size, weight, and power	1
NASA	National Aeronautics and Space Administration	1
AFRL	Air Force Research Laboratory	2
DARPA	Defense Advanced Research Projects Agency	2
ANGELS	Automated Navigation and Guidance Experiment for Local Space	2
AVGS	Advanced Video Guidance Sensor	2
DART	Demonstration for Autonomous Rendezvous Technology	2
ISS	International Space Station	2
GEODETICA	General Electro-Optical DEtection, Tracking, Identification, and Characterization Application	2
AFIT	Air Force Institute of Technology	2
RSO	resident space object	3
RLS	Rendezvous LiDAR Sensor	4
CV	computer vision	4
SLAM	simultaneous localization and mapping	6
CCD	charge-coupled device	9
PnP	perspective-n-point	15
SVD	singular value decomposition	19
SIFT	Scale Invariant Feature Transform	23
SURF	Speeded-Up Robust Features	23
FAST	Features from Accelerated Segment Test	24
KLT	Kanade-Lucas-Tomasi	26

Abbreviation		Page
BRIEF	Binary Robust Independent Elementary Feature	28
SFM	structure from motion	31
GPU	graphics processing unit	32
PTAM	Parallel Tracking and Mapping	33
HOMER	Holonomic Omni-direction Motion Emulation Robot	38
LEO	low Earth orbit	39
PRISMA	Prototype Research Instruments and Space Mission technology Advancement	47
ASTRO	Autonomous Space Transport Robotic Operations	65
NEXTSat	Next Generation Satellite and Commodities Spacecraft	65
RMS	root mean squared	77

A MONOCULAR SLAM METHOD TO ESTIMATE RELATIVE POSE DURING SATELLITE PROXIMITY OPERATIONS

I. Introduction

1.1 Background

The ability of a satellite to rendezvous with another satellite provides the Air Force with an immense set of capabilities. Performing maneuvers near another satellite, known as proximity operations or *proxops*, allows the user to collect data and interact with the target satellite in many ways that are impossible from the surface of the Earth. Reliable proxops technologies enable numerous mission concepts, some of which include inspection and maintenance satellites, refueling, and debris removal. The ability to perform these operations autonomously further enhances mission capabilities and minimizes the required on-orbit size, weight, and power (SWAP).

However, a satellite performing proxops with an unknown or uncooperative space object must be able to, at a minimum, estimate its relative position with respect to the target object to be able to successfully perform relative navigation. Additionally, a spacecraft must be able to accurately estimate the target object's orbit to develop a maneuver strategy. As the distance between the two satellites decreases, knowledge of the target spacecraft's orientation and dimensions may be required as well to avoid a collision, especially if docking or other physical interaction is required.

In the early years of space exploration, spacecraft rendezvous was performed between two manned spacecraft under the command of an astronaut [15]. In more recent years, missions from the National Aeronautics and Space Administration (NASA), the

Air Force Research Laboratory (AFRL), the Defense Advanced Research Projects Agency (DARPA), SpaceX, and Orbital Sciences Corporation aim to advance prox-ops technology and demonstrate capability to rendezvous with various spacecraft, each with different levels of autonomous operation and sensor data from the target satellite (hence a cooperative target).

AFRL's Automated Rendezvous System uses an active LiDAR sensor on board XSS-11 (eXperimental Satellite System 11), which is matched to a known model of the spacecraft to estimate the relative pose between the two spacecraft [1]. AFRL more recently launched the Automated Navigation and Guidance Experiment for Local Space (ANGELS) satellite which hosts a sensor payload to evaluate techniques for detecting, tracking, and characterizing space objects [2]. DARPA utilizes a system they developed called the Advanced Video Guidance Sensor (AVGS) on board their Orbital Express spacecraft and the Demonstration for Autonomous Rendezvous Technology (DART) spacecraft, which irradiates retro-reflectors of a known orientation with a laser to solve the relative pose problem [11]. The SpaceX Dragon capsule and Cygnus by Orbital Sciences Corporation both send unmanned resupply capsules to the International Space Station (ISS) that perform autonomous rendezvous (also based on a known model of the target) until within reach of a robotic arm on the ISS.

1.1.1 The GEODETICA Tool.

The General Electro-Optical DEtection, Tracking, Identification, and Characterization Application (GEODETICA) is a tool being developed by the Air Force Institute of Technology (AFIT), Virginia Tech, and AFRL in order to provide a software suite for processing unresolved space-based imagery. Currently, the tool detects and tracks objects in a continuous stream of unresolved imagery (long range images in which target objects and stars appear as points or streaks) [40]. With the track point

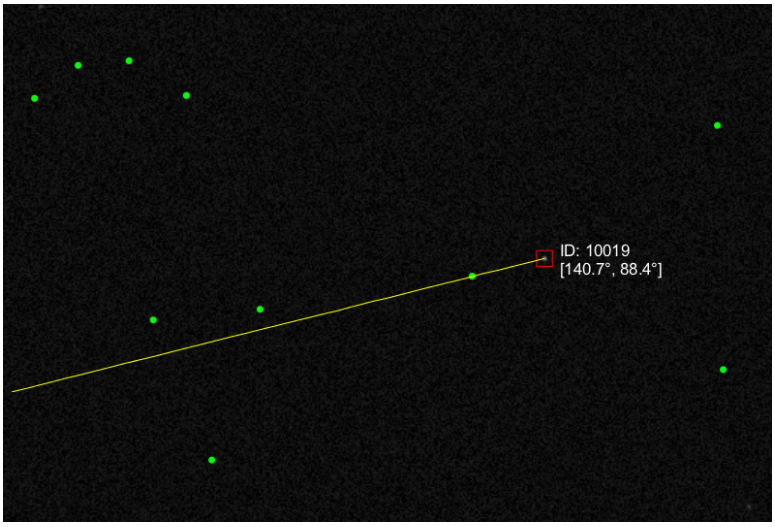


Figure 1. Example RSO Discrimination and Tracking in a Simulated Scenario [40].

information, the tool can perform attitude determination, angular rate estimation, and resident space object (RSO) detection. Figure 1 shows a simulated RSO discrimination and tracking scenario. The concept for the future of GEODETICA is a complete end-to-end tool to perform the image processing and CV algorithms for RSO detection, long distance rendezvous, and close range proxops with an uncooperative space object, and provide the information for the autonomous control of the spacecraft throughout the process.

1.2 Research Focus

1.2.1 Motivation to Use Computer Vision for Proxops.

The proxops methods listed in Section 1.1 have limitations. A LiDAR sensor for range finding requires an additional sub-system with additional hardware. Size, weight, and power each come at a very expensive premium in space. AVGS alone weighs approximately 8 kilograms, occupies slightly more than 10 liters of volume, and draws 30 watts of power during tracking mode [11]. Additionally, the Rendezvous

LiDAR Sensor (RLS) and methods based on fiducial markings both require detailed knowledge about the target object. These methods may provide less uncertainty for a single view of the target, but severely limit the capabilities of the proxops satellite if the mission requires approaching an unknown or uncooperative object.

Designing an image-based computer vision (CV) method to perform these required tasks that only uses a camera and a CPU (and very sophisticated software) would greatly minimize the additional sub-system requirements on the spacecraft. Most spacecraft already have star trackers, which use dedicated CPUs to perform stellar rectification with a catalog for attitude determination. Suddenly, with a dual purpose image processing sub-system that performs attitude determination and CV-based proxops navigation, the required SWAP to perform a proxops mission may be much smaller than previously imagined (possibly no additional hardware).

Once within range to acquire resolved imagery, it may also be possible to construct a model of the target spacecraft using a CV-based proxops system, using multiple view geometry methods [16]. This type of method removes the requirement for *a priori* knowledge of the shape and structure of the spacecraft in question. This method further expands the potential mission areas into operations near uncooperative targets, which may include sequential rendezvous with space debris for removal, or operations with a damaged or unresponsive spacecraft for information gathering purposes.

1.2.2 Proxops From a CubeSat.

CubeSats are becoming increasingly popular as alternatives to much larger spacecraft [42]. With a smaller and lighter platform, flexible launch options become available, making CubeSats a viable and affordable space platform for universities. Advances in technology are allowing CubeSat designers to fit an incredible amount of mission capability into very small spaces as well. Figure 2 shows one unique method

Figure 2. A Unique Way to Launch CubeSats: From the ISS [31].

for deployment, launching three 1U CubeSats from the ISS [31].

Two-camera CubeSat star trackers that occupy 1/2U to 1U are commercially manufactured as mission attitude knowledge requirements increase [6]. Developing CV algorithms and implementing them on a star tracker-like platform seems to be the most feasible implementation of an autonomous proxops system on a CubeSat.

One popular concept for proxops satellites is the inspector/maintenance satellite. An inspector satellite can gather valuable data from space about its target satellite, potentially providing the right information to extend the target satellite's time on-orbit (therefore decreasing the cost per year). To make the inspector satellite a feasible idea, it needs to be a cheaper alternative to replacing the original satellite a few years early. Flexible and affordable launch options provide a significant motivation behind implementing a CV-based proxops method on a CubeSat. Additionally, sharing a ride with a larger spacecraft automatically places the inspector in the same orbit as its target and removes long range rendezvous from the list of required capabilities.

Spatial information is capable of being extracted from a stereo rig of two cameras with known relative pose (a fixed relative rotation and translation). However,

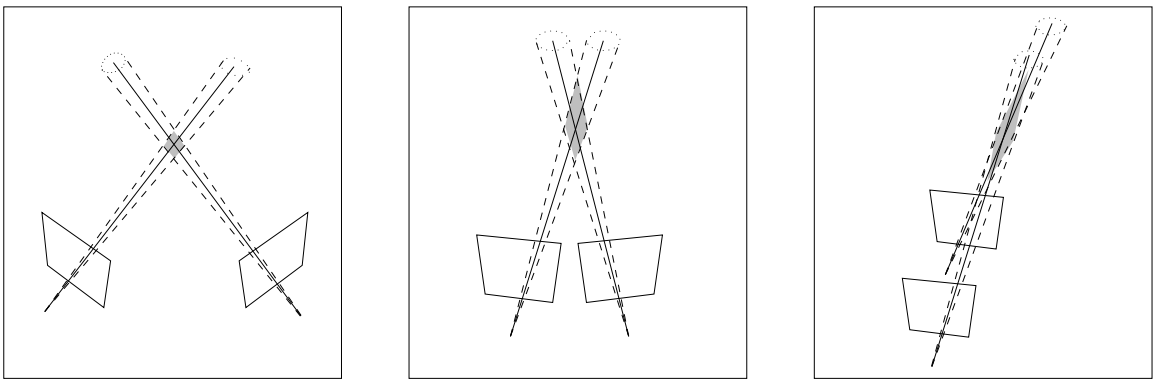


Figure 3. Increased Uncertainty As Relative Camera Separation Decreases [16].

accuracy of the spatial information decreases as the distance from the target increases relative to the camera separation. As the distance from the target continues to increase, eventually no additional information providing a second perspective is provided, shown by Figure 3.

Since CubeSats have very little linear dimension to provide separation between cameras, and distances in the space environment can be quite large, no additional useful information is gained by utilizing a stereo camera setup until the target is in extremely close proximity. The small relative space places an additional constraint on the method to be implemented: the algorithms developed herein are to be calculated from the motion of a single camera, or a monocular system. Estimating camera pose from corresponding features between multiple views from one camera, projecting the tracked features to a 3D map, and localizing the camera in the map is known as simultaneous localization and mapping (SLAM).

1.3 Goals

The main goal of this research is to design a monocular SLAM method for implementation in an indoor proxops simulation laboratory and eventual integration with the GEODETICA tool. The main product of this method will be relative pose

(position and orientation) and 3D structure estimates through relative motion with a target object. The SLAM method will be robust enough to handle the specifics of the space environment, including reflective surfaces, dynamic backgrounds, and changes in lighting. Key considerations to be addressed are the accuracy of the resulting structure and motion estimates in comparison to truth data. Near real-time operation is required as well to eventually provide state information for a hardware control system.

Another important aspect of this research is the ease of interpretation, customization, and testing. It is very important that the code is written in an easily understood format, is well commented, and ties directly to the explanation provided by this thesis. Successors to this research will use this software to perform testing, customization, and inevitably optimize this implementation. Writing the method developed here in a manner that promotes efficiency and rapid prototyping for its successors ensures it plays a part in meeting the larger goal of developing a space-rated monocular SLAM proxops system.

1.4 Document Layout

Chapter II provides a literature review of CV topics and algorithms used in this research, such as point tracking methods, multiple view geometry, pose estimation, projecting 2D point matches to 3D space, bundle adjustment, and a survey of other researchers performing monocular SLAM. Chapter III delves into the specific methodology that ties the chosen CV algorithms together to create a product that performs structure and motion estimation. Chapter IV presents SLAM results on space related video sequences, as well as an accuracy analysis, a processing time study, and a parameter sensitivity analysis. Finally, Chapter V presents the conclusions of this research, recommendations for future work, and the summary.

II. Literature Review

2.1 Relevant Theory

This section provides an overview of relevant theory necessary to design a monocular SLAM system for use in the space environment. It begins with a description of the multiple view geometry theory[1] that leverages changes in perspective of multiple points on an object from different views, including epipolar geometry, pose estimation, and methods to estimate spatial information. Three dimensional information can only be extracted from a series of images if there are accurate point correspondences on an object in multiple images throughout relative motion between the camera and the object. The next section will explore methods of obtaining 2D point correspondences throughout a series of frames, while keeping in mind that it needs to occur near real-time (as close to 30 frames per second as possible). Finally, the theory behind bundle adjustment is discussed, which is a method of reducing the reprojection error to refine estimated 3D motion and 3D point projections from multiple 2D images of each point.

2.1.1 The Pinhole Camera Model.

The pinhole camera model defines the geometric relationship between a 3D scene and a 2D image in an ideal camera with no lens, where the camera aperture is defined by a single point. Using the pinhole camera model, the 3D scene is projected on to a 2D plane at a distance f from the camera center, where f is the camera's focal length. A point in space is mapped on to the image plane at the intersection of a

[1]Richard Hartley and Andrew Zisserman provide what may be the industry standard for a collection of theory behind geometry from multiple perspective views in their text *Multiple View Geometry in Computer Vision* [16], as well as descriptions of numerous algorithms. This text is frequently cited during the review of applicable theory, especially through the assistance of their figures, to accompany a hopefully simpler and more concise explanation.

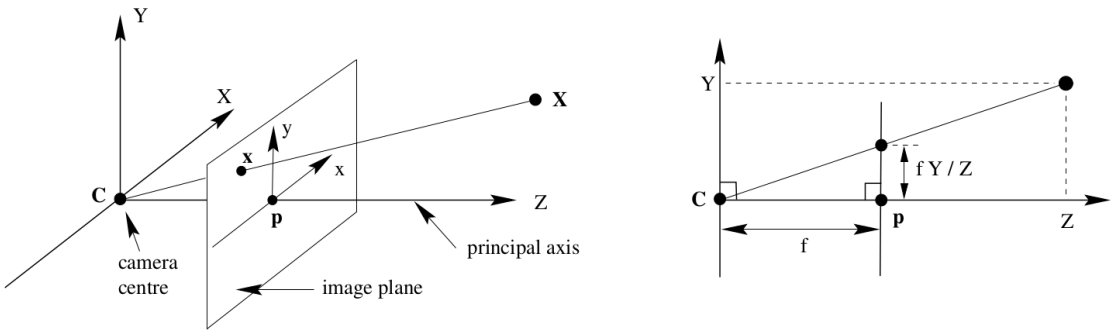


Figure 4. The Pinhole Camera Model [16].

line from the camera center to the point's 3D location (X, Y, Z) and the image plane. Figure 4 shows that through similar triangles, the 3D point at (X, Y, Z) is mapped to the 2D image plane to the coordinates $(fX/Z, fY/Z)$. The pinhole camera model is a popular approach in computer vision due to its simplicity. However, since it does not incorporate image distortions, camera calibration is required to account for deviations from the model.

Camera Calibration.

Real-world cameras do not perform the ideal projection described by the pinhole camera model. Lenses are used which introduce nonlinear distortions in imagery. Short focal length cameras and low-cost cameras compound these distortions. Distortions internal to the camera, or the camera's intrinsic parameters, are accounted for through the use of a camera calibration matrix $\mathbf{K}$. Equation 1 shows the general form of the calibration matrix of a charge-coupled device (CCD) camera, where f is the camera's focal length (in mm), s_x and s_y are the ratio of physical size of the imaging sensor per number of pixels (mm/pixel) in the x and y directions on the image plane, and (x_0, y_0) is the location of the camera's principal point, or image center, relative to the center of the image plane. A skew parameter is included as well, where Θ represents the angle between the x and y axes, which is usually $\pi/2$.

Additional parameters defining radial and tangential distortion can be included as well, but it is often not necessary to use them to their full extent with standard field of view cameras (non wide-angle lens).

$$\mathbf{K} = \begin{bmatrix} f/s_x & f/s_x \cot\Theta & x_0 \\ 0 & f/s_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Calibration of the camera's internal parameters is typically performed experimentally through the use of a calibration checkerboard. Zhang describes the theory behind this process [49]. Numerous packages exist to automate the procedure: a well-known toolbox for MATLAB by Bouget [8], a MathWorks implementation in the MATLAB Computer Vision System Toolbox [30], as well as an implementation of Bouget's algorithm in the open source computer vision library, OpenCV [9].

2.1.2 Multiple View Geometry.

As shown by the pinhole camera, a single image from a camera takes the 3D world and projects it into 2D space on the image plane. Unfortunately, during this process, all the important depth information is lost. At least one more perspective is required to recover depth information. Human eyes work in this manner: with two perspectives taken from a fixed distance apart (stereo vision), humans are able to perceive depth because each eye receives a slightly different view of the world (or knowledge of an object's size).

Suppose the world coordinate frame is defined by the left eye (zero rotation and translation). From the center of the right eye, in the right eye's coordinate frame, the origin of the world coordinate frame (and location of the left eye) is defined by the vector $\mathbf{t}_{\mathbf{R}}^{\mathbf{R/W}}$. The subscript on this notation signifies the vector's coordinate

frame: the right eye's coordinate frame. The superscript signifies the vector direction: pointing from the center of the right eye to origin of the world coordinate frame. The standard notation that follows will represent all quantities in a camera-centered coordinate frame unless otherwise stated. The shorthand version of this vector and the notation implemented is $\mathbf{t_R}$, which identifies the location of the world origin in the right eye's coordinate frame, with the vector originating from the right eye.

Orientation is defined in a camera-centric format as well. A change in orientation of the right eye with respect to the world coordinate frame is represented by the 3 x 3 rotation matrix $\mathbf{R_{RW}}$, which rotates vectors from the world frame to the right eye's frame. Equation 2 shows the calculation for the location of the right eye in the world coordinate system, $\mathbf{n_W^R}$.

$$\mathbf{n_W^R} = -[\mathbf{R_{RW}}]^{-1}\mathbf{t_R} \tag{2}$$

It is possible to construct a matrix (known as the camera matrix, $\mathbf{P}$) that identifies the location and orientation (pose) of each eye with respect to the world frame. The result is a 3 x 4 matrix that is a concatenation of the camera-centric rotation matrix and translation vector, with respect to the world coordinate system, shown by Equation 3. The left eye is represented in Equation 3 by $\mathbf{P_L}$ and the right eye by $\mathbf{P_R}$, where $\mathbf{K_L}$ and $\mathbf{K_R}$ are each cameras's respective intrinsic calibration matrices from Equation 1. Since the world coordinate system is aligned with the left eye, no rotation or translation is required. This is represented in Equation 3 by $\mathbf{I}$, the 3 x 3 identity matrix, and $\mathbf{0}$, a 3 x 1 vector of zeros.

$$\mathbf{P_L} = \mathbf{K_L}\,[\,\mathbf{I}\,|\,\mathbf{0}\,] \qquad \mathbf{P_R} = \mathbf{K_R}\,[\,\mathbf{R_{RW}}\,|\,\mathbf{t_R}\,] \tag{3}$$

Using $\mathbf{P}$, the 2D projection of a 3D point on the image plane for camera $\mathbf{n}$ occurs

according to Equation 4, where s is an unknown scale factor. In Equation 4, the lowercase $\mathbf{x}$ on the left side represents the 2D location of a point on the image, while the uppercase $\mathbf{X}$ on the right side represents the same point in 3D space.

$$s\,\mathbf{x}_n = \mathbf{P_n}\mathbf{X} \tag{4}$$

Equation 5 shows a summary of all of the quantities assembled in the 2D projection operation from Equations 1, 3, and 4. The quantities r_{mn} and t_{m} represent the individual elements of the appropriate rotation matrix and translation vector. The lowercase quantities x and y represent the image coordinates of point $\mathbf{x}$. The uppercase quantities X, Y, and Z represent the spatial coordinates of point $\mathbf{X}$.

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f/s_x & f/s_x \cot\Theta & x_0 \\ 0 & f/s_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

The Epipolar Plane.

As a camera moves in 3D and gains additional information through a second view (like your eyes), additional depth information about the 3D scene becomes available. However, unlike your eyes, the location of the camera which captures the second image is unknown. In order to gather any information about an object's 3D location through triangulation, the pose of the second camera must be known. The pose of the second camera is estimated through the use of epipolar geometry.

Epipolar geometry describes the relationship between two perspective views of the same 3D object. The epipolar plane is the plane passing through three points: the two camera centers and a corresponding point in each view. Figure 5 shows the

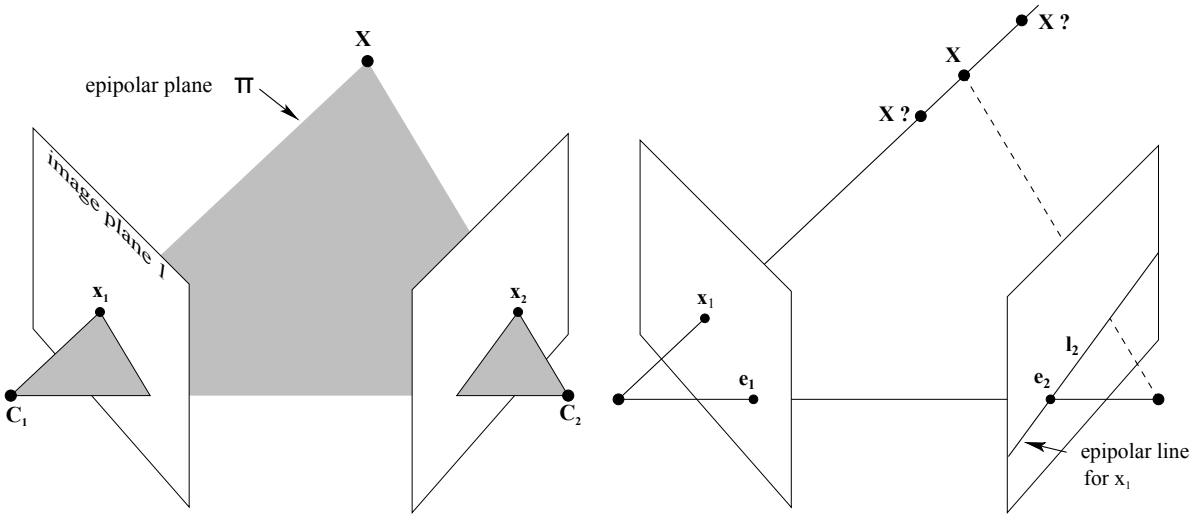


Figure 5. The Epipolar Plane. Depth Information is Revealed From Multiple Views [16].

epipolar plane. When observing the 3D point $\mathbf{X}$ from $\mathbf{C_1}$, all that is known is the 2D location of that point in the image, represented by $\mathbf{x}_1$. The 3D location could exist anywhere on the line $\mathbf{C_1} - \mathbf{X}$. However, when a second view is incorporated from camera $\mathbf{C_2}$, the exact point on that line can be identified, revealing the depth from camera $\mathbf{C_1}$. This concept leads to an important realization: a corresponding point in one image must lie on a particular line in the second image. The line that the corresponding point $\mathbf{x}_2$ must exist on is the intersection of epipolar plane with the second image plane, and is called an epipolar line, $\mathbf{l}_2$ in Figure 5.

The line connecting the base of the triangle from $\mathbf{C_1} - \mathbf{C_2}$ is called the baseline. At the intersection of the baseline with each image plane is each image's respective epipole, $\mathbf{e}_1$ and $\mathbf{e}_2$. Since the baseline is shared by any epipolar plane formed between two camera locations, only one point in the first image is needed to determine the epipolar line in the second image, on which the corresponding point must be found. If the images from $\mathbf{C_1}$ and $\mathbf{C_2}$ are not taken at the same time, one important assumption when applying these geometric relationships is that the scene must be rigid and non-changing.

The Essential and Fundamental Matrices.

Suppose three vectors are created along each edge of the triangle forming the epipolar plane, in the frame of camera $\mathbf{C_2}$. The first vector points from $\mathbf{C_2}$ to $\mathbf{x}_2$, $\mathbf{x}_2$. The second vector points from from $\mathbf{C_2}$ to $\mathbf{C_1}$ and is equivalent to the relative translation between the two cameras, $\mathbf{t_2}$. The third vector points from $\mathbf{C_1}$ to $\mathbf{x_1}$, translated to the $\mathbf{C_2}$ frame. As a reminder, the notation for the rotation matrices is camera-centered, so $\mathbf{R_2}$ brings vectors from $\mathbf{C_1}$ to $\mathbf{C_2}$. The vector pointing from $\mathbf{C_1}$ to $\mathbf{x}_1$ is then equivalent to $\mathbf{R_2 x}_1$. Since all three vectors are coplanar, Equation 6 is true (using homogeneous image coordinates) [25],

$$\mathbf{x}_2^{\mathrm{T}} \left(\mathbf{t_2} \times \mathbf{R_2 x}_1 \right) = \mathbf{x}_2^{\mathrm{T}} \left([\mathbf{t_2}]_{\mathrm{x}} \mathbf{R_2 x}_1 \right) = 0 \tag{6}$$

where the cross product is expressed as the product of the skew-symmetric matrix, $[\mathbf{t_2}]_{\mathrm{x}}$. Substituting $\mathbf{E} = [\mathbf{t_2}]_{\mathrm{x}} \mathbf{R_2}$, the essential matrix, yields Equation 7 (the epipolar constraint equation).

$$\mathbf{x}_2^{\mathrm{T}} \mathbf{E x}_1 = 0 \tag{7}$$

The essential matrix is the algebraic representation of the epipolar geometry for a known camera calibration, constraining a point in one image to a corresponding epipolar line in a second image (through a change in camera pose, and at a minimum, a non-zero translation vector; otherwise no additional perspective information is gathered). It is possible to convert the essential matrix from the camera coordinate system to image points in pixels using the inverse of the camera calibration matrix. Equation 8 shows the essential matrix rewritten in image coordinates, known as the fundamental matrix, $\mathbf{F}$ [14].[2]

$$\mathbf{F} = \mathbf{K_2}^{-\mathrm{T}}\mathbf{E}\mathbf{K_1}^{-1} = \mathbf{K_2}^{-\mathrm{T}}[\mathbf{t_2}]_{\mathrm{x}}\mathbf{R_2}\mathbf{K_1}^{-1} \tag{8}$$

Equation 9 shows the epipolar constraint equation rewritten with the fundamental matrix, giving the relationship between corresponding image points in pixel coordinates.

$$\mathbf{x}_2^{\mathrm{T}}\mathbf{F}\mathbf{x}_1 = 0 \tag{9}$$

2.1.3 Pose Estimation.

The motivation behind explaining the fundamental matrix is that it can be derived experimentally through multiple point correspondences in different images. Finding the fundamental matrix reveals the essential matrix if the camera calibration parameters are known. Then, the essential matrix is a combination of the relative rotation and translation between camera locations, $\mathbf{E} = [\mathbf{t}]_{\mathrm{x}}\mathbf{R}$, and can be decomposed to estimate the relative pose between cameras. This section first includes a review of commonly used algorithms to estimate the fundamental matrix from point correspondences, followed by a description of an iterative method to estimate a parameter in a data set that contains outliers called random sampling and consensus (RANSAC). A method of triangulating 2D point matches to a 3D location is described. This section concludes with a description of a method to estimate pose from a set of point correspondences with known 3D locations, called the perspective-n-point (PnP) problem.

[2]The fundamental and essential matrices have many different properties and uses not included here. Many nuances exist regarding obtaining optimal camera motion and degenerate cases that are not explained. This document seeks to reach a balance between including the relevant information without omitting important details. The bibliography includes numerous sources with additional information.

Algorithms to Estimate the Fundamental and Essential Matrices.

The simplest calculation of the fundamental matrix is the Normalized 8-point Algorithm [25]. From eight point matches in two images, the fundamental matrix is found by constructing the epipolar constraint equation for each, which is shown by Equations 9 and 10. The resulting linear equations formed are represented by Equation 11, and are followed by a least-squares solution. Only eight points are required as the fundamental matrix is a 3 x 3 matrix determined up to an arbitrary scale factor, which yields one constraint:

$$\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0 \tag{10}$$

and can be rearranged to represent n linear equations from n sets of point matches:

$$\begin{bmatrix} x_1'x_1 & x_1'y_1 & x_1' & y_1'x_1 & y_1'y_1 & y_1' & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n'x_n & x_n'y_n & x_n' & y_n'x_n & y_n'y_n & y_n' & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = 0 \tag{11}$$

Another algorithm exists for calculating the fundamental matrix that enforces an additional constraint: the fundamental matrix must be a singular matrix of rank two. The result is seven degrees of freedom and the seven-point algorithm, which is the

minimum case for calculating the fundamental matrix, but introduces a nonlinear constraint equation. Additional details are available [16].

The essential matrix, on the other hand, only has five degrees of freedom. Three degrees of freedom are from the rotation matrix, $\mathbf{R}$, three are from the translation vector, $\mathbf{t}$, and the scale constraint is applied again, reducing the number of degrees of freedom by one. A fairly complicated five-point algorithm exists for estimation of the essential matrix by Nister [32]. Nister shows that the five-point method clearly outperforms the other methods for sideways motion and the eight-point method is best for forward motion. Another drawback of the five-point method is it requires that the camera intrinsics be known, while the seven-point and eight-point methods can operate independently of this knowledge. An implementation of Nister's algorithm is included in OpenCV 3.0 [35].

RANSAC.

Typically, more point matches will exist than are required by the estimation algorithm. One of the most difficult aspects of implementing a CV routine is gracefully implementing a method to select the best points. Anomalies affect even the best point trackers. Image noise occurs as well. A common method implemented to remove outliers is RANSAC. Consider developing a solution to a problem based on a set of data points that contains outliers. Using a least-squares type solution, outlying points will skew the results. RANSAC, on the other hand, randomly samples groups of test points, computes a solution based on the reduced set, and estimates the total error between the remaining points and the solution based on the minimized set. After a number of iterations, the returned solution is the set that has the least overall error and the outliers are the points which deviate from that solution greater than a threshold value. In this manner, identified outliers do not affect the solution.

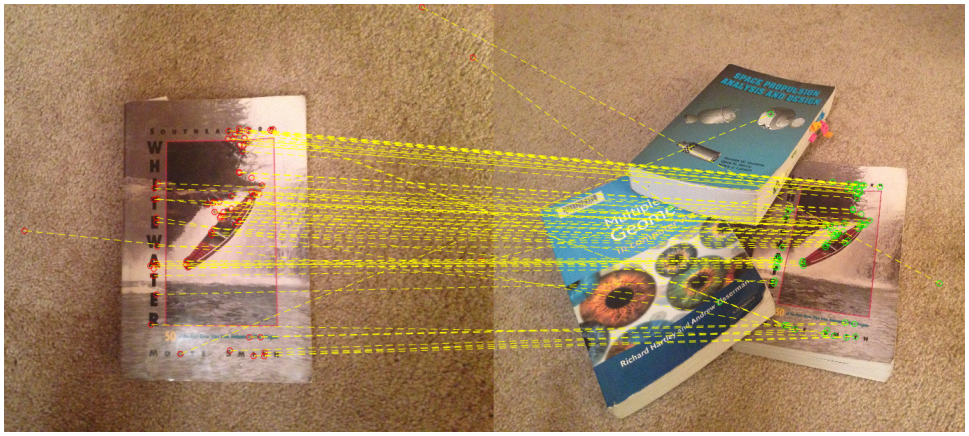


Figure 6. Initial Set of Point Correspondences (Credit: S. Kelly).

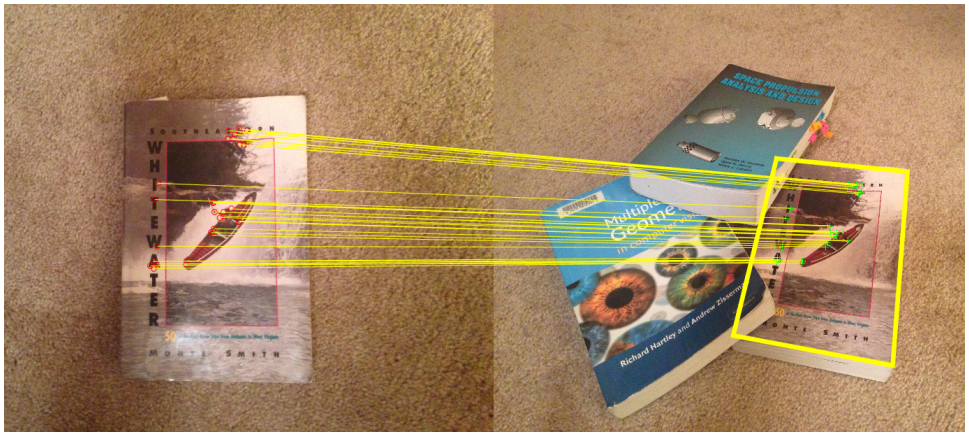


Figure 7. Point Correspondence Inliers After RANSAC (Credit: S. Kelly).

Figures 6 and 7 show RANSAC applied to a series of image point matches. Figure 6 shows initial point matches, with some outliers present. Figure 7 shows the remaining points after RANSAC has selected the least-error solution and removed outliers that don't meet a certain threshold.

RANSAC has advantages and disadvantages. In a group of data points that contains outliers, a big advantage of RANSAC is its ability to perform robust estimation of model parameters. However, approximately 50% of the points need to be inliers to converge on an accurate solution. Depending on the confidence level required, there is no upper bound on the time it may take to determine a solution. Limiting the

number of iterations may result in a solution that is less accurate than the least-squares solution of all available points. To obtain timely and accurate results, the points provided to RANSAC need to be carefully selected while using an appropriate number of iterations. MATLAB includes a RANSAC function implementation in the Computer Vision Systems Toolbox with the number of iterations and error threshold value as input options.

Decomposing the Essential Matrix to Estimate Pose.

Since the essential matrix confines points to exist on a plane, it is a singular matrix of rank 2. More specifically, a 3 x 3 matrix is an essential matrix if and only if two of its singular values are equal, and the third is zero. Using a method called singular value decomposition (SVD), it is possible to extract the translation and rotation information shown in Figure 8 from the essential matrix. The specifics of this method can be found in [16].

However, as a result of essential matrix decomposition, an ambiguity arises for the direction of rotation and translation. The result is four possible combinations, meaning four potential camera poses (a twisted pair). Using a test point, or a series of test points from point matches, it is trivial to determine which of the four relative pose combinations is correct, as only one will exist in which the test points occur in front of both cameras. Figure 9 shows two perspective views of this phenomenon for clarity. The original camera pose is represented by the green camera, pointing toward the projected points. The other four cameras represent the twisted pair, with the correct pose in blue, and the other three incorrect solutions in red. Notice the blue camera is pointing toward the points, while the other three cameras are not.

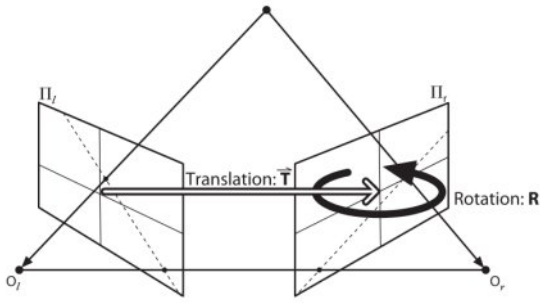


Figure 8. Relative Rotation and Translation From Essential Matrix Decomposition [10].

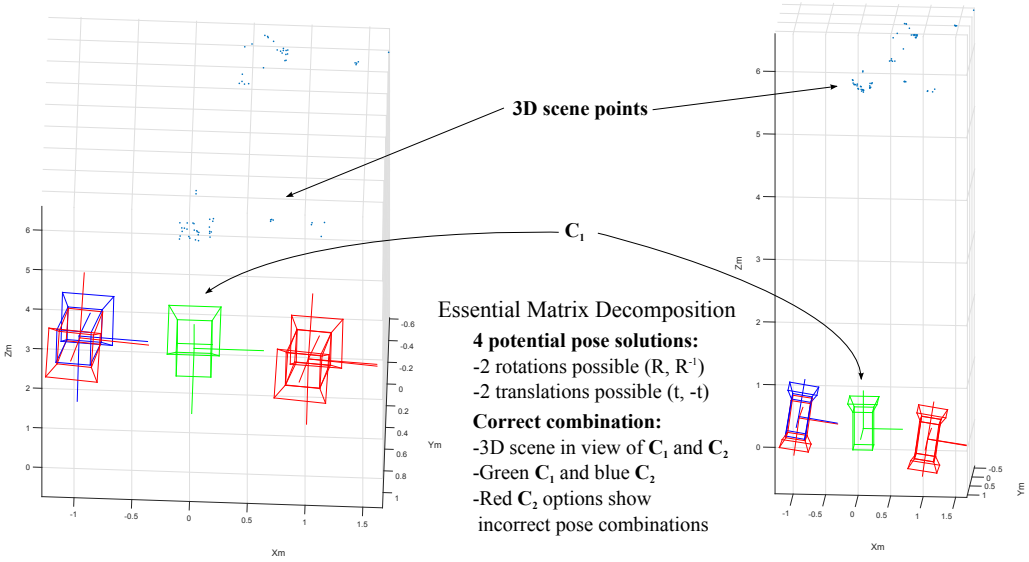


Figure 9. Decomposing the Essential Matrix: Two Views of the Twisted Pair Ambiguity.

Triangulation.

Triangulation is the process of using two known camera poses, $\mathbf{P_1}$ and $\mathbf{P_2}$, and a point correspondence in each image, $\mathbf{x}_1$ and $\mathbf{x}_2$ to find the point's actual location in 3D space, $\mathbf{X}$. Geometrically, triangulation is the process of finding the location of the intersection of the epipolar lines. A linear method exists to perform triangulation. Working in reverse, the 2D point location in each image is found from the camera matrix and the 3D point location through a 2D projection on to a surface. For each image projection, a 3D point is projected on the image plane using Equation 4. Using two images of the same point (Figure 8), $\mathbf{x}_1 = \mathbf{P_1 X}$ and $\mathbf{x}_2 = \mathbf{P_2 X}$. Since the quantities on each side of the equation are equal, a cross product between them equates to zero. For the first image, $\mathbf{x}_1 \times (\mathbf{P_1 X}) = 0$. The results of carrying out the cross product are shown in Equations 12 through 14, where $\mathbf{p_1}^i$ represents the i-th row from the corresponding camera matrix, $\mathbf{P_1}$.

$$x_1(\mathbf{p_1}^3\mathbf{X}) - (\mathbf{p_1}^1\mathbf{X}) = 0 \tag{12}$$

$$y_1(\mathbf{p_1}^3\mathbf{X}) - (\mathbf{p_1}^2\mathbf{X}) = 0 \tag{13}$$

$$x_1(\mathbf{p_1}^2\mathbf{X}) - y_1(\mathbf{p_1}^1\mathbf{X}) = 0 \tag{14}$$

Constructing a matrix with four rows, $\mathbf{A}$, consisting of Equations 12 and 13 for each view (Equation 15), allows for the solution of the 3D point location of the form $\mathbf{AX} = \mathbf{0}$. Equation 15 shows the final system of equations for linear triangulation [16].

$$\begin{bmatrix} x_1(\mathbf{p_1}^3) - (\mathbf{p_1}^1) \\ y_1(\mathbf{p_1}^3) - (\mathbf{p_1}^2) \\ x_2(\mathbf{p_2}^3) - (\mathbf{p_2}^1) \\ y_2(\mathbf{p_2}^3) - (\mathbf{p_2}^2) \end{bmatrix} \mathbf{X} = \mathbf{0} \tag{15}$$

However, due to image noise and track point error, these lines do not intersect perfectly at $\mathbf{X}$. Numerous triangulation methods exist to attempt to reduce this error [18].

Estimating Pose with the Perspective-n-Point Problem.

The PnP problem is a CV topic that is used to determine calibrated camera pose from one image based on a series of 2D points in the image with known 3D locations. This process can be seen as roughly the opposite of triangulation. Rather than solving for the location of $\mathbf{X}$ in 3D space based on two perspectives ($\mathbf{P_1}$ and $\mathbf{P_2}$) and a 2D point match ($\mathbf{x}_1$ and $\mathbf{x}_2$), $\mathbf{P_n}$ is unknown and is estimated based on i 2D point locations $\mathbf{x}_n^i$ and their known 3D spatial locations $\mathbf{X}^i$. Another notable difference is the number of points and perspectives required. Triangulation requires two separate known perspectives to estimate the location of one or more 3D points. The PnP problem solves for only one perspective, but requires at least three 2D-3D point correspondences. Numerous approaches exist to solve this problem.

One linear algorithm establishes a set of quadratic equations with coefficients depending on image measurements and distances to 3D points. These equations solve for the camera pose by finding the intersection of the rays at the optical center from the 3D points through the image plane. The main advantage of this algorithm is a fast calculation of a set of linear equations, which guarantees the correct solution in a noiseless case. This linear method also avoids local minima solutions which are

possible when using iterative algorithms [3]. However, iterative approaches typically result in reduced error when image noise exists [19].

The iterative method is based on Levenberg-Marquardt optimization of the camera matrix to minimize reprojection error. A point is projected to the 2D image plane from 3D using Equation 4. Reprojection error is the Euclidean distance, represented here by $\text{dist}(\mathbf{x}_1, \mathbf{x}_2)$, between the location of the 2D image point to the projected point location through a camera matrix $\mathbf{P}$. This iterative method serves as a maximum likelihood estimator for $\mathbf{P}$, even in the presence of image noise. Equation 16 shows the minimization function where i is the total number of points.

$$\min_{\mathbf{P}} \sum_{i=1}^{m} \text{dist}(\mathbf{x}^i, \mathbf{P}\mathbf{X}^i) \tag{16}$$

Additional methods have been developed in an attempt to reduce the number of operations required while maintaining the robustness of the iterative approach [50, 24, 36]. OpenCV includes numerous algorithm implementations for performance comparisons if desired [35].

2.1.4 Feature Detection.

Numerous methods exist for detecting features to track in an image. Two of the most popular algorithms for robustness and matching accuracy commonly found in a literature search are the Scale Invariant Feature Transform (SIFT) [27] and Speeded-Up Robust Features (SURF) [5], each of which contain their own proprietary feature detection and descriptor-extractor routine. These types of feature descriptors excel at extracting features from a known object and identifying the same object in a different scene.

The SIFT algorithm has been demonstrated to generate a descriptor for a patch of an image that is invariant to scale, rotation, and illumination. A SIFT descriptor

patch consists of 128 floating-point values. The SURF feature was developed following SIFT in an effort to reduce computational complexity. The SURF descriptor consists of 64 floating-point values, which is faster to compute and can be more robust than the original SIFT descriptor [5]. Figures 6 and 7 in Section 2.1.3 show an example of SURF feature extraction and matching with the MATLAB Computer Systems Vision Toolbox implementation.

While SIFT and SURF are popular algorithms that yield high quality features, they are too computationally expensive to implement in a real-time application that runs at or near 30 fps on a one core single-board computer. The entire series of events to determine a series of point matches can take roughly half a second to multiple seconds, depending on image resolution, feature density, and the RANSAC confidence threshold desired [13].

Feature descriptors that operate more quickly than SIFT or SURF exist. However, unlike SIFT and SURF which have a detection step and an extraction step, most faster running descriptors only describe a location in an image via a specific set of metrics. For this reason, reviewing other feature detection algorithms is required prior to running a point tracker or feature descriptor. When identifying points in an image, it is important to select unique and identifiable locations that are unambiguous to their surroundings. In [41], Shi and Tomasi identify good features to track based on "some measure of texturedness or cornerness such as a high standard deviation in the spatial intensity profile, the presence of zero crossings of the Laplacian of the image intensity, and corners." Figures 13 and 14 in Section 3.2 show the importance of selecting good feature locations.

A standard in the machine learning field for corner detection is the Features from Accelerated Segment Test (FAST) algorithm by Rosten and Drummond. In real-time applications with 30 fps video, the available time for processing before the next frame

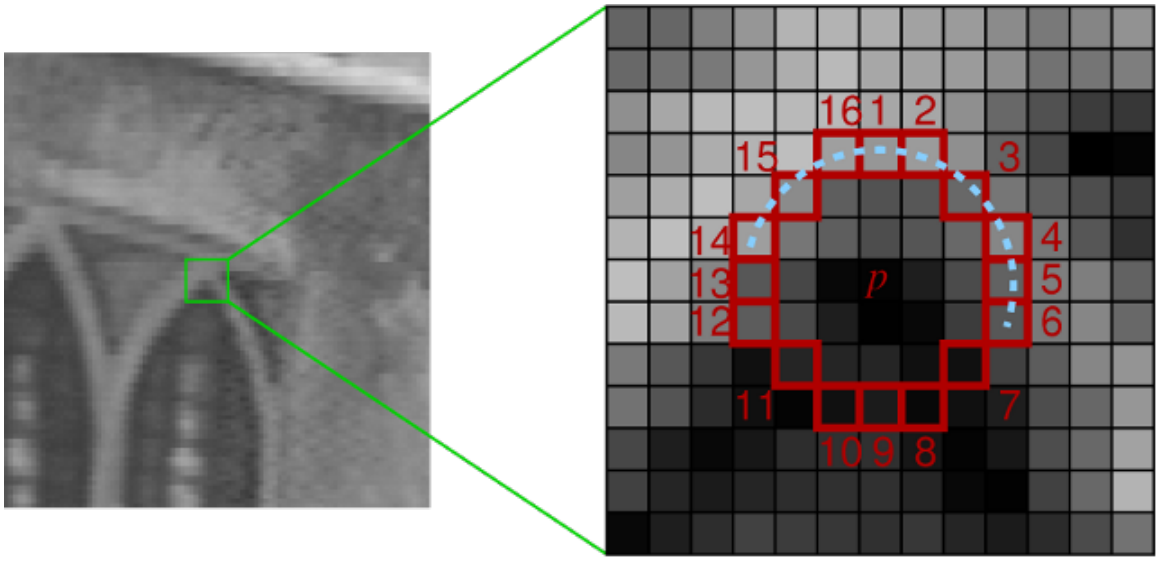


Figure 10. The FAST Corner Detector [39].

is captured is 1/30th of a second. The FAST algorithm can fully process a frame in less than 7% of the available processing time. As a comparison, the Harris corner detector operates in 120% of the available time, and only the detection phase of SIFT in 300%. The FAST detector uses a circle of 16 pixels to test if a specific point is a corner. Figure 10 shows the circle around the candidate point, p, with the pixels in the circle labeled from 1 to 16. A point is a corner if N continuous points in the circle have an intensity greater than the intensity of p plus some threshold value t. A point is also a corner if N continuous points have an intensity less than p minus t. The original FAST algorithm uses an N value of 12 [39].

2.1.5 Two-dimensional Point Tracking.

In [13], Doyle, a predecessor to this research at AFIT, performed a run-time and accuracy comparison of numerous methods of feature tracking. As expected with such a popular but costly descriptor, SIFT and SURF returned the highest accuracy results. SIFT is also the slowest tracking algorithm in the study, with

a series of tracks between two frames with a dynamic background taking nearly 3 seconds. SURF resulted in a slight decrease in accuracy during Doyle's test, but ran faster than SIFT, achieving a track in the same scenario in approximately 1 second. In off-line implementations where accuracy is more important than run-time, SIFT or SURF are most likely the descriptors of choice. The fastest method in this test, while only sacrificing a slight amount of accuracy, is the pyramidal implementation of the Lucas-Kanade method. As a side note, FAST was not considered in Doyle's study.

The Lucas-Kanade method considers a small window of an image at an interest point in one frame, and searches for the most similar window in a subsequent frame by assuming that the flow of the pixels is essentially constant in the local neighborhood. Using this assumption, the optical flow equation, Equation 17, must apply for all the pixels in the current window around the track point [28]. In Equation 17, V_x and V_y are the optical flow velocity components, and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ are the image derivatives.

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \tag{17}$$

Rearranging Equation 17 and representing the image derivatives with I_x, I_y, and I_t results in the series of equations applied to each pixel (q_1 through q_n) during the Lucas-Kanade method, shown by 18. Solving for the velocity vector using least-squares results in the updated window location.

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ &\vdots \\ I_x(q_n)V_x + I_y(q_n)V_y &= -I_t(q_n) \end{aligned} \tag{18}$$

The Kanade-Lucas-Tomasi (KLT) feature matching algorithm is an improvement

to the least-squares approach. One drawback of the Lucas-Kanade method is that the flow velocity vector between frames must be small enough to find a solution, which is often less than the pixel spacing. The KLT method solves this problem by creating a reduced-scale version of the images, or pyramids, for tracking large scale motion. Two papers contributed to the KLT tracker [28, 44]. The MATLAB Computer Vision toolbox includes an implementation of the KLT tracker. OpenCV includes an implementation as well, known as pyramidal Lucas-Kanade optical flow.

In [41], Shi and Tomasi propose using an affine transformation on the tracking template in addition to the translation-based system. Using an affine motion model, a deformation matrix, $\mathbf{A}$, modifies the track template to further minimize the dissimilarity between frames as a result of changes in camera perspective. Using this model, an image point $\mathbf{x}$ in the first image I is tracked to $\mathbf{Ax} + \mathbf{d}$ in the next image, J. The tracker works by finding the parameters $\mathbf{A}$ and $\mathbf{d}$ that minimize the dissimilarity, ϵ, in Equation 19, where W is the given template window and $w(\mathbf{x})$ is a weighting function (typically $w(\mathbf{x}) = 1$ or a Gaussian function are chosen to emphasize the center of the window).

$$\epsilon = \int \int_W \left[J(\mathbf{Ax} + \mathbf{d}) - I(\mathbf{x})\right]^2 w(\mathbf{x}) d\mathbf{x} \tag{19}$$

2.1.6 Binary Feature Descriptors.

SIFT and SURF are based on histograms of gradients, which means the gradient of each pixel in the feature window must be computed, a computationally expensive process. However, when it comes to feature descriptors, SIFT and SURF are not the only choices available. Binary feature descriptors describe a point in an image based on a binary string of intensity measurement comparisons. Computing and storing a binary feature into memory is very efficient with typical descriptor sizes

of 128, 256, or 512 bits. Additionally, comparing binary strings to test a feature match involves computing the Hamming distance, which only requires a bit count or XOR operation between the two strings, a very efficient calculation compared to the Euclidean distance calculation required for SIFT and SURF.

Binary feature descriptors create a binary string to describe an image patch based on a series of point pairs from a sampling pattern. In a 256-bit descriptor, 256 point pairs are chosen. For each entry in the binary string, the intensity values of the two points in a pair are compared. If the intensity value of the first point is greater than the second point, the binary value for that point pair is 1; otherwise, it is 0. Comparing two features then only involves a binary comparison of the feature strings to find the Hamming distance.

Numerous binary feature descriptors exist. The Binary Robust Independent Elementary Feature (BRIEF) descriptor, introduced by Calonder, is chosen based on successful matching results, comparable to SURF while obtaining the efficiency advantages of binary features in [12]. Calonder shows, through a series of tests, that a random sampling of point pairs in an isotropic Gaussian distribution, G II in Figure 11, gives the best results in terms of recognition rate.

When dealing with point comparisons via the Hamming distance, it is important to determine a value under which two features will be considered a match. In [12] a Hamming distance comparison test between actual matching points and non-matching points is performed. Figure 12 shows the results of the Hamming distance comparison test, where the blue lines are the Hamming distance between matching points, and the red lines are the Hamming distance between non-matching points. The distribution of distances for non-matching points is roughly Gaussian and centered around the middle of the possible range. As the plots progress in Figure 12, camera separation increases, causing an increase in dissimilarity between points and a corresponding

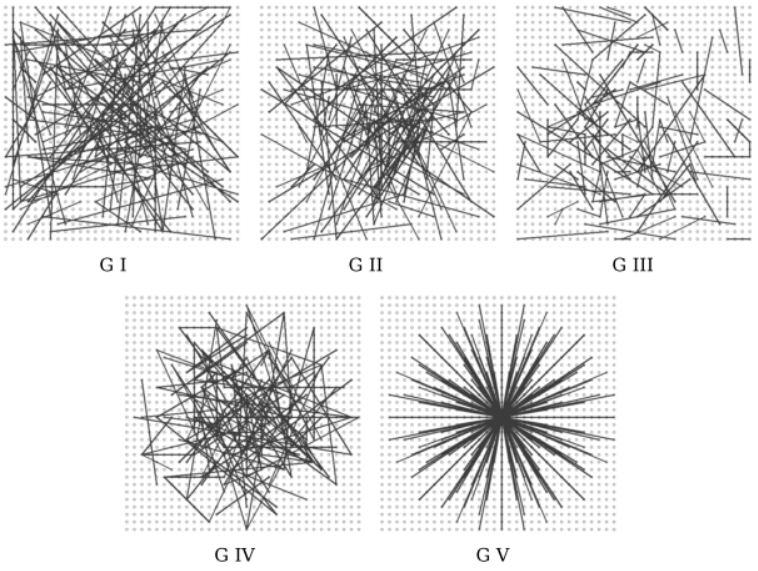


Figure 11. Different BRIEF Sampling Approaches Tested in [12].

increase in the Hamming distance for positive matches.

2.1.7 Bundle Adjustment.

In [45], bundle adjustment is defined as "the problem of refining a visual reconstruction to produce *jointly optimal* 3D structure and viewing parameter (camera pose and/or calibration) estimates. *Optimal* means that the parameter estimates are found by minimizing some cost function that quantifies the model fitting error, and *jointly* that the solution is simultaneously optimal with respect to both structure and camera variations." The name bundle adjustment refers to adjusting the bundles of light rays that point from each structure point to the camera centers.

Bundle adjustment begins with a set of initial 3D scene and camera parameter estimates based on a set of 2D image observations. A nonlinear cost function is developed in which the set of 2D points, $\mathbf{x}$, are typically compared with a projection of the 3D points on the image plane based on the camera parameters (Equation 4). The 2D Euclidean distance between a reprojected 3D point from its initial 2D image

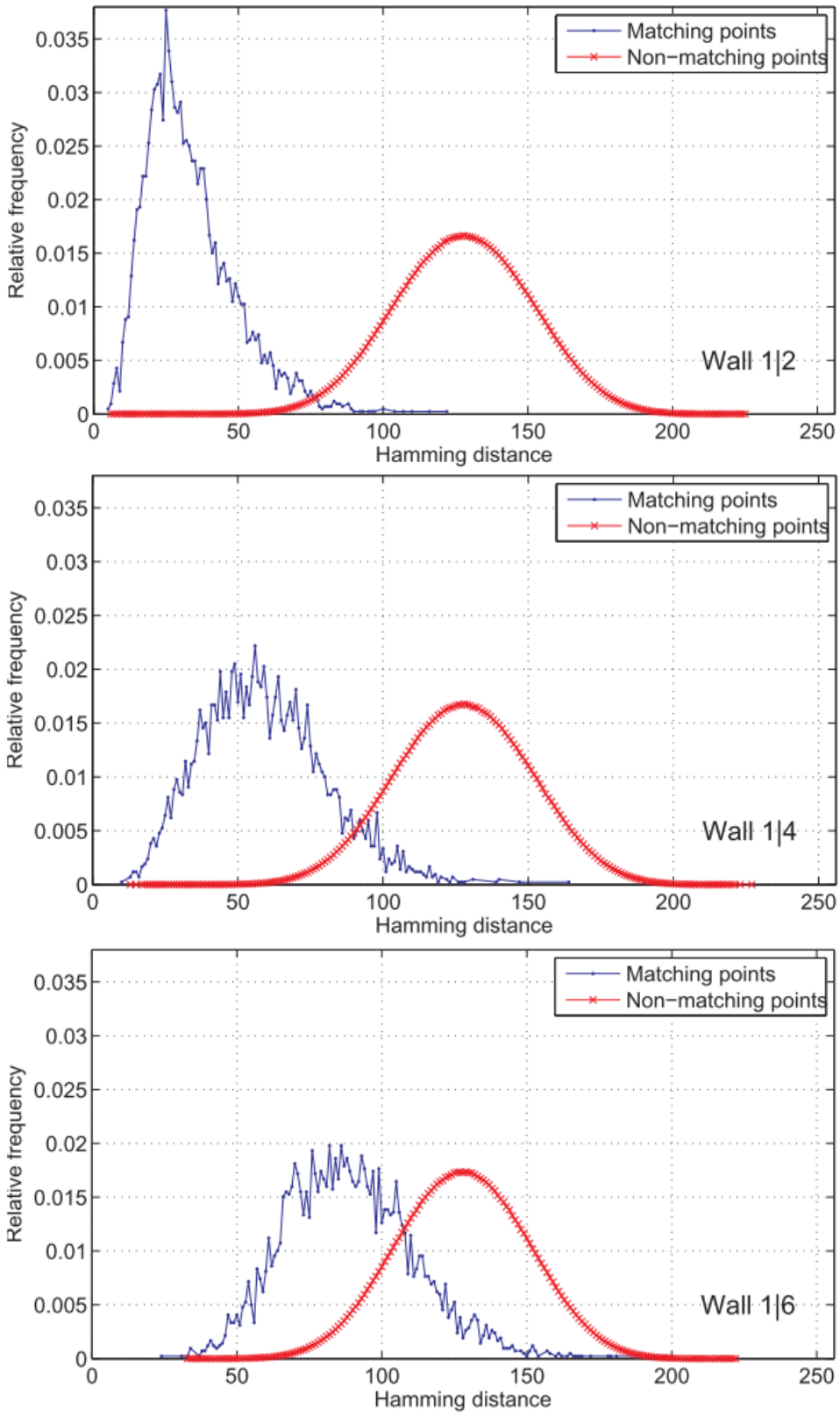


Figure 12. Hamming Distance Distribution for Matching and Non-Matching Points [12].

location yields a quantity known as the *reprojection error*. Numerical techniques are then used to minimize the total reprojection error of m 2D point projections across n views by optimizing camera pose estimates $\mathbf{P_n}$, the 3D scene points $\mathbf{X}^i$, and even calibration parameters $\mathbf{K}$ if desired. Equation 20 is the nonlinear cost function constructed if only one camera is used and there is no change in calibration parameters between views. Bundle adjustment is the model refinement portion of this problem.

$$\min_{\mathbf{P}_j, \mathbf{X}^i, \mathbf{K}} \sum_{j=1}^{n} \sum_{i=1}^{m_j} \text{dist}(\mathbf{x}_j^i, \mathbf{P}_j \mathbf{X}^i) \tag{20}$$

There are many different ways to incorporate bundle adjustment into a 3D reconstruction method. Various strategies exist for the number of frames, structure points, and parameters to modify. For off-line systems that do not require high-speed run times, an entire set of observations can first be made prior to running bundle adjustment. Using these observations, there is no time limit to optimize the resulting nonlinear system and globally converge on a solution. However, in on-line systems that require performance at live video rates, performing the problem incrementally with a higher threshold for allowable error may be a requirement. Section 2.2.1 explores various research approaches and numerous bundle adjustment strategies.

A range of numerical methods may be implemented for optimization of the cost function as well. The Levenberg-Marquardt Algorithm is a popular approach and is implemented in the generic C/C++ sparse bundle adjustment package, SBA [26] (further discussed in Chapter III).

2.2 Similar Approaches

This section reviews similar approaches other researchers have taken. Specific works are mentioned due to unique bundle adjustment approaches implemented, as well as accurate and fast running structure from motion (SFM) and SLAM systems.

2.2.1 Survey of Monocular SLAM Research.

VisualSFM [47] is a software package for performing 3D reconstruction on an unordered set of images. It uses a graphics processing unit (GPU) implementation of SIFT to identify accurate point matches significantly faster than the traditional SIFT algorithm. It also exploits multi-core bundle adjustment, and despite its operation on a batch of images, still performs SFM in an incremental fashion. In [46], Wu describes the strategy behind the approach to minimize bundle adjustment run-time in the VisualSFM implementation. Incremental SFM is known to accumulate error over multiple relative camera pose calculations. Wu proposes an approach in which failed feature matches are regularly re-triangulated during the incremental SFM process to reduce error. Principles from VisualSFM can be applied to this research, however run-time limitations of an off-line SFM software package prevent direct application.

In [4], Balzer and Soatto describe a method they call *CLAM: Coupled Localization and Mapping with Efficient Outlier Handling*, which generates a model of a small-scale object from a video. It excels at handling self-occlusions from in-plane rotation, which challenges other SFM/SLAM systems. They implement the KLT tracker and a method to apply a consistency check using binary features and an affine transformation of the track window to increase similarity for successful matches. Their outlier handling techniques specifically avoid the use of expensive RANSAC techniques. This work achieves very impressive results considering accuracy and run-time in comparison with other on-line SLAM algorithms. Bundle adjustment is only run on a two-frame basis to minimize computation times; as a result, the global accuracy suffers in comparison to off-line methods. In addition, Balzer generated a series of synthetic scenes using Blender to benchmark his algorithm and shared them online to save time for other CV researchers. Some of these scenes are used in this work, with results shown in Chapter IV.

Another SLAM method developed by Klein and Murray called Parallel Tracking and Mapping (PTAM) [22, 23] uses parallel processors to separate the tracking and mapping tasks. The tracking thread processes the input video and tracks the pose of the camera relative to the map. The mapping thread creates the map, selects the appropriate video frames to expand the map, and triangulates new features to the map that exist in two camera poses. It also performs bundle adjustment to optimize pose estimates and feature positions. Point tracks are initiated based on the FAST corner detector. A search along the point's matching epipolar line in the subsequent frame minimizes the search area required for updating point tracks.

PTAM allows the mapping thread to perform local bundle adjustment in which only a subset of poses is adjusted. For example, only the parameters for the most recent 3 frames are optimized. These optimizations, however, are based on measurements from the 7 prior frames. PTAM also interrupts the bundle adjustment process if a new frame is available to be added to the map, which serves to integrate the newest information as soon as possible. On the other hand, when no additional information is provided from updated frames because no new track points have become visible, the mapping thread uses this free time to improve the map. Klein and Murray present creative hardware implementations of PTAM as well. In one implementation they perform augmented reality in real time on the touch screen of a camera phone.

2.3 Summary

The intent of this work is to use the theory described in this chapter to write a modular SLAM method for implementation in an indoor proxops simulation laboratory. Specifically, the SLAM method will be written to handle the demands of the space environment. First, the method should be robust enough to manage point tracks on reflective surfaces with changes in lighting. Isolating target points from a

dynamic background, such as the Earth or stars, is a key design parameter as well. Finally, the method should automate the rate at which pose is estimated to handle varying rates of relative motion.

Other SLAM codes, such as PTAM and CLAM, are shared on the internet. However, a direct implementation of another researcher's code without a detailed understanding of the underlying theory makes testing and customization for the space environment nearly impossible. Nevertheless, this thesis does share many of the same fundamentals and leverages applicable lessons learned by other researchers. The following chapters of this thesis and the included set of (thoroughly commented) MATLAB routines provide this foundation in an environment that is easily interpreted and enables rapid prototyping.

III. Methodology

This chapter provides a description of the SLAM method developed to meet the research objectives. First, Section 3.1 details the specifics of the hardware and software systems used for development. Next, Section 3.2 shows how 2D point correspondence data is acquired across video frames. Section 3.3 describes how these 2D points are used for structure and motion estimation. Finally, Section 3.4 provides a summary of the method developed.

This implementation is unique because it is designed to operate on objects in the space environment. It stands out from other proxops methods because it is able to operate without any knowledge of, or cooperation from, the target object, using only one camera. This process is based on the theory explained in Chapter II. The ease of prototyping new ideas is a key aspect of the design as well. This implementation will inevitably be improved upon by future researchers.

3.1 Research Platform, Devices, and Software

3.1.1 Hardware.

A Dell M6600 Precision laptop was used for development and prototyping. Table 1 shows the computer specifications.

Table 1. Computer Specifications

Manufacturer:	Dell
Model:	M6600 Precision
Processor:	Intel Core i7-2820QM @ 2.30 GHz
Installed memory (RAM):	16.0 GB
Graphics Card:	NVIDIA Quadro 4000M
Dedicated Graphics Memory:	2.0 GB RAM
Processor Cores:	4
System type:	64-bit

A Point Grey Research Flea 3 camera was used to acquire video sequences analyzed in Chapter IV. Table 2 shows the camera specifications and Table 3 shows the specifications of the attached lens.

Table 2. Camera Specifications

Manufacturer:	Point Grey Research, Inc.
Product Name:	Flea 3 USB 3.0
Model:	FL3-U3-13S2C/M-CS
Megapixels:	1.3
Imaging Sensor:	Sony IMX035 CMOS, 1/3"
Max Resolution:	1328x1048
Max Frame Rate:	120 fps
Pixel Size:	$3.63\mu m$
Transfer Rate:	5 GBit/sec
Digital Interface:	USB 3.0

Table 3. Lens Specifications

Model:	LENS-30F2-V80CS
Focal Length:	$2.8 \sim 8mm$
Angle of View $(H \times V)$:	$100°00' \times 73°45'$
Angle of View $(H \times V)$:	$109°45' \times 59°51'$
Operation:	Manual zoom, focus and iris
Iris Range:	F1.2 $\sim$ Close

3.1.2 Software.

The methods developed in this thesis were written in MATLAB r2014b. The MATLAB Computer Vision Systems Toolbox and the open computer vision library (OpenCV 2.4.10) with the MEX-file (MATLAB Executable) interface [48] were also utilized. MATLAB is the development environment of choice because of its simplicity and speed in quickly prototyping new ideas. Unfortunately, MATLAB is not the most computationally efficient programming language and is not tailored for large image

sequences. Near real-time operation has been achieved despite the inefficiencies associated with MATLAB. Porting the methods developed to a more suited environment for CV applications such as Python or C++ is not exceedingly difficult should it be necessary, and MATLAB code will be the easiest to interpret for a successor of this work.

3.2 Creating Robust Point Tracks in a Video Sequence

Selecting good features to track is a critical part of this process since 2D point locations and correspondence between images make up the source data from which motion and structure are estimated. Uncertainty in the location of point matches will cause error to propagate during subsequent steps and serve as a poor foundation for calculations. An important aspect of generating accurate point tracks is identifying image features that are the best for tracking. The goal for detecting features in an image is finding unique and identifiable points which are easy to compare and track.

Figure 13 is shown for demonstrating the importance of selecting features that are ideal for tracking. Figure 14 shows six small windows of individual features extracted from Figure 13 for demonstration purposes only. The locations of some features are more easily identified in Figure 13 than others. Features 14a and 14b are nearly impossible to identify due to being mostly a flat surface. Features 14c and 14d are easier to identify, but uncertainty still exists. Since they are edge features, it is trivial to match them with other edges; but the exact point on the edge is very difficult to identify. Finally, features 14e and 14f are very easily identifiable because they occur at corners, removing much of the ambiguity with edge features. This analogy can be made for the ability of a computer to track specific features as well. If a feature is difficult for a human to identify, it will most likely be difficult for a computer to identify as well. These issues were described in detail by Shi and Tomasi who reviewed

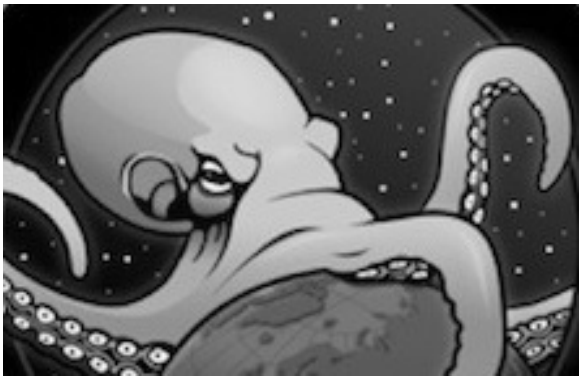
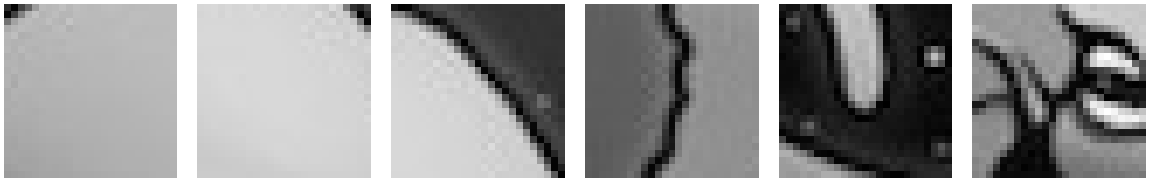


Figure 13. Sample Image for Feature Extraction.



(a) (b) (c) (d) (e) (f)

Figure 14. Six Features from Figure 13.

the methods for selecting the best features to track and demonstrated quantitatively that corner points and features with texture are ideal choices for tracking [41].

3.2.1 Initial Point Detection.

The FAST algorithm (Section 2.1.4) was selected for corner point detection primarily for its improved speed when compared with other options. While FAST is not as robust to high levels of noise, it is several times faster than other existing corner detectors [39]. Figure 15 shows the results of applying the FAST corner detector to an image of Texas A&M's Holonomic Omni-direction Motion Emulation Robot (HOMER) [20].

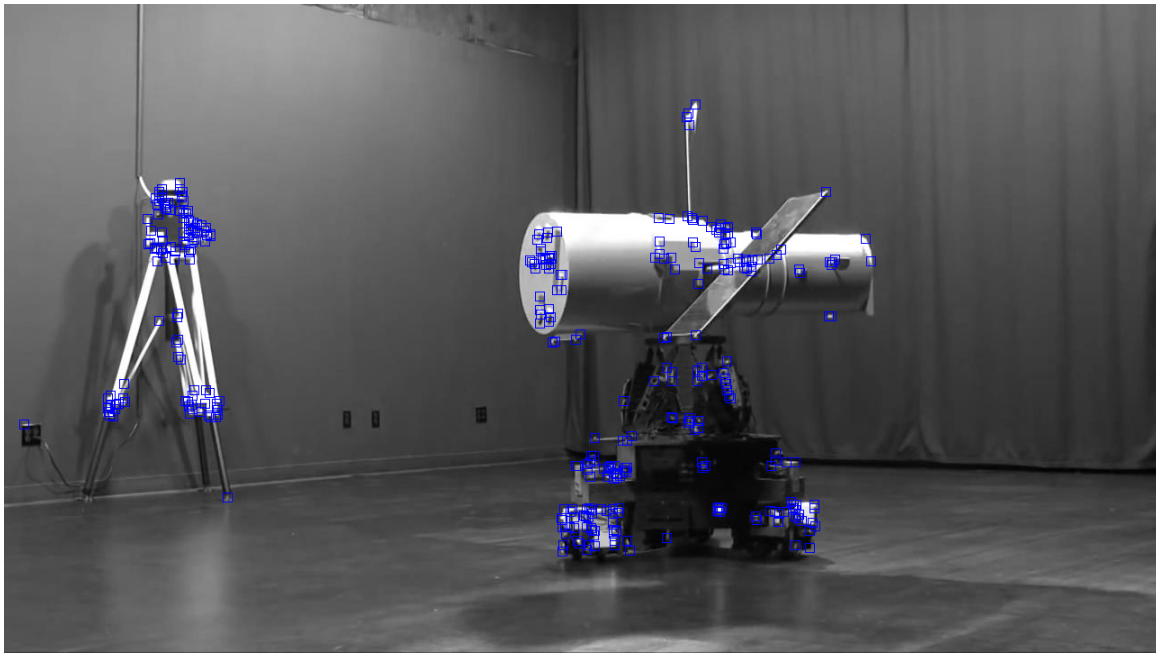


Figure 15. FAST Algorithm Applied for Corner Detection.

3.2.2 Minimizing Point Detections to a Region of Interest.

A challenge using CV pose estimation methods for a proxops simulator such as the one shown in Figure 15 is the satellite model moves independently from the environment. Typically, background features of a scene provide reference points for updating pose estimation based on the camera's motion. However, in this case, the relative motion results from the rotation of the satellite model while the camera stays fixed. Any background points identified must be ignored, as they do not follow the same motion model as the target. Any points from the background that are included in the fundamental matrix estimation step will undoubtedly cause erroneous results as there is no relative pose change with respect to the background.

The same concept applies to a background of stars or the Earth's surface for a satellite in low Earth orbit (LEO) facing the nadir direction. Just like the lab proxops simulator example, to obtain relative pose between spacecraft, track points on the target object must be isolated from points identified in the environment. Dealing

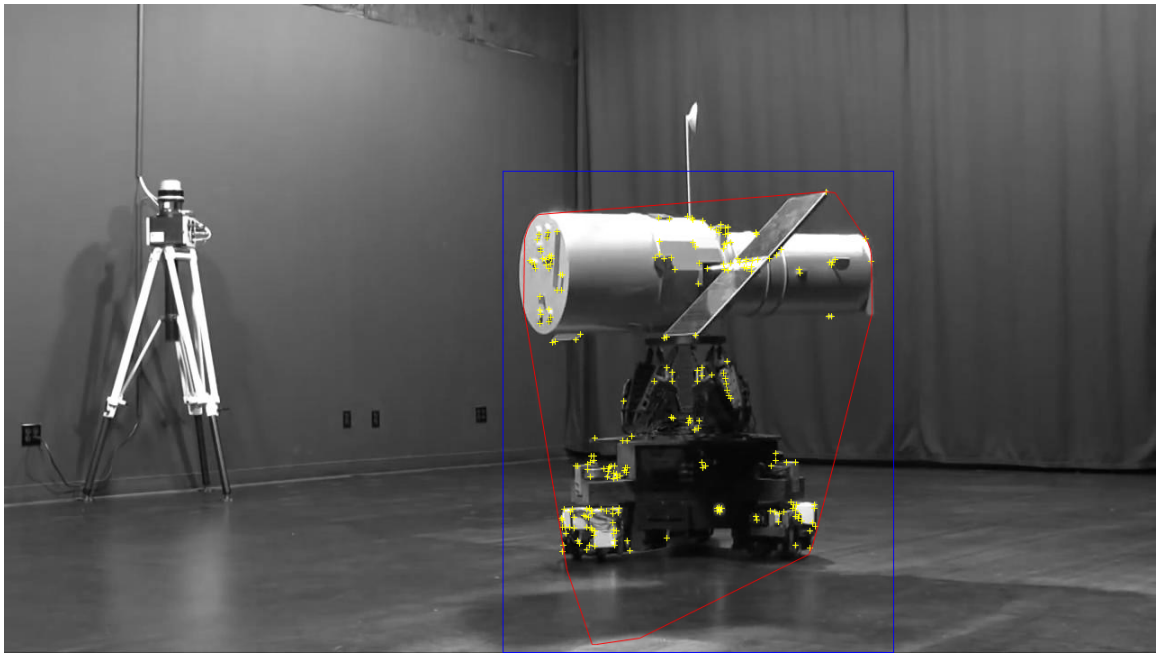


Figure 16. FAST Algorithm Applied for Corner Detection in a Motion-Based Region of Interest.

with the Earth's surface in the background is less of a concern for satellites at higher altitudes, such as geosynchronous orbit, since the Earth will subtend a significantly smaller solid angle.

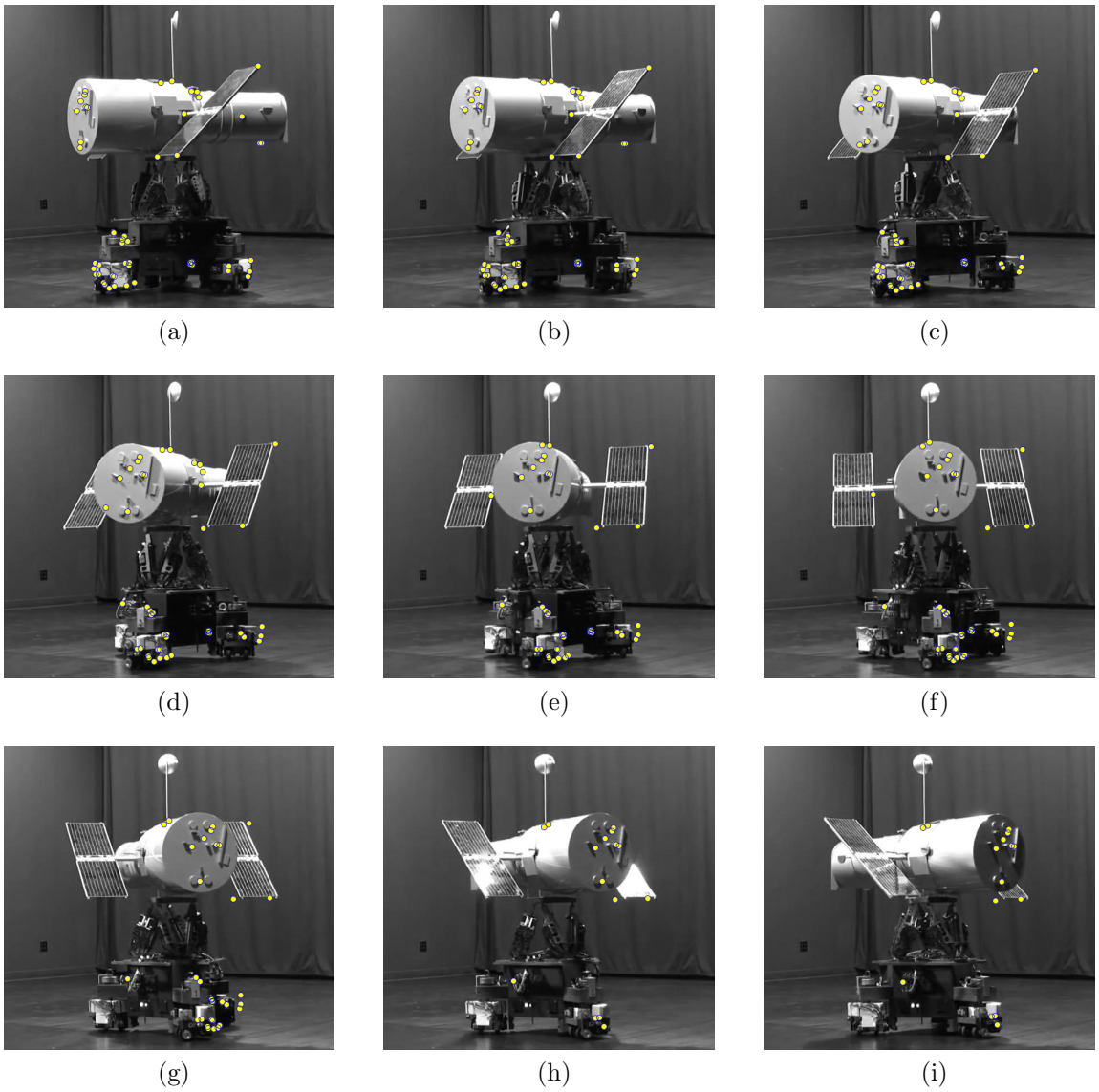
To isolate background points for the proxops simulator case, a background subtraction/motion detection algorithm written here is used, defining a smaller sub-region to apply the FAST algorithm. Applying the FAST algorithm to this sub-region of interest has a secondary benefit of reducing the number of operations that must be performed. The first step is a normalized image difference, followed by convolution with a Gaussian filter to smooth image noise, and finally a thresholding operation to define a convex polygon that encapsulates a region of interest. Figure 16 shows the FAST corner points detected inside blue rectangular region of interest, with the convex polygon shown in red.

3.2.3 Point Tracking.

Once points are identified on the object using the FAST algorithm, the tracker developed by Kanade, Lucas, and Tomasi is used to track these points through subsequent frames (the KLT tracker, introduced in Section 2.1.5). The KLT tracker is a sparse optical flow algorithm that tracks a feature by minimizing an objective function to update the location in the next frame. A translation-only model exists in which a small image patch in one frame is compared with image patches in the next frame in order to estimate the most likely updated location based on an iterative search. Figure 17a shows the initial corner point detection using the FAST algorithm (a minimized set for demonstration). Figures 17b through 17i show the subsequent results of the KLT tracker through the rotation of the satellite model.

In Figures 17b through 17i, general KLT track point motion appears to move consistently with the rotation of the model. Upon further analysis, a few observations can be made. First, the points are drifting from the initially detected location on the model. Ideally, each point track should exist the entire time it is in view, from detection until the point rotates out of view. Drift occurs in a few different instances in the series of point tracks. A close-up view of one point track is shown in Figures 18a through 18l in which a point is initially detected and tracked as the model rotates. However, as the point moves to one side of the object, the corner begins to look more like an edge. Once the KLT track point is aligned with the side of the model as it rotates, the small window extracted near the point looks very similar from frame to frame and it continues to be detected as a successful track even though the actual point is occluded. Drifting points can cause gross errors on the subsequent position estimation.

Another observation from Figure 17 is the disappearance of points as motion occurs, which is expected. As an object rotates, eventually all the initial point detec-



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 17. KLT Track Points.

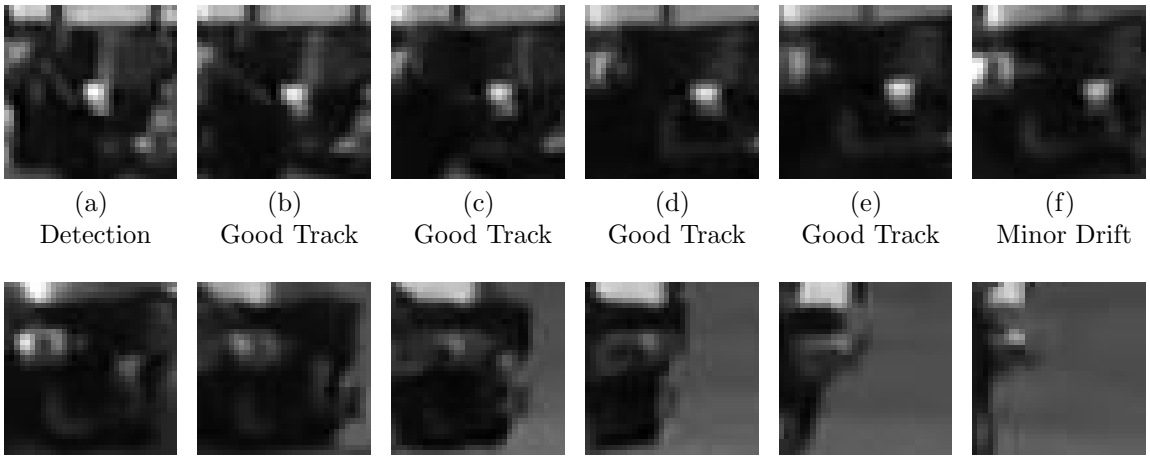


Figure 18. The History of a Drifting Translation-Only KLT Track from Figure 17.

tions will be facing the opposite direction and no longer visible. This disappearance of points, however, requires the implementation of a method to detect when a new area of the object has appeared and requires point detection, while gracefully being combined with the previously tracked motion.

3.2.4 Methods to Eliminate Point Drift.

A significant issue with the KLT point tracker presented is point drift. The KLT translation-only tracker identifies a point track in frame n based on the best match in frame n-1. For example, the window in Figure 18a is very similar to 18b, and 18b is very similar to 18c. However, 18a and 18f begin to show dissimilarities, and the deviation becomes significant with windows 18i through 18l. Updated versions of the KLT tracker utilize a method known as an affine consistency check [41, 7], an iterative method that computes the translation and iteratively applies an affine transformation to the feature window to additionally minimize error against the initial occurrence. Implementation of the affine consistency check on a large scale with hundreds of point tracks has been prohibitively slow; therefore, a separate secondary comparison

is utilized.

The BRIEF Feature.

A secondary method to verify point tracking with the KLT tracker was explored to recognize and remove drifting point tracks. One of the goals of this secondary approach is to keep the process as simple as possible, still able to run in real time, while being universal and robust. As described in Section 2.1.6, calculating a BRIEF feature is incredibly fast. Comparing BRIEF features to one another is also very efficient, only requiring the calculation of the Hamming distance, a significantly faster computation than the typical Euclidean distance.

The secondary verification method starts with extracting a BRIEF feature upon initial point detection (the *library* BRIEF feature). As the translation-only KLT tracker updates the point location, another BRIEF feature is extracted. The new BRIEF feature is compared to the library feature via the Hamming distance. If the Hamming distance exceeds a threshold value, then the track point is ended. An issue that arises from analysis of Figure 12 is, as in-plane rotation increases and the perspective of the target changes, determining a positive match using the Hamming distance becomes increasingly ambiguous. To reduce this ambiguity, the library BRIEF feature is periodically updated (if it is still a match) based on the amount of relative motion that has occurred since initial detection or the last update.

Figure 19 shows the result of no updates to the BRIEF feature without ending any tracks from Figure 18. The blue line corresponds to the feature windows in Figure 18 with the letters corresponding to each specific figure along the top of the plot, starting with Figure 18a. The other two lines represent other point tracks from Figure 17 to show the similarity. Figure 20 shows the resulting Hamming distance history if the source BRIEF feature is periodically updated automatically based on

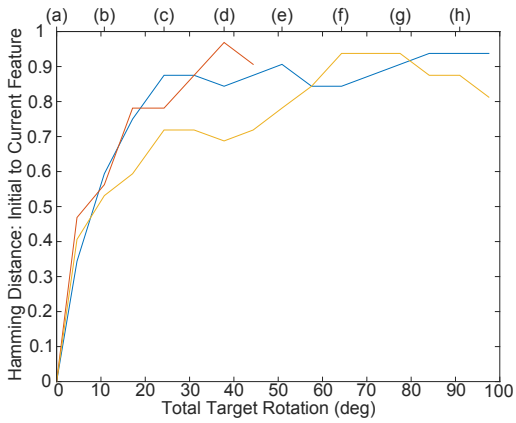


Figure 19. Hamming Distance with No Feature Updates on the Drifting Points from Figure 18.

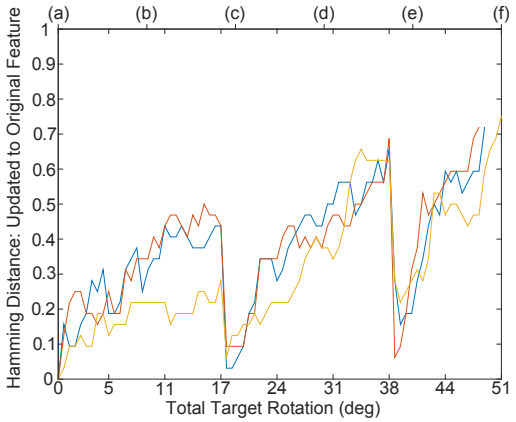


Figure 20. Hamming Distance with Motion-Based Feature Updates. Point Tracks End When Hamming Distance Threshold is Exceeded.

camera translation. In the current example, this method ends the track between Figures 18e and 18f. In this example, rotation estimates on the X-axis of Figures 19 and 20 are made by running the completed version of the SLAM code on the same video sequence and correlating the frame numbers to the amount of rotation.

The Hamming distances for BRIEF feature comparison shown in Figure 12 are for a single comparison only. A Gaussian distribution exists for incorrect matches around the 50% similarity point. However, the method of implementation in this thesis compares the current feature to the library feature, at a minimum, on the order of 10 to 20 times before the library feature is updated (depending on the relative motion rate, which determines the rate at which the library feature is updated). Statistically, it becomes very difficult for an incorrect match to exist over this many comparisons. Even with a Hamming distance threshold as high as 50%, if track points are not projected to 3D until they exist through a minimum amount of relative motion and a comparable number of Hamming distance tests, the result is removing the majority of drifting and incorrect points matches. However, statistically, some successful point tracks are ended prematurely as well. The probability of a successful match with a Hamming distance greater than 50% is small (the blue lines in Figure 12), but it does

increase when multiple comparisons are performed over increasing relative separation.

An additional verification method for the KLT tracker is to perform the track in reverse to the previous frame. The difference in pixel distance from the location in which the track originated to the reverse track location is called the bi-directional error. A threshold is set for the maximum allowed bi-directional error. If the point track exceeds a threshold for the bi-directional error, then the point track is ended. This additional verification method succeeds in ending point tracks which contain ambiguous features, such as edges, which contribute to inaccurate point tracks. A maximum bi-direction error value of 0.1 pixels is used in this method.

The BRIEF feature update method is very quick to calculate and successfully eliminates the majority of point drift. Unfortunately, some point tracks are ended prematurely while the point is actually still in view. Ending point tracks early is not ideal from the standpoint of minimizing error across numerous views. Maximizing point track life and accuracy while in view is one area that should be considered for future work and is discussed in Chapter V.

3.2.5 Handling Additional Outliers.

Outliers still exist that can't be discovered locally. Glossy surfaces can be incredibly difficult to maintain accurate point tracks and the external surfaces of spacecraft can be very reflective. For example, a bright reflection on a shiny surface may change gradually, resulting in a point track that follows the reflection rather than the physical surface location. For this reason, global outliers are detected and removed separately. These outliers are dealt with based on global motion and reprojection error and are discussed in Section 3.3.2.

3.2.6 Detecting New Points.

As relative motion occurs and point tracks disappear, it is necessary to detect new points for continuous tracking and motion estimation. It is also important to proactively detect new points and initiate new tracks as points appear, rather than retroactively detecting points once all tracks have ended. The methodology used for performing structure and motion initialization and the subsequent perspective calculations will highlight the importance of incorporating new points in Section 3.3.1.

The detection of new points is based on the density of currently visible point tracks in sub-regions of the target region of interest. The region of interest (discussed in Section 3.2.2) is divided into n sub-regions. If the number of points in any sub-region falls below a specific threshold value, the FAST algorithm is used to detect new points, applied only to this specific sub-region. In this manner, track points are gradually added to the image in sub-regions that have recently rotated into view or are becoming unobscured for other reasons. Figure 21 shows a sample point density calculation with 9 sub-regions in a video from the Prototype Research Instruments and Space Mission technology Advancement (PRISMA) Satellite [37] performing on-orbit formation flying.

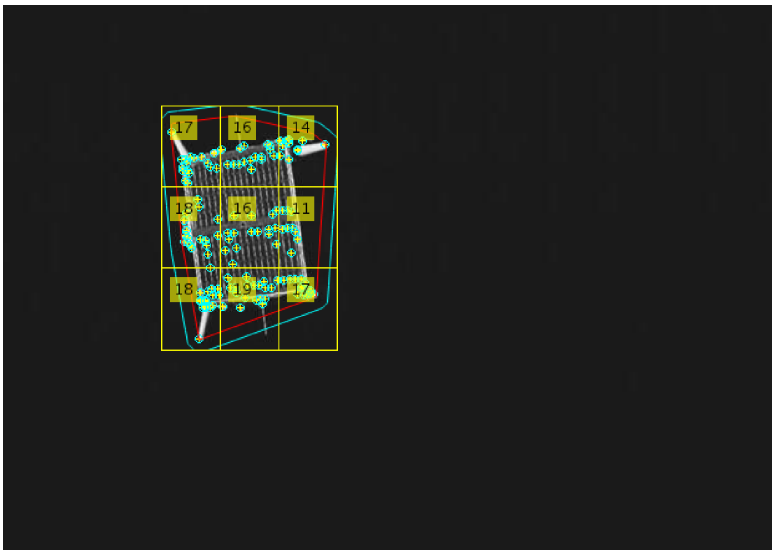


Figure 21. FAST Algorithm in Sub-regions with Point Density (Original Picture Credit: [37]).

3.3 Simultaneous Localization and Mapping

The methodology for generating point tracks implies it is a sequential process: points are identified, points are tracked to subsequent frames, and new points are identified and tracked. The track point locations provides the data for the structure and motion estimation, a separate process, relying on an entirely different foundation of CV theory. Contrary to the sequence in which these two topics are presented, they occur together in this section; they are simultaneously performed and jointly reduce error to optimize the structure and motion estimation.

3.3.1 Structure and Motion Initialization.

Initially, point correspondences are generated between two frames. Nothing is known about the location of either of the cameras capturing the points or the 3D location of the points themselves. Section 2.1.2 introduced multi-view geometry theory, including the fundamental and essential matrices. Section 2.1.3 utilized the essential matrix to estimate the relative pose between two cameras, based only on

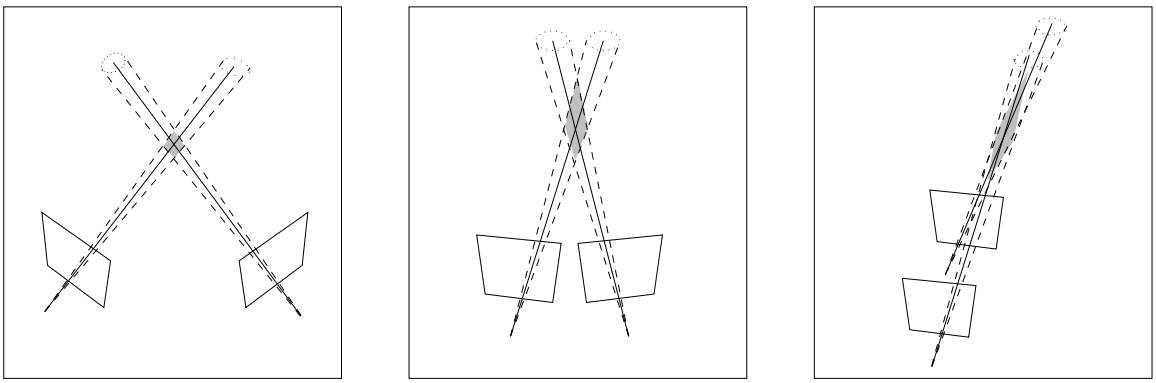


Figure 22. Increased Uncertainty As Relative Camera Separation Decreases [16].

point matches between two different views. Once two camera poses are estimated, the 2D point matches are triangulated to determine a 3D location. Performing the first set of relative pose estimates and projecting the initial set of 2D point matches to 3D will be referred to as *initialization*.

It is important that initialization is accurate as subsequent pose estimates will rely on the estimates generated during this step. Gross error during this step will most likely cause complete failure of the method in future steps. A very important nuance during this step that has a significant effect on accuracy is the variation in pose between camera views. If two frames from a video or live feed are selected to perform initialization, there must be adequate relative translation between cameras. Figure 22 depicts this phenomenon: if there is small relative motion (or only camera rotation) no additional depth information is acquired and the fundamental matrix will be poorly conditioned. An additional note is that a pure rotation of the target object (the case with the HOMER video sequence in Figure 17) provides the relative rotation and translation required from the perspective of the camera for successful initialization.

The first step in initialization is calculating the fundamental matrix using the Normalized 8-point Algorithm (Section 2.1.3) [17]. The essential matrix is then cal-

Figure 23. Epipolar Lines Shown For Fundamental Matrix Estimation.

culated using the camera intrinsics and the fundamental matrix. Figure 23 shows the point matches and epipolar lines described by the fundamental matrix.

Using singular value decomposition, the relative rotation, $\mathbf{R}$, and translation, $\mathbf{t}$, are decomposed from the essential matrix. Since only relative motion is known, the frame of the first camera can be placed in any location and orientation desired based on external information (a star tracker, for example). For simulation, the first camera is placed at the origin of the world frame, $[0, 0, 0]^{\mathrm{T}}$, pointing along the z-axis (the x-axis and y-axis being in the image plane and the z-axis toward the scene being imaged). In this manner, the camera matrix for camera one is represented by $\mathbf{P_1}$ in Equation 21. The relative rotation and translation is used to construct the camera matrix for camera two, $\mathbf{P_2}$. Since the camera intrinsics do not depend on the scene being imaged, the same $\mathbf{K}$ matrix can be used as long as the focal length is fixed. This is an adequate assumption as long as a variable zoom lens is not used.

$$\mathbf{P_1} = \mathbf{K}\,[\,\mathbf{I}\,|\,\mathbf{0}\,] = \begin{bmatrix} f/s_x & f/s_x \cot\Theta & x_0 \\ 0 & f/s_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{21}$$

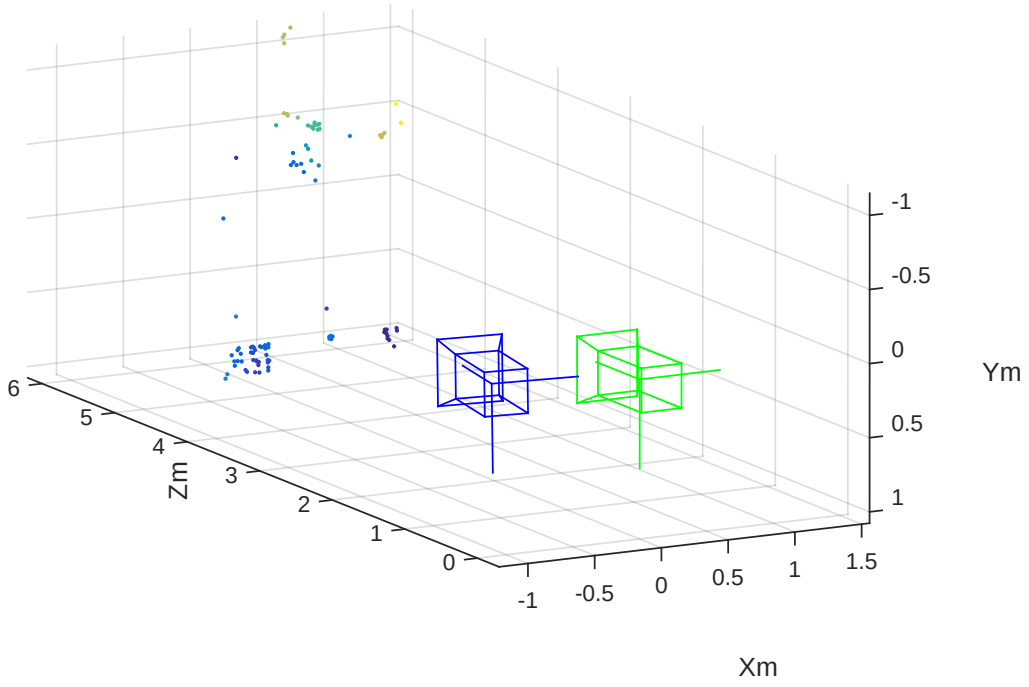


Figure 24. Structure and Motion Initialization.

$$\mathbf{P_2} = \mathbf{K}\,[\,\mathbf{R_2}\,|\,\mathbf{t_2}\,] = \begin{bmatrix} f/s_x & f/s_x \cot\Theta & x_0 \\ 0 & f/s_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{22}$$

The next step is to use triangulation to localize points in 3D based on two known perspectives and their 2D image locations. Figure 24 shows the result of initialization. The green camera model represents the pose of the first camera at the world origin. The blue camera model represents the pose of the second camera (a counter-clockwise rotation of the satellite model, or a relative translation and clockwise rotation of the camera). Finally, the triangulated points on the model are represented by the point cloud.

One important note regarding the specific size and units of the model is that they are unknown. The relative dimensions of the size of the model compared to

the motion of the camera are maintained, but the scale of the scene is ambiguous. Without additional information about the scene, the scale ambiguity remains. Recall the scale factor s (Equation 4 in Section 2.1.2). In a spacecraft performing proxops, a common choice for the data providing scale information may be the range (depth) to the target. The range will most likely be available from previous steps in the process based on the knowledge of the target object's orbit, which is required to rendezvous with a RSO. Section 4.2 shows the results of performing this scaling operation based on camera depth in a laboratory-acquired training video.

When performing initialization, a balance is made between obtaining a maximum amount of motion (more frames apart) versus maintaining enough point matches to perform pose estimation. Using a standard number for frame separation to perform initialization is not an adequate method as variable relative motion rates are possible.

To automate the initialization process, the distance between each track point's current location at frame n, $\mathbf{x}_n^i$, and the track point's initially detected location in frame 1, $\mathbf{x}_1^i$, is calculated. Comparing the average translation of m track points that still exist in image n, $\left[\sum_{i=1}^{m} dist(\mathbf{x}_1^i, \mathbf{x}_n^i)\right]/m$, to the diagonal dimension of the image (the distance from one corner to the opposite corner in pixels) yields a relative value for average pixel motion to the image size. Initialization is performed when the average pixel translation of all track points in the image sequence is at least equal to the relative pixel motion threshold. Section 4.5 shows the results of a sensitivity analysis of this parameter on the average success rate of initialization and subsequent SLAM steps for the training videos used in Chapter IV. The required relative camera translation issue (Figure 22) still exists. It is up to the user to ensure initial relative motion occurs in an adequate manner. However, following initialization, no motion constraint exists.

The number of track points remaining is an important consideration for initializa-

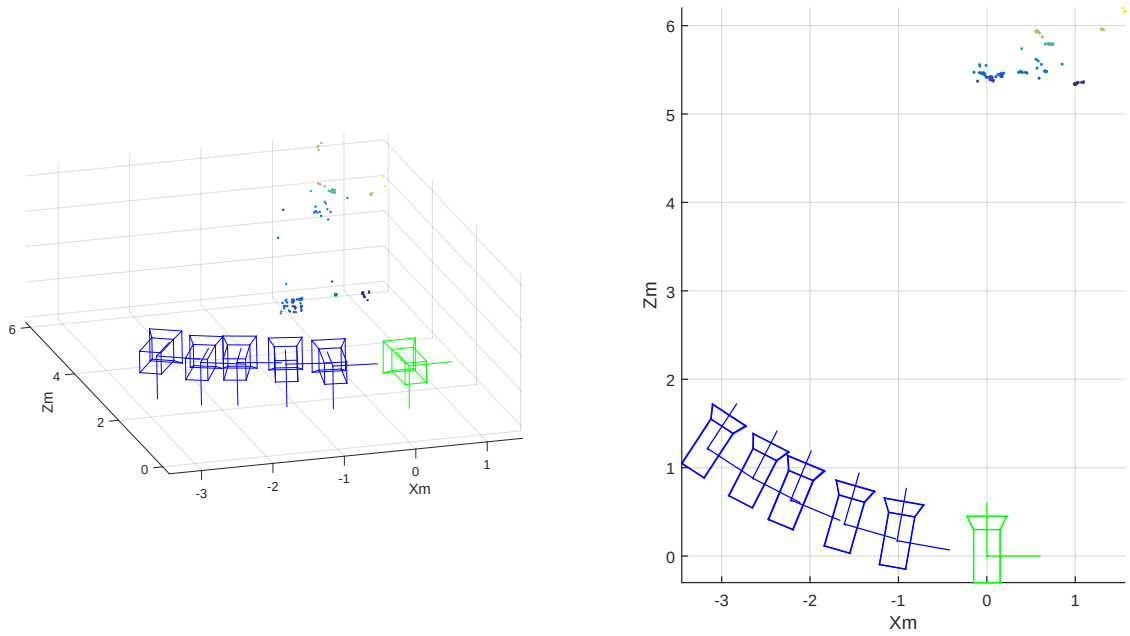
tion as well. As a fail safe to only using the relative pixel motion threshold method, initialization is automatically performed if less than 20% of the initially detected points remain. As an additional method of reducing error and ensuring a successful initialization, bundle adjustment is performed on the first three structure and motion estimates, incorporating an additional view in case the first initialization was not a complete success (discussed further in Section 3.3.3).

3.3.2 Sequential Model Construction and Pose Estimation.

The PnP Problem.

Once initialization has been performed, subsequent iterations of structure and motion estimation build iteratively on the initial map created. In an image, if a set of 2D points is found with known 3D locations, calculating the potential camera pose to view the subject points in their current orientation is referred to as the PnP problem. The required relative camera translation in Figure 22 is also no longer a constraint since pose is estimated with the PnP problem from only one image.

The points used for initialization now have a 3D location associated with each specific 2D point. Continuing the tracking process to the next frame yields a new set of 2D image coordinates for points with known 3D locations. To estimate the new camera pose based on these 2D-3D correspondences, an iterative implementation of the PnP problem based on Levenberg-Marquardt optimization is utilized and requires at least three points. In this implementation, the function finds a pose that minimizes reprojection error, essentially performing bundle adjustment on only one frame. Finally, RANSAC is applied to the PnP problem as a method of eliminating points in which the reprojection error exceeds a user defined threshold. The motion-based RANSAC method succeeds in eliminating points that may experience drift in special cases, such as following a surface reflection rather than a point on an object,



(a) (b)

Figure 25. Updated Camera Poses Based On Initialization Point Cloud.

as long as at least roughly 50% of the remaining points are motion inliers. Points are eliminated in this implementation with a reprojection error greater than 4 pixels. Figure 25 shows subsequent pose estimates through relative motion in blue, all based on the initial set of projected 3D points by solving the PnP problem.

Projecting Newly-Detected Points.

The initial point cloud, in the case of Figure 25, can only continue to provide pose estimates while the 2D point tracks remain in view. As these points become occluded due to rotation, the track points end and there are no 2D image points to 3D model point correspondences to provide a pose estimate. To solve this issue, as new points become visible and tracked through multiple views, they are continuously projected into 3D space using triangulation. These updated point projections con-

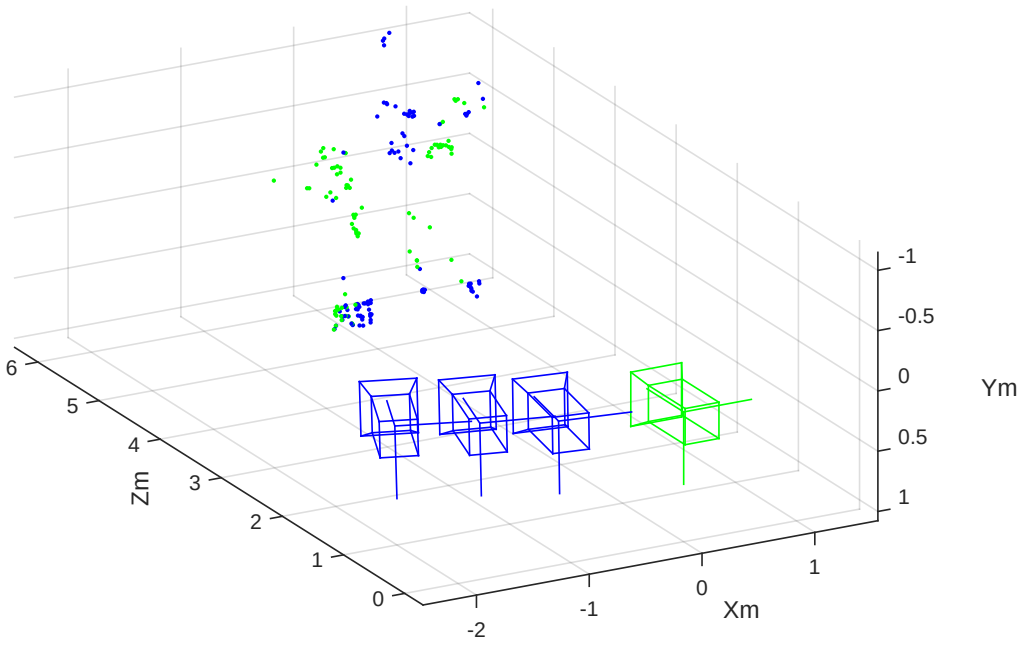


Figure 26. Initial Point Cloud and First Subsequent Point Projection.

tinue to provide a 3D source for pose estimates when the initialization point cloud has disappeared. Figure 26 shows the initial point cloud in blue and the first subsequent projection from newly-detected points through rotation in green. Continuing the process of performing pose estimates and projecting points as they exist over multiple images results in continuous construction of the structure of the model and knowledge of relative pose. Chapter IV shows the results of the continuation of the SLAM process.

Selecting Points for Projection and Removing Additional Outliers.

As it turns out, the point detection process and KLT tracker result in hundreds of points being detected in a scene. However, each of these points is not utilized for SLAM. Numerous outlier methods previously described pare down the tracked points over a portion of a video sequence. The BRIEF feature descriptor is utilized to remove KLT points which have undergone drift. Applying RANSAC to the PnP

problem removes points with greater than a certain reprojection error threshold. Still, this may not solve the problem of a dynamic background (the Earth) and results in an overwhelming number of points for calculation.

A very simple, yet very effective, method found to solve the dynamic background problem is to track the number of frames a point is visible. The number of frames that a point track has existed will be referred to as the *point life*. Utilizing a method that selects points with a life greater than a certain threshold, yet have survived the previous outlier removal techniques, ensures that only the most sustained point tracks on the object will be used for calculations. Tracks in dynamic backgrounds are typically short lived. This point life method results in dynamic background points being ignored. Combined with the region of interest method in Section 3.2.2, backgrounds are successfully isolated from the object of interest in the frame.

The point life threshold is assigned based on the rate of relative motion. Automated initialization based on initial track point motion results in two pose estimates. The amount of translation between the first two pose estimates serves to provide a value for the current rate of relative motion. Restricting triangulation to points that exist through pose estimates, with a change in position at least equal to the current motion rate, succeeds in isolating points to the target.

Figure 27 shows track points detected on the Space Shuttle performing rendezvous pitch maneuvers with the Earth in the background. Since motion is slow, the motion-based point life threshold will be large, isolating points to the Space Shuttle. In Figure 27, all tracked points are shown by yellow crosses. Track points with a life greater than the motion-based point life threshold (which is updated each iteration based on the rate of motion) are emphasized with a cyan circle around the cross.

Another benefit of using point tracks that meet a point life threshold is a reduction in error. Each iteration of estimating pose with PnP, detecting new points, and

Figure 27. Isolating Targets From Dynamic Backgrounds Using Point Life.

projecting to 3D results in an increase in overall error. Maximizing the track point life results in utilizing a set of points over more frames for pose estimation. This reduces error by minimizing the number of times new points are detected, projected, and used for pose estimates.

Using these methods, a few outliers may still sneak through the cracks. The resulting projections to the point cloud typically appear visibly to be incorrect. Some examples are a 3D point at an abnormally large depth or a negative depth (behind the camera). At this stage of the process, if bad points do exist, they are usually very few. Performing basic statistics on the location of the 3D points is an effective method of removal. Points are removed with a distance greater than three standard deviations from the center of the point cloud. Points are also removed with a negative depth. The 2D point tracks are also ended in these infrequent cases.

3.3.3 Bundle Adjustment Integrated with SLAM.

The First Increment of Bundle Adjustment.

The accuracy of 3D point locations is limited based on image resolution, image noise, track point drift, and errors during the fundamental matrix estimation process.

Performing bundle adjustment after initialization and one additional pose estimate succeeds in reducing this error. $\mathbf{P_1}$ and $\mathbf{P_2}$ are estimated during initialization. A third pose, $\mathbf{P_3}$, is selected based on the translational distance between the current pose estimate and $\mathbf{P_2}$. Once the current camera translation has at least exceeded the distance between $\mathbf{P_1}$ and $\mathbf{P_2}$, the third pose estimate, $\mathbf{P_3}$, is included in an error reduction operation using the cost function in Equation 23 (without optimizing $\mathbf{K}$). In this manner, three different perspectives provide the most accurate initialization with the data available without a superfluous use of computational resources on a larger number of images. Error reduction on initialization is incredibly important to prevent additional propagation of error to subsequent structure and motion estimates.

$$\min_{\mathbf{P}_{1:3},\mathbf{X}^i} \sum_{j=1}^{3} \sum_{i=1}^{m_j} \text{dist}(\mathbf{x}_j^i, \mathbf{P}_j \mathbf{X}^i) \tag{23}$$

The C/C++ package for generic sparse bundle adjustment called SBA is implemented for this application [26]. Using SBA provides numerous benefits: a well documented software package with instructions, implementation examples, and a MATLAB MEX interface that runs more efficiently external to MATLAB.

Incremental Bundle Adjustment.

Additional increments of bundle adjustment can be performed to minimize error throughout the SLAM process. Many different strategies exist for error reduction. Bundle adjustment implementations range from optimizing parameters across two sequential images, to a global manner over all possible images. The bundle adjustment implementation strategy is very important as processing time can escalate quickly as the number of nonlinear equations grows. Tailoring the bundle adjustment approach for specific applications is very important and is another candidate for future work. Currently, this implementation does not utilize bundle adjustment following the first

three pose estimates (except for the iterative PnP method to solve for camera pose between only two frames).

3.4 Method Summary

Algorithm 1 provides a summary for the specific implementation of the SLAM method described.

Algorithm 1 Monocular SLAM Method Summary

Load video, extract two frames, find target ROI
Detect corners in ROI using FAST and extract BRIEF features in frame 1
Create KLT tracker
Set mode to initialization, initialize threshold variables
for $i = 2$ to the total number of frames **do**
  Update current video frame
  Use KLT tracker to find new point locations
  Extract BRIEF features from current frame
  if current-library feature Hamming distance is > threshold **then**
    End point track
  end if
  if a BRIEF feature has not been updated within the feature update rate **then**
    Update library BRIEF feature to current feature
  end if
  if initialization mode **then**
    Calculate relative point track motion
    if relative point track motion > initialization threshold **then**
      Estimate $\mathbf{F}$ using Norm-8-point, convert to $\mathbf{E}$ using $\mathbf{K}$
      Decompose $\mathbf{E}$ to extract $\mathbf{R}$ and $\mathbf{t}$
      Triangulate the same points used for $\mathbf{F}$ to 3D
      Set mode to PnP, set motion threshold and feature update rate
    end if
  end if
  if in PnP mode & motion threshold is reached **then**
    Find visible 2D track points that match a projected 3D point
    Solve PnP with RANSAC to estimate updated pose ($\mathbf{P_n}$), remove outliers
    Find 2D track points without a 3D projection that existed in $\mathbf{P_{n-1}}$
    Triangulate points to 3D using $\mathbf{P_n}$, $\mathbf{P_{n-1}}$, $\mathbf{x}_n^i$, and $\mathbf{x}_{n-1}^i$
    Clean up bad points, update motion threshold and feature update rate
    Incremental BA if desired
  end if
  if the number of pose estimates == 3 **then**
    Perform BA to optimize motion ($\mathbf{P_{1:3}}$) and structure ($\mathbf{X_{1:3}^i}$) estimates
  end if
  for $k = 1$ to the number of ROI sub-regions **do**
    if point density in sub-region < threshold value **then**
      Detect new points in sub-region using FAST and extract BRIEF features
    end if
  end for
  Update the KLT point tracker to the remaining inliers
end for

IV. Results and Analysis

The results in this chapter were generated by applying Algorithm 1 to various video sequences. Section 4.1 presents the results generated from space-related video sequences. This data was used for a significant portion of the development of this monocular SLAM algorithm to ensure its operation on data generated during prox-ops. No quantitative accuracy analysis is performed in this section as no truth data is available. However, a qualitative inspection yields visibly successful structure and motion estimations. Section 4.2 shows results generated from a CubeSat air bearing testbed. The air bearing tests provide a known orientation for results verification. Section 4.3 provides a similar verification of results from a series of synthetic video sequences with known camera orientation. A run-time analysis and parameter sensitivity analysis are performed in Sections 4.4 and 4.5.

4.1 Results of Space-Related Video Sequences

4.1.1 HOMER.

Texas A&M University's Aerospace Engineering Department designed an indoor proxops simulator called the Holonomic Omni-directional Motion Emulation Robot (HOMER) [20], which provides ideal relative motion and video sequences for testing a space-customized monocular SLAM algorithm. HOMER data is advantageous for development and testing because it provides smooth and untethered motion using space-representative hardware and lighting. Figure 28 shows ten sample frames from the HOMER video sequence that was used extensively for monocular SLAM development. In this sequence, the smooth, reflective surfaces and target rotation result in a challenging scenario for maintaining accurate point tracks. Figure 29 shows the SLAM results, in which the green camera model represents the initial camera pose

with each subsequent camera pose estimate represented by the blue camera models. Figure 30 shows a close-up and different perspective of the estimated 3D point cloud with color variations by location to show depth.

In the HOMER video sequence, one unique aspect is that the camera is stationary while the rotation of the target provides the change in perspective required for structure and motion estimation. This is unique in the fact that the relative rotation and translation of the camera is a result of only a rotation by the target. A very important note is that if the roles were reversed and the only relative motion is pure rotation by the camera, structure and motion estimation is impossible as there is no gain in the additional perspective information required. One final note is that performing SLAM on this sequence requires the elimination of any background points that are detected, as the results will be grossly erroneous in an attempt to match one relative motion solution to two separate motion models.

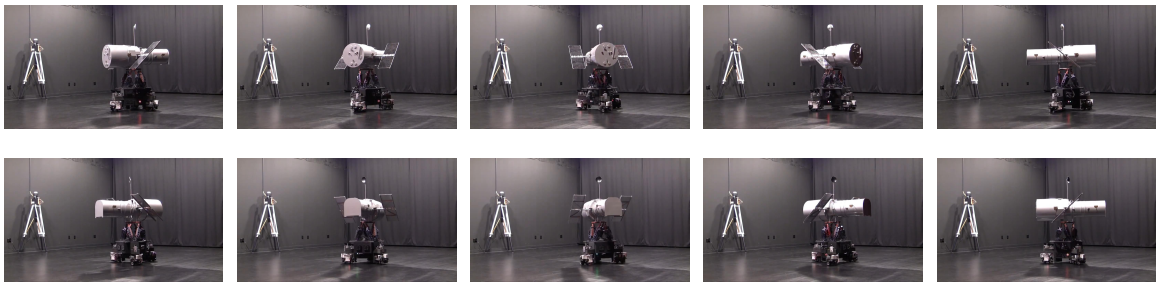


Figure 28. HOMER Video Sequence [20]: Sample Frames.

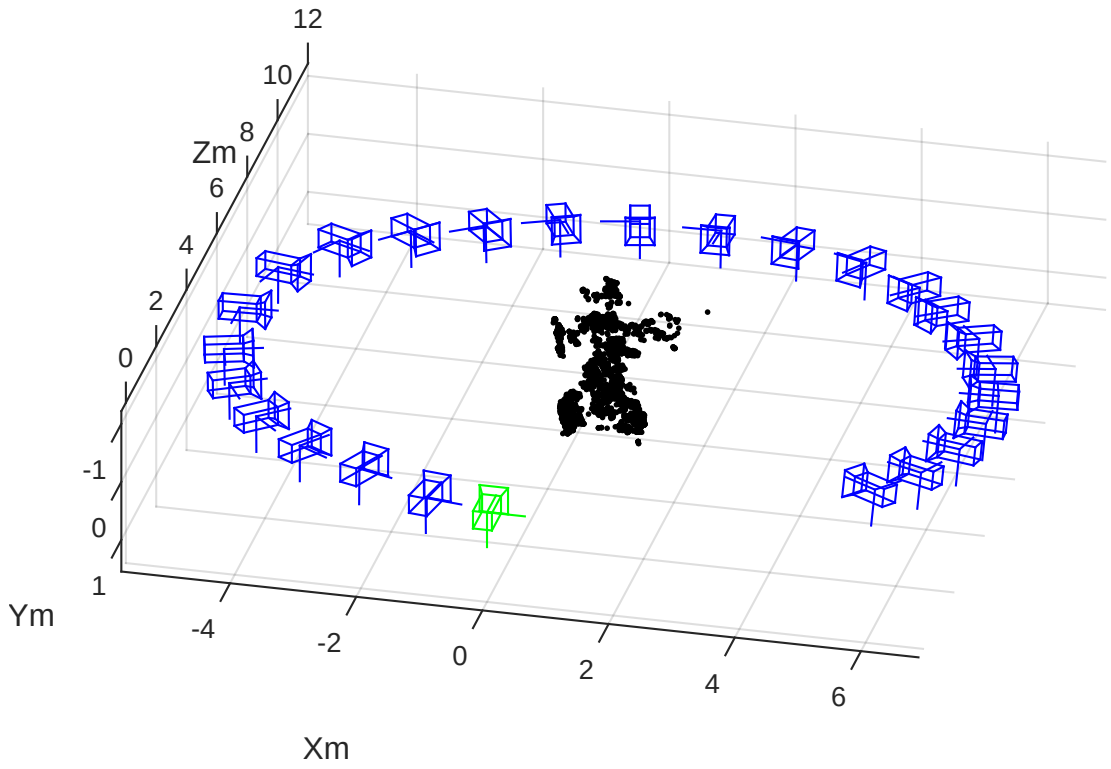


Figure 29. HOMER Video Sequence [20]: SLAM Results.

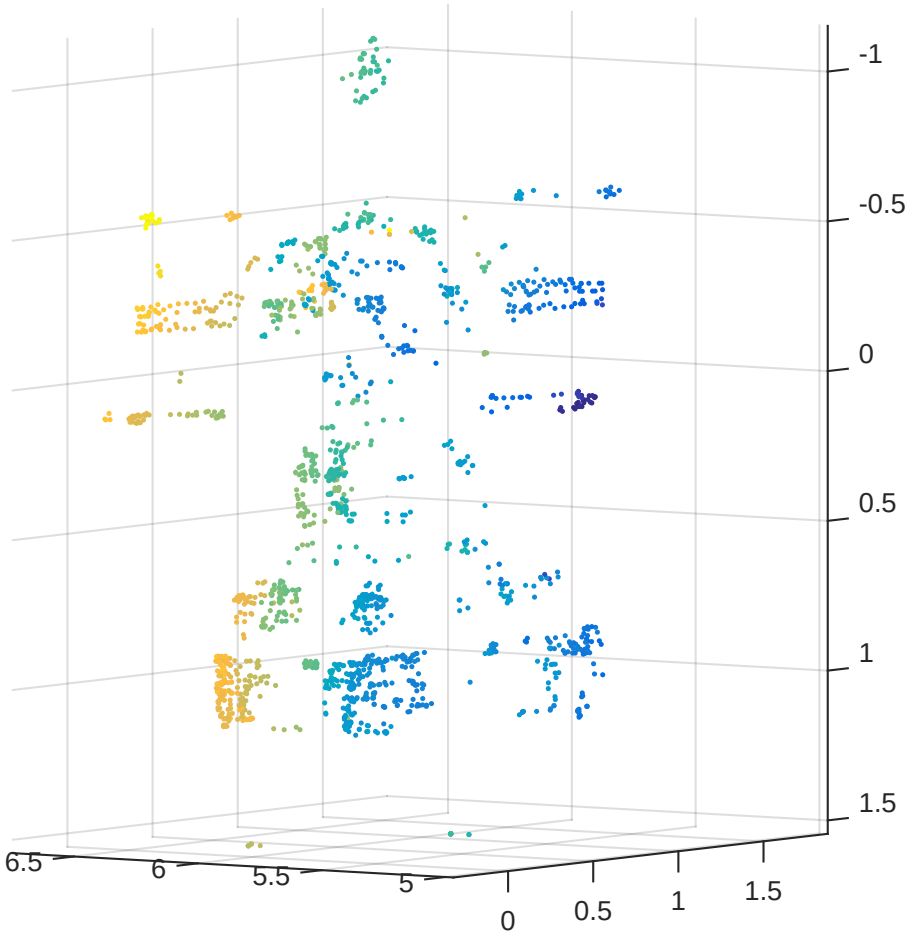


Figure 30. HOMER Video Sequence [20]: Point Cloud.

4.1.2 Orbital Express.

Orbital Express is an on-orbit rendezvous mission managed by DARPA that launched two spacecraft in 2007. The purpose of the program was to develop "a safe and cost-effective approach to autonomously service satellites in orbit."[34] The first spacecraft, represented as the larger spacecraft in the foreground of Figure 31, is a servicing satellite called the Autonomous Space Transport Robotic Operations (ASTRO). The other spacecraft is the Next Generation Satellite and Commodities Spacecraft (NEXTSat), which serves as the target spacecraft for ASTRO's proxops maneuvers. DARPA continues to share still image and video clips from this satellite's on-orbit missions for research purposes [33].

The first set of results related to the Orbital Express mission is a computer simulation of ASTRO and NEXTSat rigidly connected together [33]. Ten frames from the video are shown in Figure 32. Figure 33 shows the SLAM results, including the pose estimates through relative motion and the structure points on the spacecraft. The green camera model represents the first pose, with subsequent pose estimates shown in blue. Figure 34 shows a larger and different perspective of the structure point cloud estimated in Figure 33. Since only one side of the spacecraft is seen in the simulation sequence (through less than a quarter of a full rotation), points only exist on one side of the model.

The next set of Orbital Express data is a composition of still images taken of NEXTSat from the ASTRO spacecraft during a decoupling maneuver [33]. Figure 35 shows ten sample frames from the sequence. Figure 36 shows the SLAM results and 37 shows a side perspective of the structure point cloud. In this sequence, the solar panel provides the best visual features for tracking, which account for the majority of the point cloud. SLAM ends relatively soon as the DARPA video is low resolution (320 x 320 pixels), which leaves very few features for tracking as depth to the target

Figure 31. Orbital Express Concept [33].

increases.

The final set of Orbital Express data is captured from a camera on ASTRO's robotic arm during a self inspection procedure. Figure 38 shows ten sample frames from the video sequence captured by the arm. Figure 39 shows the SLAM results, with the camera models representing the relative pose estimates of the camera on the arm. Figure 40 shows another perspective of the point cloud. Additional video sequences of the arm exist; however, quick movements and numerous sequential estimations from close-up views cause significant errors to propagate and build up over time. Increasing robustness and reducing error over a long sequence like this is a useful topic to pursue in future work.

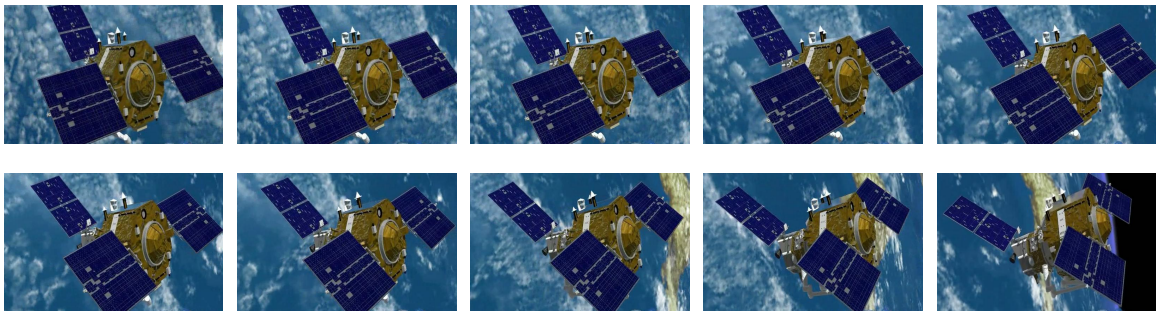


Figure 32. Orbital Express Simulation [33]: Sample Frames.

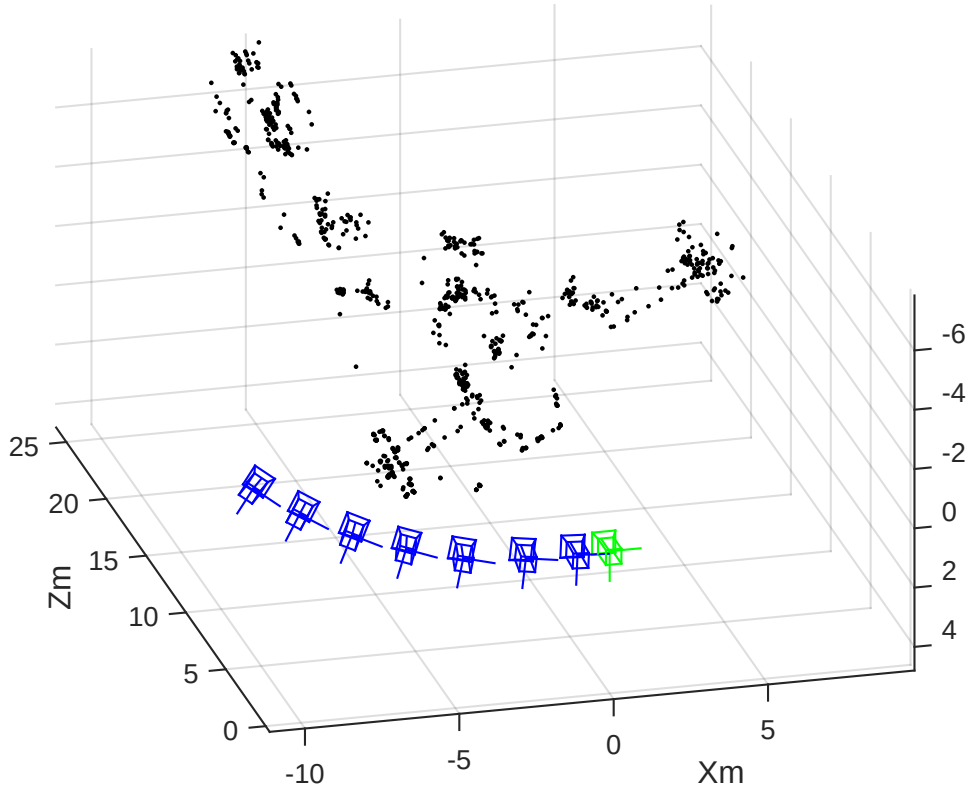


Figure 33. Orbital Express Simulation [33]: SLAM Results.

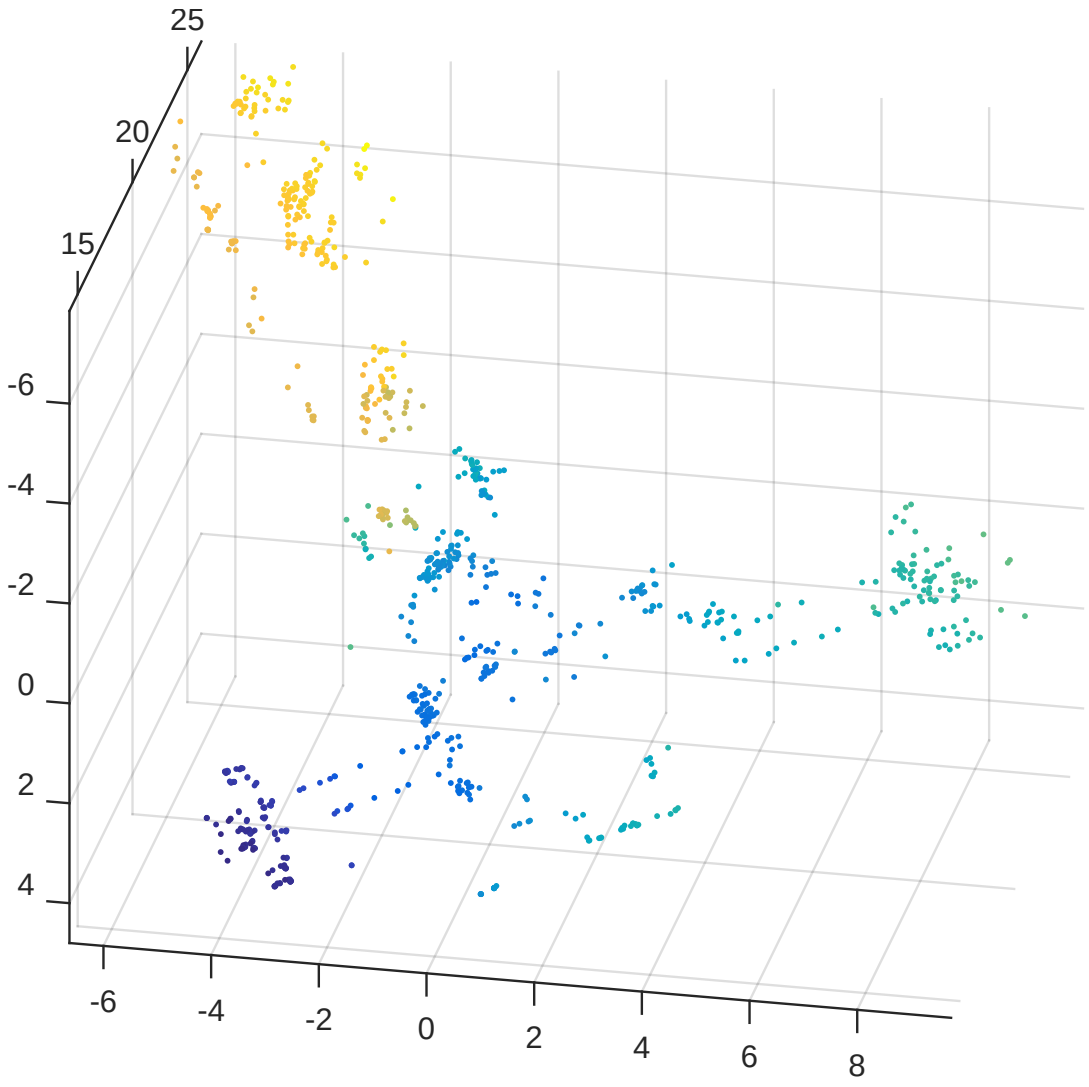


Figure 34. Orbital Express Simulation [33]: Point Cloud.

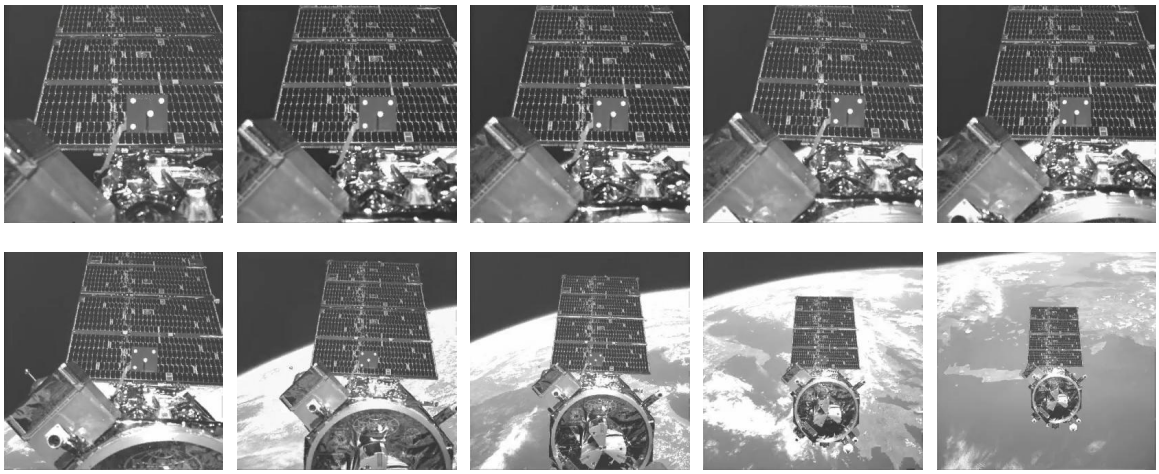


Figure 35. Orbital Express Proxops Video [33]: Sample Frames.

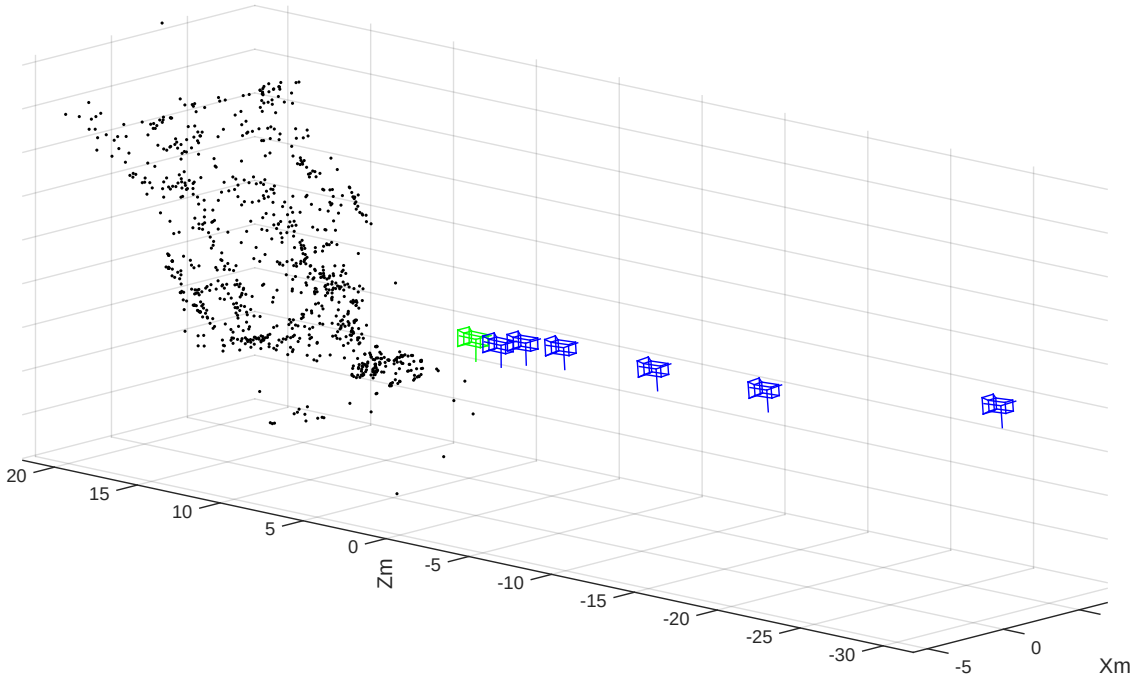


Figure 36. Orbital Express Proxops Video [33]: SLAM Results.

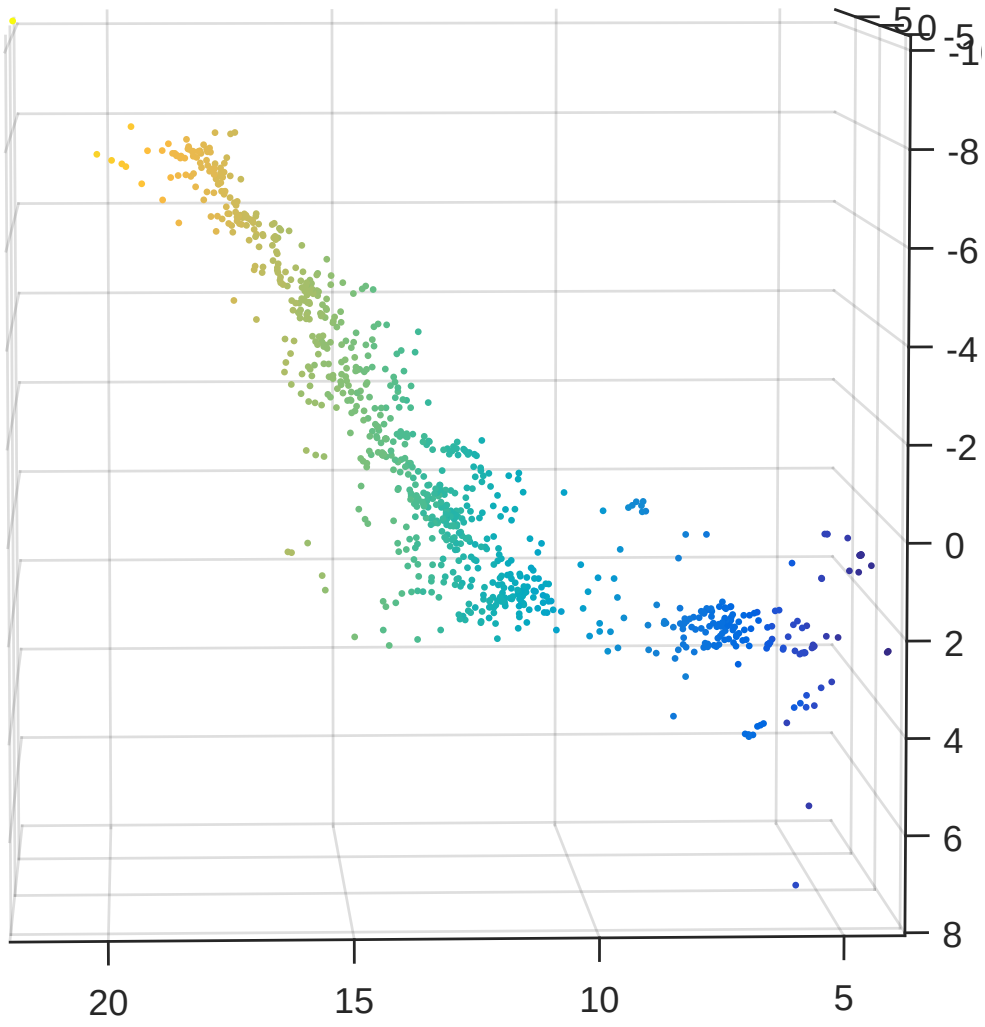


Figure 37. Orbital Express Proxops Video [33]: Point Cloud.

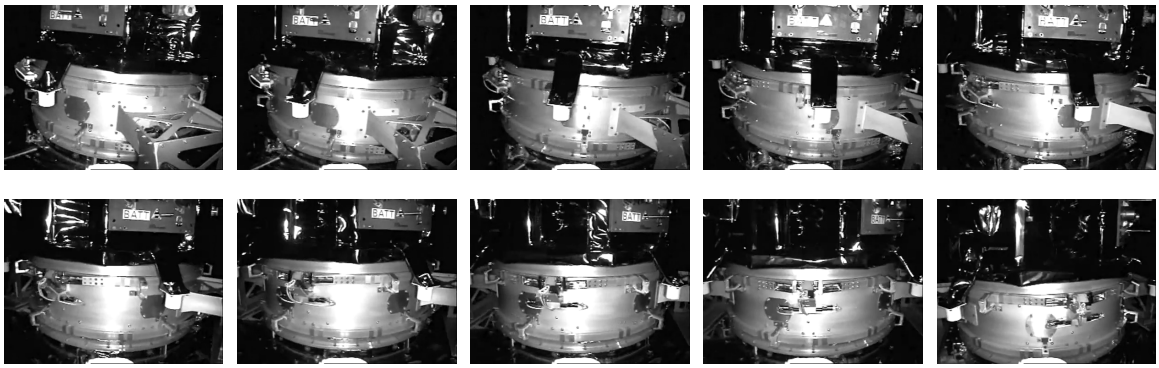


Figure 38. Orbital Express Self Inspection Video [33]: Sample Frames.

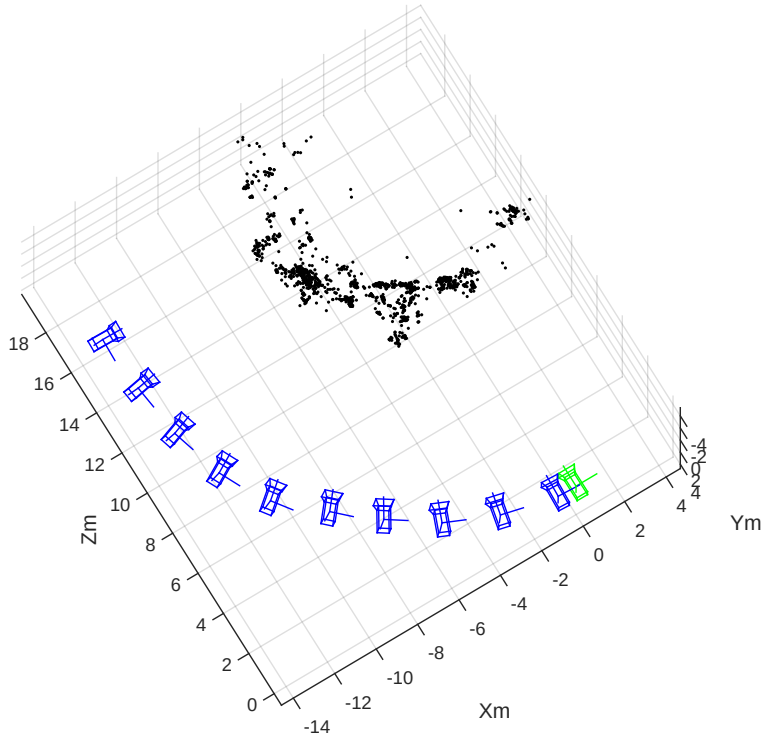


Figure 39. Orbital Express Self Inspection Video [33]: SLAM Results.

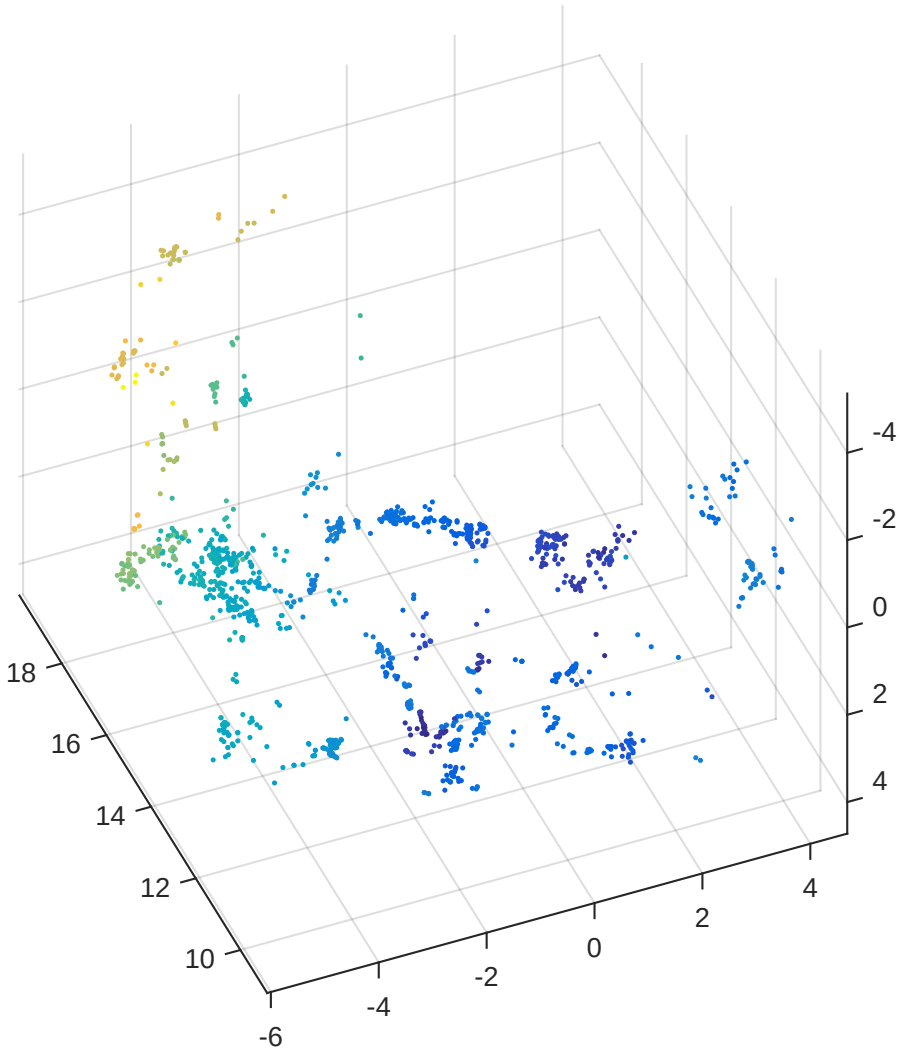


Figure 40. Orbital Express Self Inspection Video [33]: Point Cloud.

4.1.3 PRISMA.

PRISMA is a proxops mission designed by the Swedish Space Corporation for on-orbit guidance, navigation, and control demonstrations [37]. Similar to the Orbital Express mission, PRISMA consists of two spacecraft: one that is advanced and highly maneuverable called Main, and a second smaller spacecraft without maneuvering capability called Target which is stabilized by means of magnetic control only. As the names suggest, Main performs proxops around Target. The SSC has shared a selection of relatively low-resolution proxops videos from the Main spacecraft toward Target during proxops maneuvers. Figure 41 shows ten sample frames from the most dynamic proxops sequence shared, including an approach, partial orbit, and recede maneuver. Figure 42 shows the SLAM results, and Figure 43 shows the point cloud of the Target spacecraft structure estimates.

Rapid scale changes and varying levels of relative motion in the PRISMA sequence make this a difficult sequence to maintain accurate point tracks. During the approach portion of the video, significant scale changes occur due to large relative linear motion. However, during the middle portion of the video, relative motion and scale changes are small, but target rotation is large. Finally, during the recede operation, relative motion occurs extremely fast. The varying motion rates in this sequence contributed significantly to automating the rates at which the library BRIEF feature updates and pose estimates occur.

Jumpiness due to (what appears to be) dropped frames is also an attribute of this video sequence that is yet to be completely solved. The result is a significant amount of lost point tracks which are still in view, followed by point re-detection in subsequent frames, which introduces additional error and no correlation to the point's initial detection.

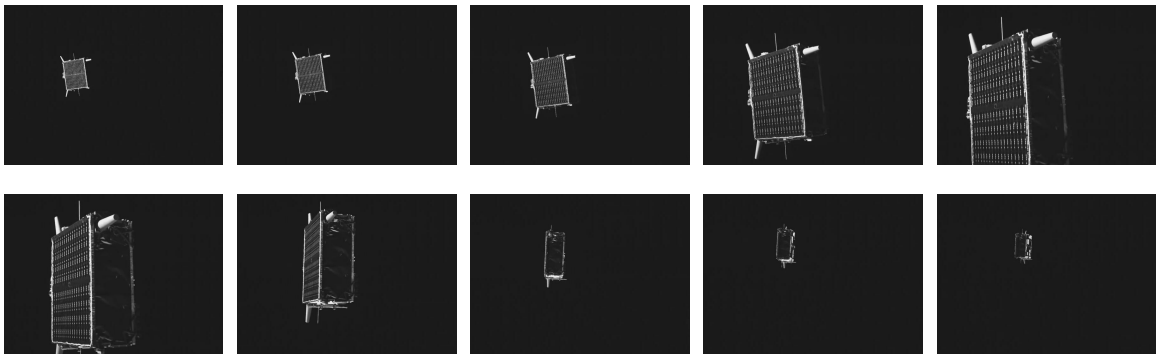


Figure 41. Prisma Video Sequence [37]: Sample Frames.

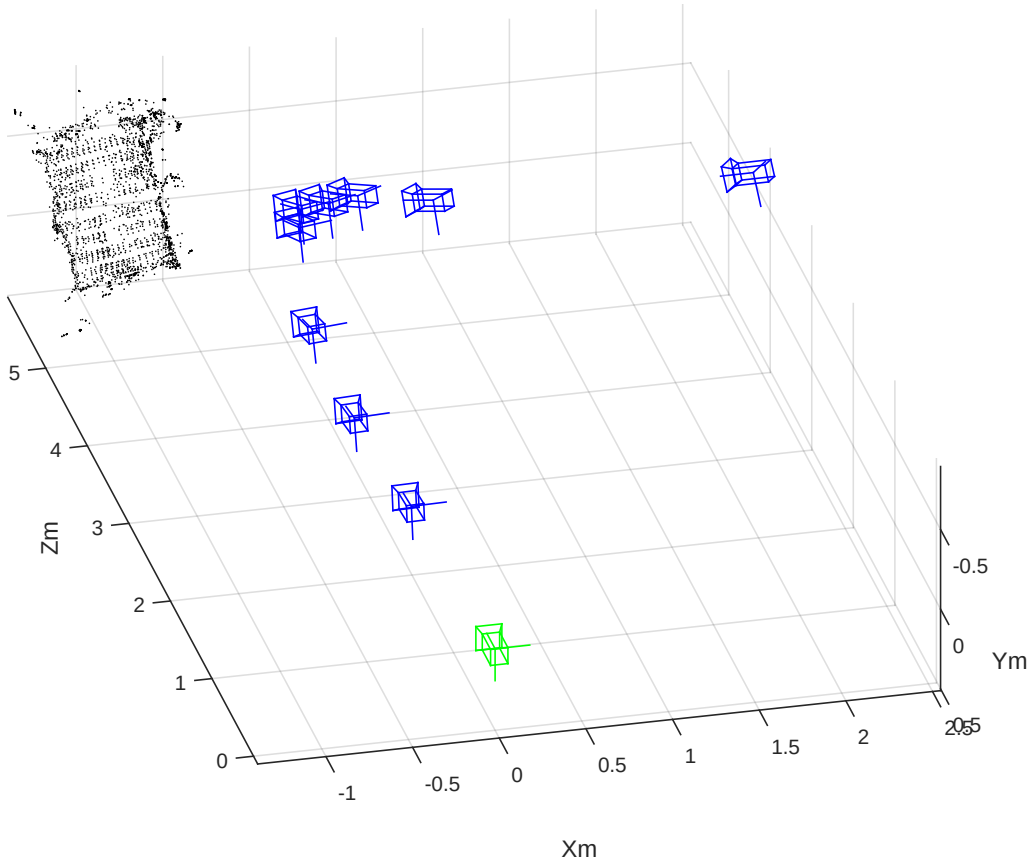


Figure 42. Prisma Video Sequence [37]: SLAM Results.

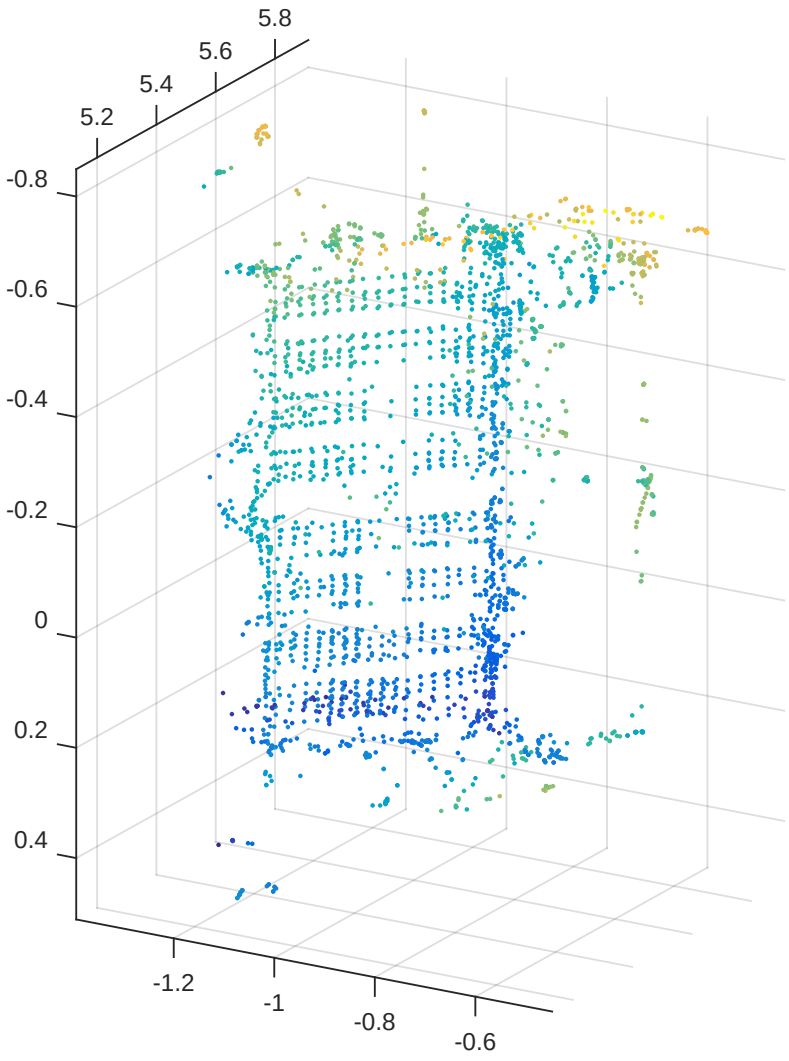


Figure 43. Prisma Video Sequence [37]: Point Cloud.

4.2 Accuracy Analysis Using a CubeSat Air Bearing Testbed

Determining accuracy of a computer vision algorithm can be challenging. Early in the design stages of this SLAM algorithm, success was based on generating a reconstruction that simply *looked like* the the object in question. With the synthetic sequences in Section 4.3, truth data is very easy to come by as an output from the modeling software used to generate the scenes. With other sequences analyzed, such as actual space based proxops data, a set of truth data may be impossible to obtain if the specific sensors on-board the spacecraft do not provide the required data at a reasonable level of accuracy. An accuracy analysis approach is therefore used in which video sequences are acquired in a laboratory setup that allows for the measurement of truth rotation values.

A commonly used test platform at AFIT is the air bearing, which creates a near frictionless environment. A typical test setup involves the AFIT 6U CubeSat chassis mounted to the air bearing inside a Helmholtz cage to simulate on-orbit magnetic fields. This platform allows for the development of the CubeSat's attitude determination and control sub-system, enabling unlimited rotation about the vertical axis and approximately 60 degrees in the second and third axes. Figure 44 shows the CubeSat-air bearing testbed.

Performing a series of tests constrained to rotate mainly about the vertical axis allows for trivial truth measurements to be taken throughout the sequence of each test. The SLAM algorithm is then used to estimate the relative pose (position and orientation) of the camera with respect to the CubeSat throughout the test sequence. Performing a series of coordinate transformations allows for the extraction of the vertical axis rotation of the CubeSat, which is compared with the truth vertical axis rotation values to determine accuracy of the SLAM method. Truth rotation values are obtained from the spot of a laser pointer that is fixed to the CubeSat and points

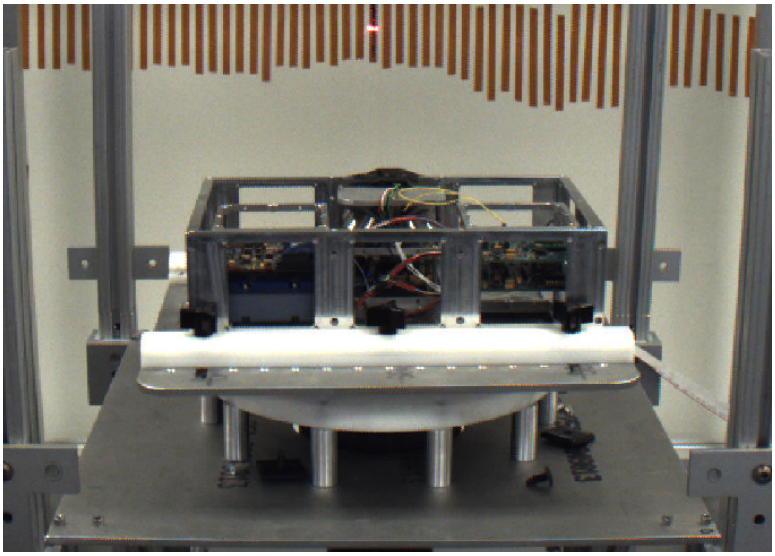


Figure 44. The AFIT 6U CubeSat Air Bearing Testbed.

to degree increments on the lab wall. The accuracy of the truth rotation data is $\pm 0.25^\circ$ [43].

The first test performed involves a clockwise (CW) vertical axis rotation from 0° to approximately 45°, followed by a roughly 90° counter-clockwise (CCW) rotation, and finally a CW rotation toward the starting position. Figure 45 shows the SLAM scene results, with a coordinate axis drawn on the CubeSat point cloud, and the coordinate axes for each estimated camera pose shown. The black arrows show the rotation sequence described above as relative rotation and translation from the camera's perspective. Figure 46 shows the truth and SLAM-estimated vertical rotation. In this test, it appears each pose estimate introduces slight error that builds up over time. At the end of the test sequence, the truth vertical axis rotation is -26.5° and the SLAM estimated vertical axis rotation is -28.1°. The result is an average error per pose estimate of 0.10° with a standard deviation of 0.53°. Analyzing the quantity of error introduced by each pose estimate yields a root mean squared (RMS) value of 1.05°. In Figure 48, these error values correspond to test sequence 1 where the

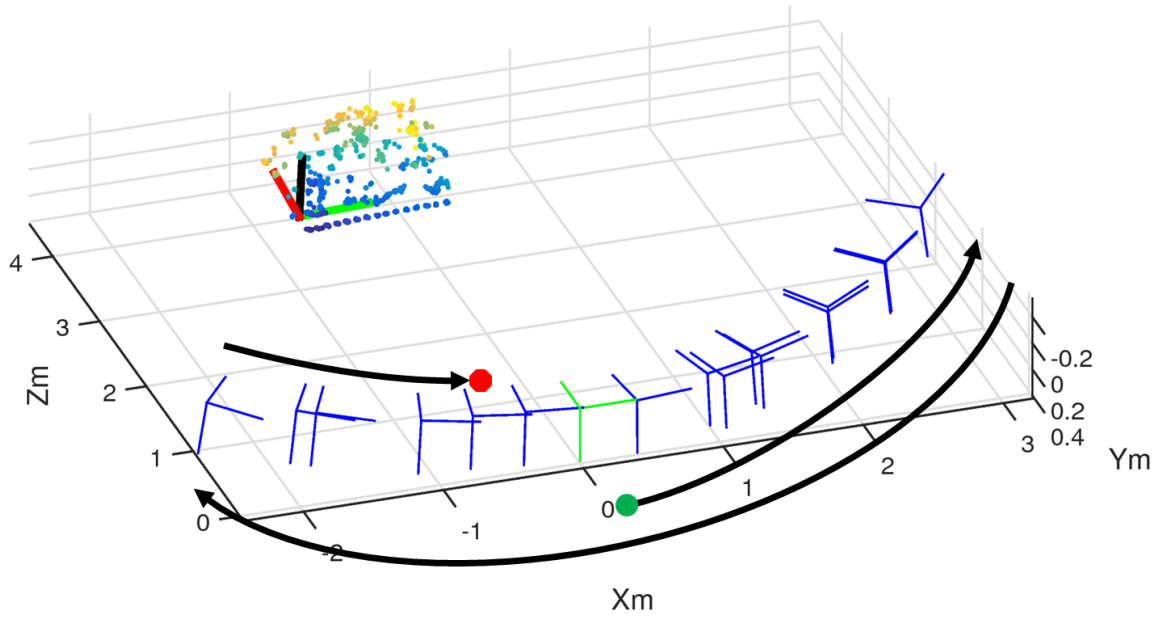


Figure 45. SLAM Results of a Single Axis CubeSat Rotation Test.

green squares represent the average error, the red error bars represent the standard deviation, and the blue bars represent the RMS value.

To observe the effect of the accumulation of error over a longer test sequence, the next test results shown include additional CW and CCW rotations. Figure 47 shows the truth and SLAM estimated vertical rotation in the test sequence. This data shows that over a longer test sequence, error does accumulate and continues to occur proportional to each pose estimate.

Unfortunately, a constant error value cannot be generalized for all targets. The error is due to a number of factors, but depends on the target being tracked. Figure 48 shows the accuracy analysis results for a series of nine tests using the method described for test sequence 1. Ending track points early due to the CubeSat's reflective aluminum surfaces contribute to error propagation. As new points are detected, their baseline location is derived from a measurement that now has a slight amount of error introduced. Accurately extending track point life for all of the frames a point is in

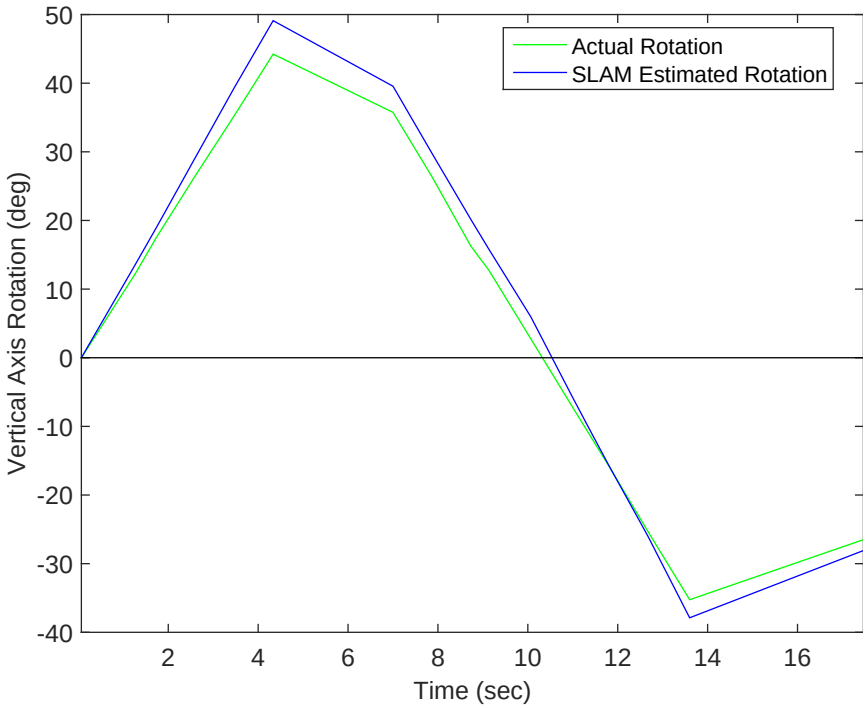


Figure 46. Single Axis CubeSat Rotation Test Results.

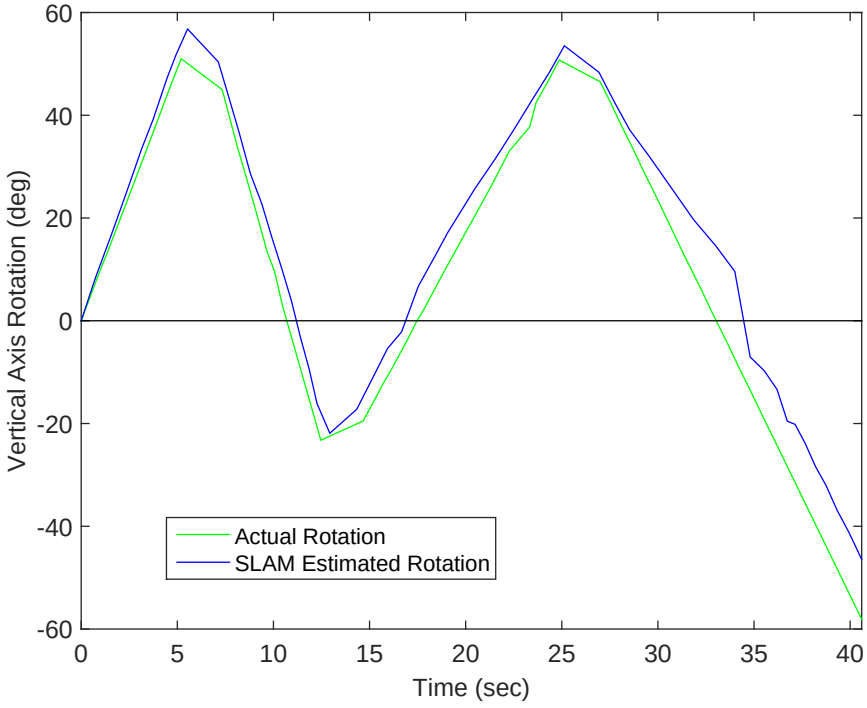


Figure 47. Single Axis CubeSat Rotation Test Results.

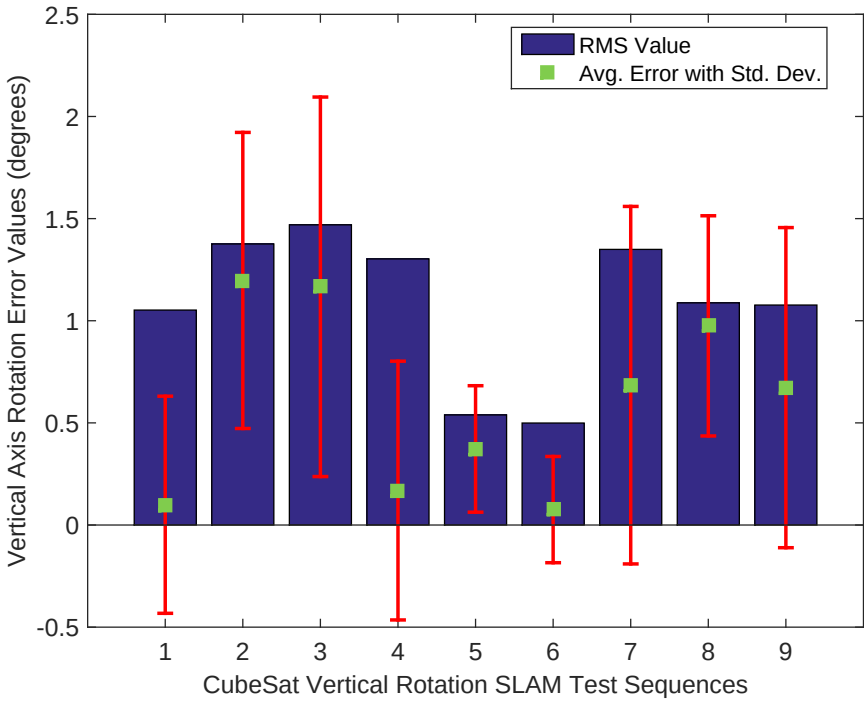


Figure 48. SLAM Accuracy Analysis of the CubeSat Single Axis Rotation Test Series.

view will reduce error and should be considered for future work.

Results also vary from test to test in Figure 48 because feature density on the CubeSat is not constant. Some areas of the CubeSat have a higher number of easily tracked features than other areas. Depending on the side of the CubeSat that is in the camera's field of view during initialization, there may be more points to track which will increase the accuracy of initialization. Improving performance during operation on targets with a small number of features to track is an important area for improvement as well.

Next, the dimensions of the estimated 3D structure point cloud are compared with the physical dimensions of the AFIT 6U CubeSat. Without a known reference distance, the units of the scene are meaningless. In this test scenario, the depth of the scene from the camera to the center of the front plane of the CubeSat is 140 cm, which is used to scale the scene. Figure 49 shows the resulting dimensions of the

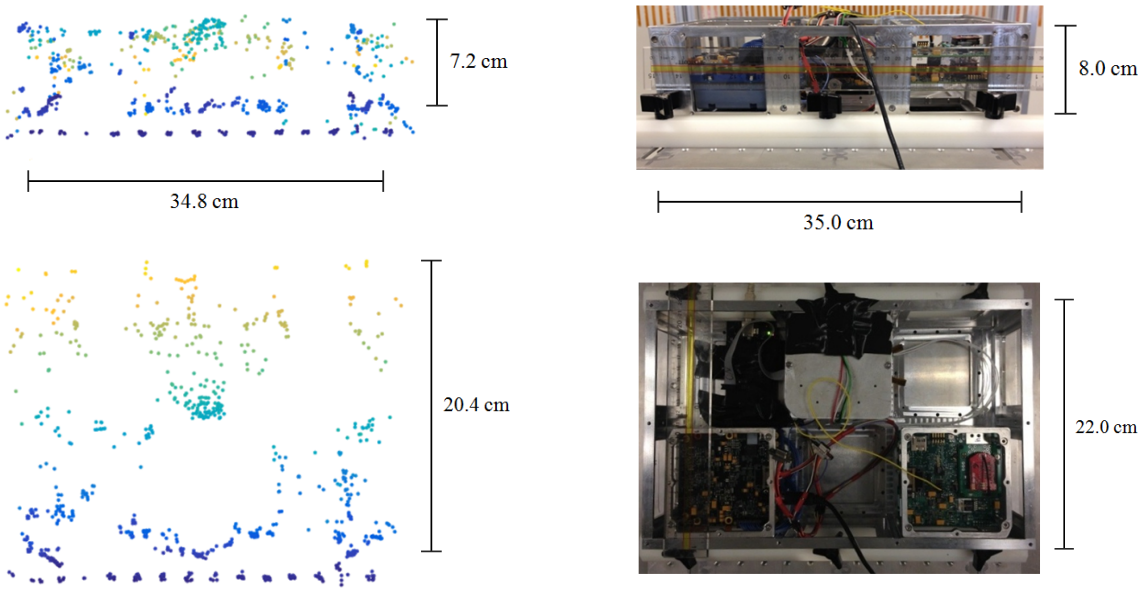


Figure 49. CubeSat Structural Point Cloud Dimensional Analysis.

estimated CubeSat point cloud on the left side, with a picture of the CubeSat and its physical dimension on the right side. The resulting error is 0.2 cm in the x-axis, 0.8 cm in the y-axis, and 1.6 cm in the z-axis. The measurements in Figure 49 yield an average error of 5.9%. This is in part due to the uncertainty associated with selecting the location on the point cloud that matches the location on the CubeSat.

4.3 Accuracy Analysis Using Synthetic Video Sequences

In [4], Balzer and Soatto share 19 synthetic video sequences created using the 3D computer graphics software package Blender. In these sequences, relative motion occurs around a synthetic object. Balzer and Soatto also publish truth camera pose data specifically for the purpose of characterizing error in CV-based structure and motion methods. Various types of motion are implemented from a simple vertical axis rotation to more complicated motion that introduces additional sinusoidal translation in other directions. Varying levels of occlusion, edge clipping, translation, and scale changes occur. Dynamic backgrounds and motion blur are included in some videos

as well, increasing the level of complexity and representative nature of a real-world video. Nevertheless, the nature of the features in the scenes and the lighting simplify point tracking in comparison to a real-world video sequence.

Figure 50 shows ten sample frames from synthetic sequence 1 which contains the simplest motion case (vertical axis rotation only). Figure 51 shows SLAM results on synthetic sequence 1. The blue camera models represent the SLAM camera pose estimates. The magenta camera models represent the known camera poses throughout the sequence. The thick red line in the foreground of Figure 51 represents the error in the position of the final pose estimate.

To analyze the accuracy of the SLAM results for synthetic sequence 1, the estimated vertical axis rotation is compared to the known vertical axis rotation. Figure 52 shows the truth and SLAM-estimated vertical axis rotation throughout the sequence. At the end of sequence 1 the SLAM-estimated total vertical axis rotation is 347.2° while the truth vertical axis rotation is 360°, resulting in 12.8° of error. Throughout sequence 1, 42 pose estimates were made. The result is an average vertical axis error of 0.31° per pose estimate with a standard deviation of 0.27°. Analyzing the quantity of error introduced by each pose estimate yields a RMS value of 0.46°. Figure 53 shows these error values for synthetic sequence 1 in comparison with the results from the remainder of the synthetic sequences. In Figure 53 the green squares represent the average error for each sequence, the red error bars represent the standard deviation, and the blue bars represent the RMS value.

Figure 54 shows the results of the SLAM method performed on synthetic sequence 15, which contains more complicated motion and multiple objects. The thick red line represents the error in the position of the final pose estimate.

Sequence 17 is the final set of synthetic results shown, incorporating complicated motion with a dynamic background. Figure 55 shows ten sample frames from this

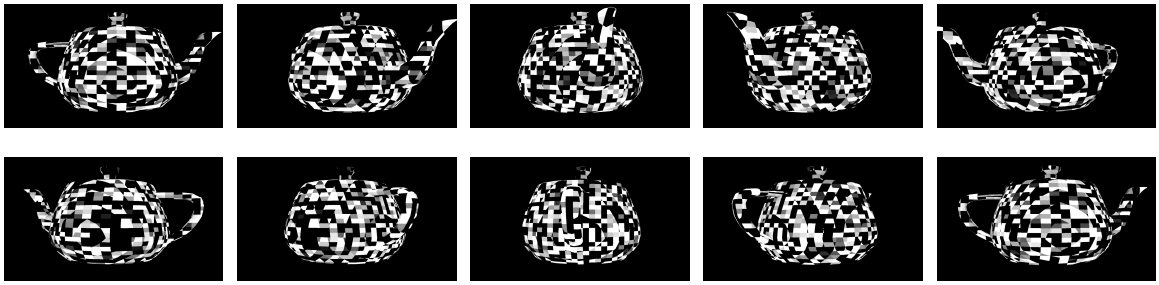


Figure 50. Synthetic Sequence 1 (Simple Motion): Sample Frames.

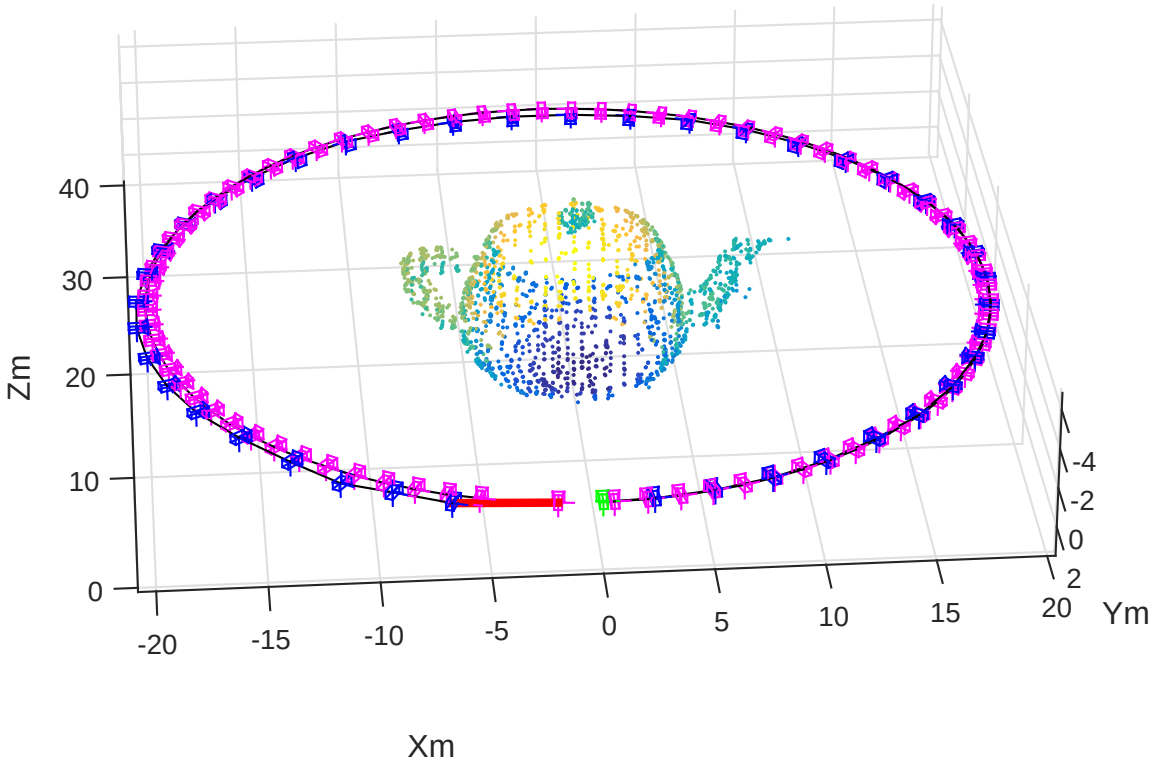


Figure 51. Synthetic Sequence 1 (Simple Motion): SLAM Results and Known Poses.

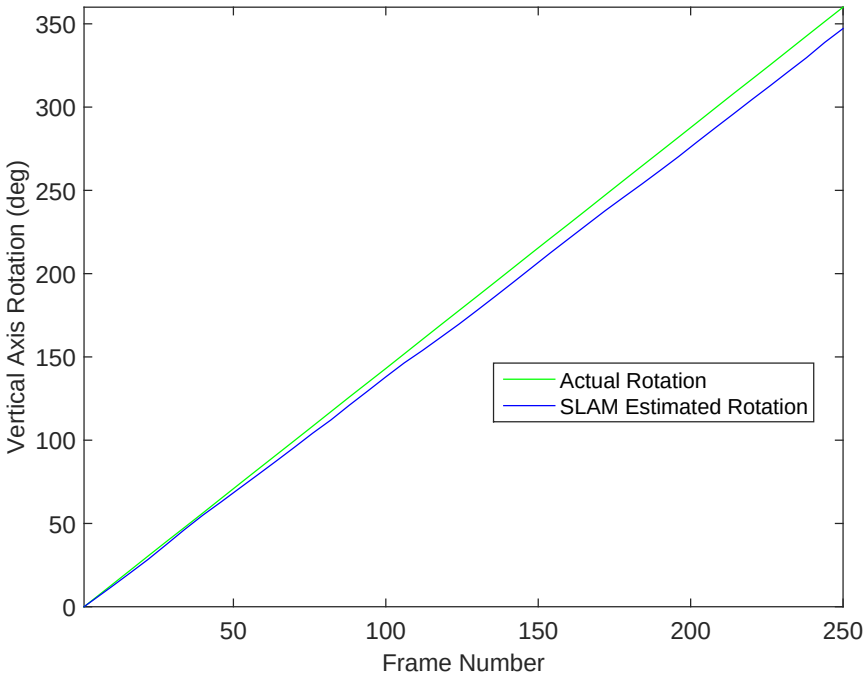


Figure 52. Synthetic Sequence 1 (Simple Motion): Vertical Axis Rotation Results.

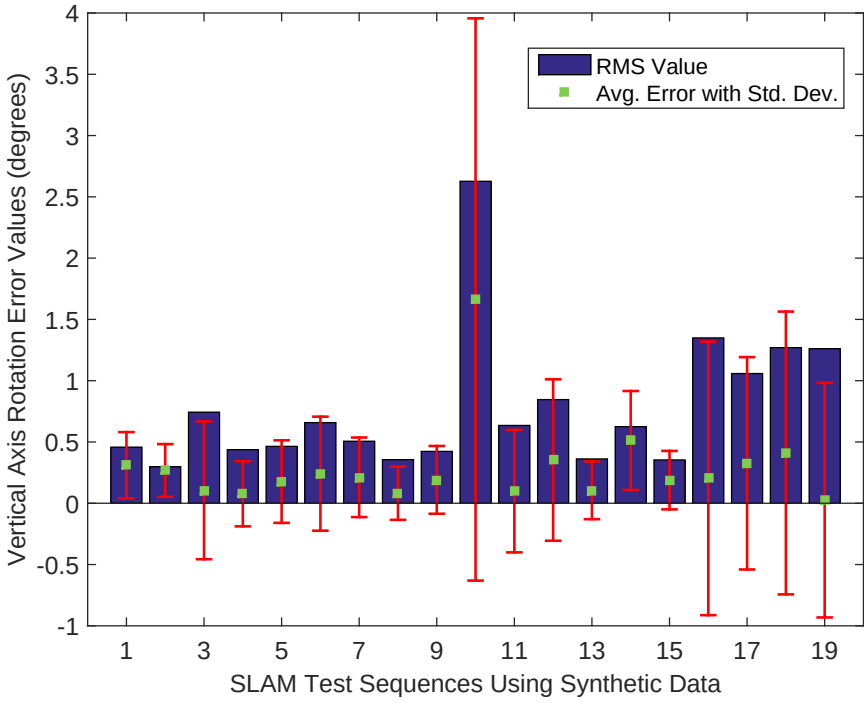


Figure 53. SLAM Accuracy Analysis of Synthetic Sequences.

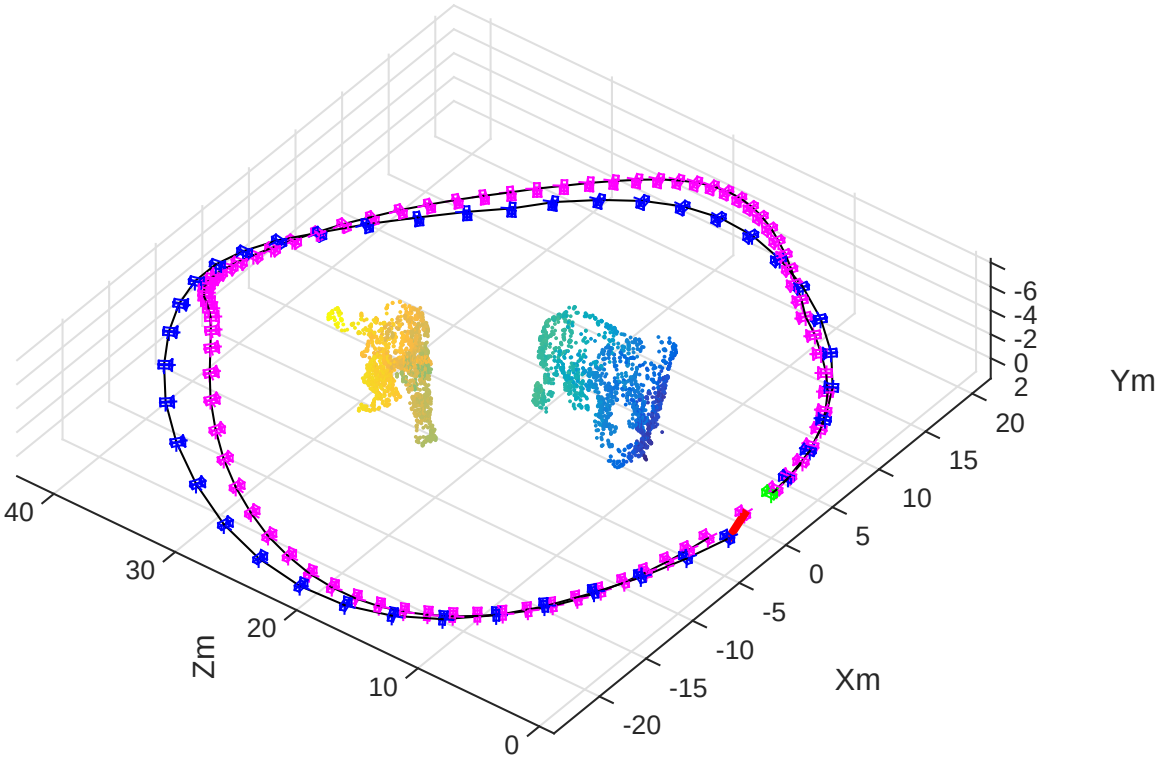


Figure 54. Synthetic Sequence 15 (Complicated Motion & Multiple Objects): SLAM Results and Known Poses.

synthetic sequence. Figure 56 shows the SLAM results and known camera poses.

Using the accuracy analysis method described for synthetic sequence 1, Figure 53 shows the average error per pose estimate, standard deviation, and the RMS value for each of the 19 synthetic sequences. The level of motion complexity increases as the sequence number increases. All sequences result in a final error between 0° and 0.5° except sequence 10. Adjusting parameters to reduce the error in sequence 10 is possible, but the default parameters are maintained for transparency of this SLAM method and to detail areas for improvement. RMS values vary more than the average error because they account for deviations from the truth in each individual pose estimate.

One way to interpret the RMS value is a representation of the accuracy of motion estimation throughout the entire sequence. The standard deviation shows that the amount of error varies throughout the sequence. Only expressing the average error at the end of the sequence does not account for deviations in intermediate pose estimates. Sequences 16 through 19, which are the most complicated, end with a small average error but the complexity of motion results in additional error in intermediate pose estimates, which is represented by a higher RMS value as well as a higher standard deviation.

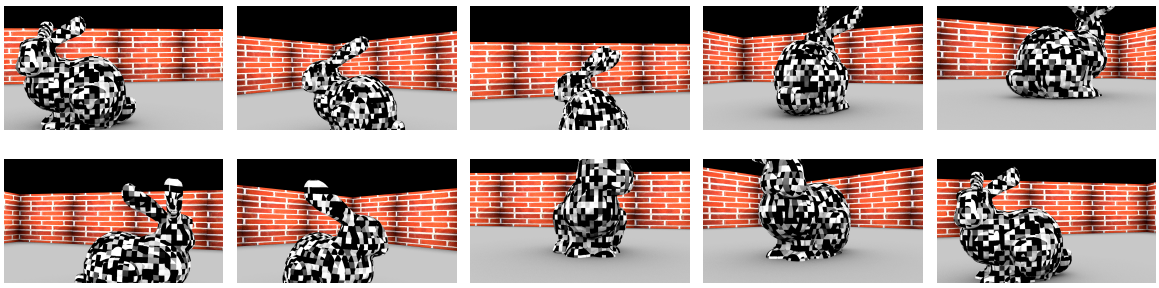


Figure 55. Synthetic Sequence 17 (Dynamic Background): Sample Frames.

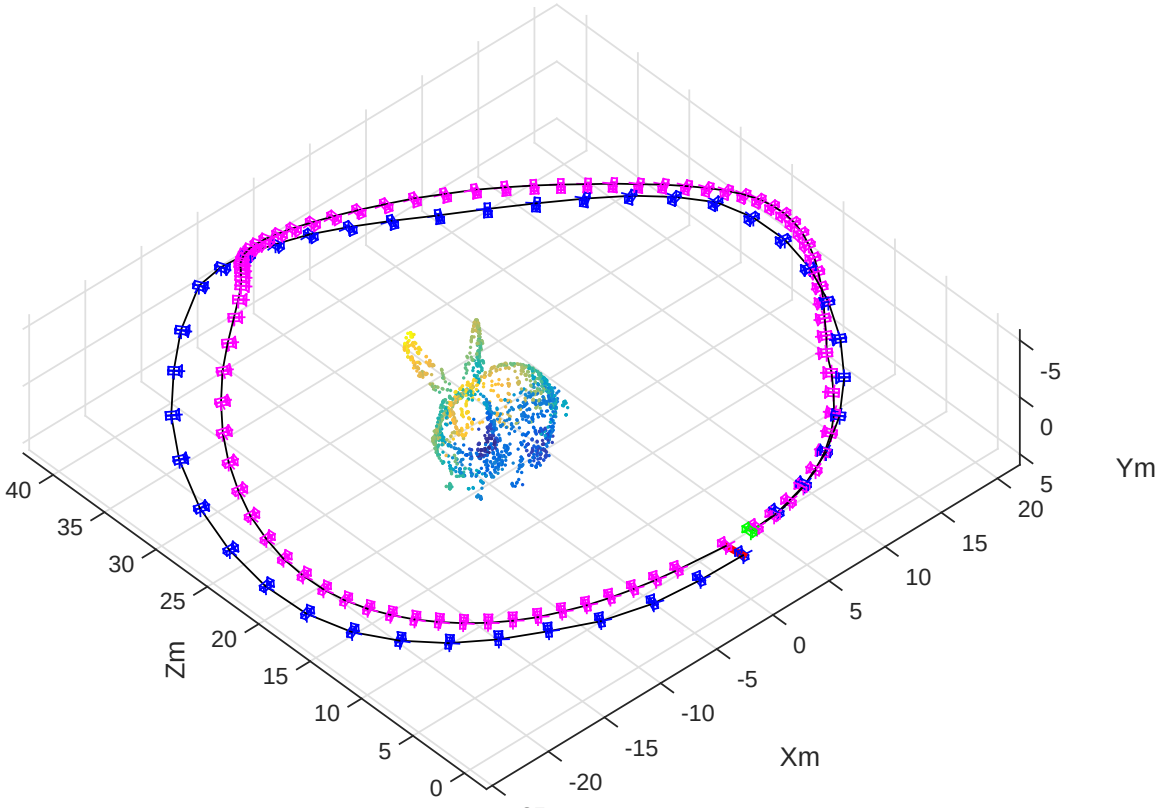


Figure 56. Synthetic Sequence 17 (Dynamic Background): SLAM Results and Known Camera Poses.

4.4 Run-time Analysis

For a selection of the video sequences analyzed, a run-time analysis is performed. Figure 57 shows the frame rate achieved for each video sequence as a separate bar, with the height representing the average frame rate over the length of the analysis. The video resolution in megapixels is also shown by the red line overlay.

The resulting frame rate correlates roughly inversely with resolution. This relationship is intuitive, as an increase in the number of pixels results in an increase in the number of corner tests to perform as well as features to track. One benefit of increased resolution is a decrease in 2D image location uncertainty for the points being tracked. This results in higher accuracy of the resulting structure and motion estimation. Unfortunately, truth data for the higher resolution videos is unavailable, so any increase in accuracy is difficult to quantify.

Other variables that affect the frame rate are feature density of the target, distance to the target, and feature density of the background. Objects with a high density of visual features are a double-edged sword. On the one hand, they provide plenty of features to track for a high confidence in motion concurrence. On the other hand, each track point requires additional computational resources. One area for future work is implementing a more efficient method of handling objects very rich in visual features other than finite thresholds for the number of tracked features in sub-regions of interest.

As distance to the target increases, the percent of the image occupied by the target and resolution of specific features decreases. The result is a decrease in the number of points tracked and an increased frame-rate, but issues may arise if not enough track points are available to perform the required motion estimates.

Another factor that potentially contributes to longer run time is the choice of programming languages. MATLAB is an interpreted programming language, which

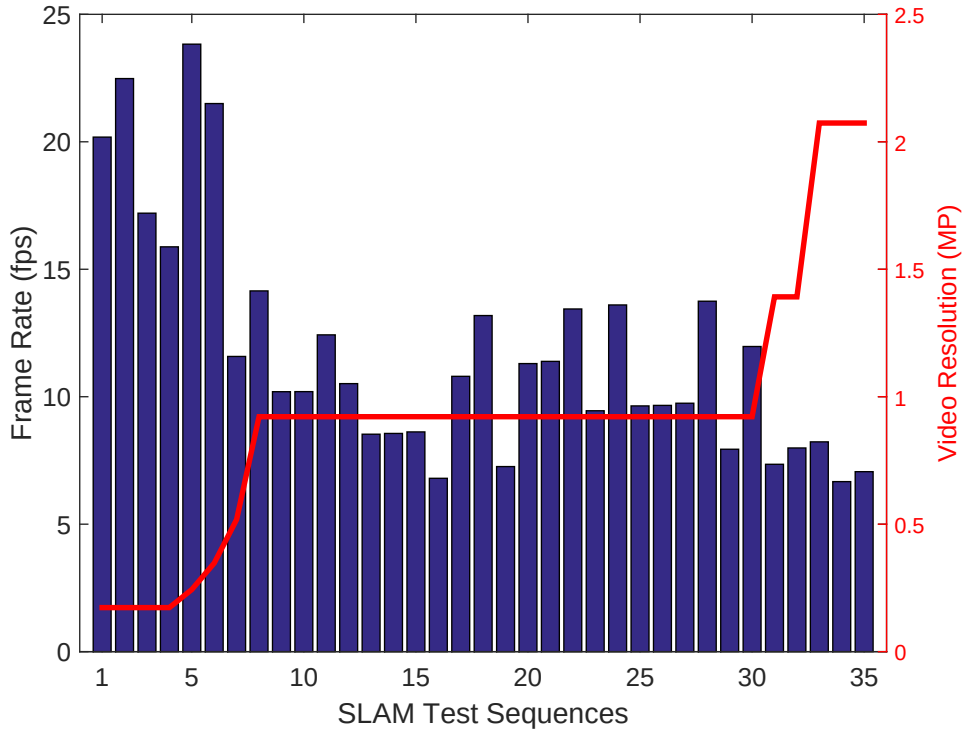


Figure 57. SLAM Sequence Run-time Results with Video Resolution.

is excellent for fast prototyping. However, run time suffers when compared to a compiled language such as C++. Despite the efforts made to optimize run time in MATLAB using pre-compiled binaries, logical indexing, and other efficient programming practices, a C++ implementation will still likely increase speed.

However, for proxops, typically the rate of relative motion is small. Specifically in the Orbital Express proxops sequences, the video analyzed is already at an advanced frame rate achieved by skipping many frames. For this reason, decreasing the frame rate in exchange for increased accuracy through incremental bundle adjustment or an iterative method of verifying point tracks may turn out to be an ideal implementation. Chapter V includes additional discussion on this topic.

4.5 Parameter Sensitivity Analysis

The success of SLAM hinges on the success of initialization and the relative pixel motion threshold used to automatically perform initialization. Subsequent 3D triangulations depend on relative motion levels which are determined initially by the amount of motion that occurred during initialization. Due to the importance of the relative pixel motion threshold parameter used for automating initialization, selecting an optimal default value is critical. Figure 58 shows the sensitivity of the results based on adjusting the relative pixel motion threshold. The relative pixel motion threshold value is varied from .0625% to 16% of the diagonal image dimension and appears on the X-axis of Figure 58. The bars represent the resulting performance of SLAM based on adjusting these parameters across all of the training videos. The height of the thick blue bar represents the success rate of initialization. The height of the thin red bar represents the success rate of subsequent structure and motion estimates after initialization. Success is determined based on the accuracy of pose estimates, visible point cloud accuracy, and model completeness.

Figure 58 shows that initialization is robust to variations in the relative pixel motion threshold. A large range in values yields a high success rate for initialization, likely due to the fact that bundle adjustment is performed on the first three frames of structure and motion estimates, compensating for potential error in the two image initialization. The success of subsequent SLAM varies more as numerous additional parameters are set based on the current rate of motion. For example, subsequent point triangulation occurs if a 2D track point exists for the same amount of camera translation that occurred in pose estimates 1-3. If pose estimates 1-3 occur very close together, subsequent triangulations will occur at a much higher rate and 2D tracks with a very short life will be triangulated quickly. The result is a very densely populated point cloud, but a higher potential for error.

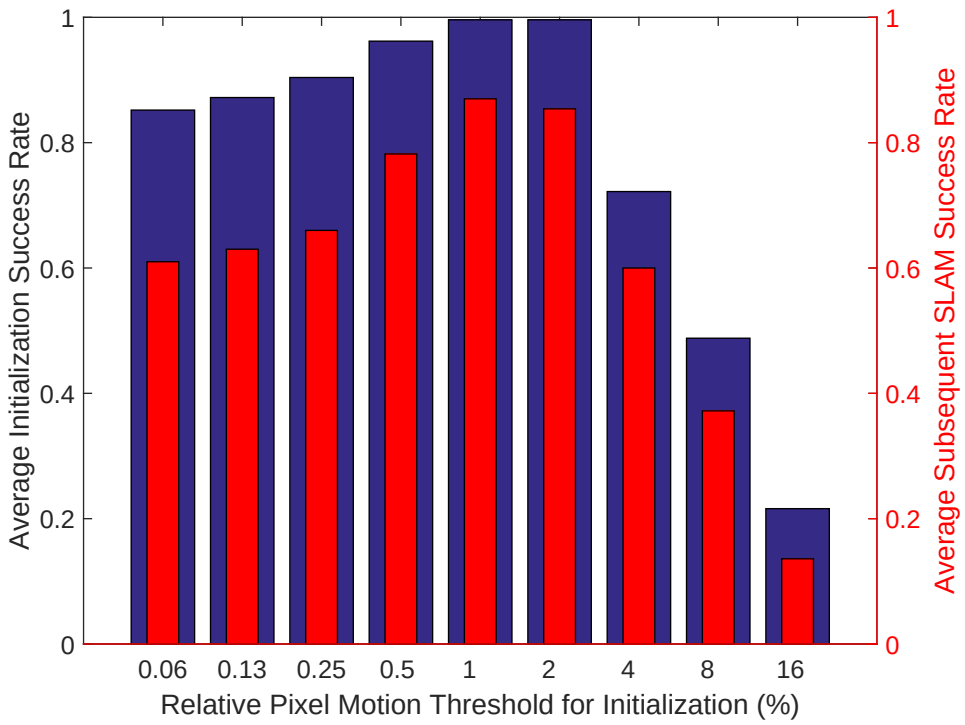


Figure 58. Sensitivity Analysis of the Relative Pixel Motion Threshold for Initialization.

On the other hand, a higher relative pixel motion threshold for initialization results in a sparse point cloud with estimates occurring at a longer interval. Additionally, only the point tracks with a relatively long life still exist and contribute to the triangulation step. In some cases, not enough point tracks are maintained to perform subsequent SLAM estimates, resulting in complete failure. The results of Figure 58 are used to select an optimal balance. A default value of 1% is selected for the relative pixel motion threshold for initialization. While this default value is not the best in every circumstance, it serves as a widely applicable starting point. The user may need to adjust this parameter, with the biggest contributing factor being the target's depth from the camera.

This sensitivity analysis is performed with the assumption that all motion begins with, at a minimum, relative translation of the camera, through which the camera obtains different perspectives of the target. Applying this method requires that the

end user ensures initialization is performed with adequate relative motion. If no additional perspective information is obtained in an initialization attempt, a failure rate of 100% is guaranteed. This failure is easily avoided though. Additionally, to obtain transparency into the actual sensitivity of the relative pixel motion threshold, the fail safe of automatically performing initialization during diminishing numbers of initial point tracks (introduced in Section 3.3.1) is not applied.

4.6 Summary

This chapter presented the results of this space-customized SLAM method applied to space-related video sequences, laboratory-acquired video sequences of a CubeSat testbed, and a series synthetic video sequences with known camera poses. An accuracy analysis using truth rotation values from the AFIT CubeSat testbed resulted in an average rotation estimation error per pose estimate of 0.60° with a standard deviation of 0.62° and a RMS value of 1.08°. The second accuracy analysis using the synthetic scenes resulted in an rotation estimation error per pose estimate of 0.29° with a standard deviation of 0.36° and a RMS value of 0.45°. Additionally, a run-time analysis of all video sequences analyzed resulted in operation at an average value of 11.8 fps. Finally, a parameter sensitivity analysis on the relative pixel motion threshold value used for initialization resulted in selecting 1% for the default value.

V. Conclusions and Recommendations

5.1 Key Advantages

This thesis presents a simultaneous localization and mapping method tailored for relative motion with objects in the space environment at a near real-time frame rate. One significant accomplishment of this implementation is the ability to perform SLAM based on relative motion from a single object in the image field of view without the assistance of pose estimation from background motion. The method is designed to exclude dynamic backgrounds that follow a different motion model from the target object, such as the Earth or stars in the space environment. The feature based detection and motion estimation method is designed to run quickly while being robust to reflective surfaces that are typically difficult to track, which are also common in the space environment.

An acceptable level of accuracy is achieved for a near real-time implementation. This SLAM algorithm is tailored to the available resources on a small space-based platform and target objects in the space environment. Two accuracy tests from a CubeSat testbed rotation test and synthetic video sequences with known poses result in average vertical axis rotation estimation values of 0.60° and 0.29°. One note about accuracy in SLAM is that perfection is not necessarily a requirement. The requirement is a level of accuracy that complies with the operational control system in which the algorithm is implemented. A run-time analysis resulted in algorithm operation at an average frame rate of 11.8 fps.

A significant advantage of this method over currently employed state-of-the-art automated proxops technology is the independence of the target object. Certain pose estimation methods involve the placement of fiducial markings on targets with a known 3D location to estimate pose. Other CV methods match the current view

of a known target with a model or a database of views to estimate pose. Operation of this method does not require any knowledge of the target object for a relative pose estimate. Estimating the structure also allows for automated avoidance if close proximity navigation is required. Cooperation from the target is not a requirement.

Another advantage of this work is its low SWAP requirements on the spacecraft. One of the goals of this thesis is to investigate the feasibility of employing prox-ops technology on a CubeSat-sized platform. The nature of this monocular method constrains operations to images from one camera, as a stereo camera rig offers no additional information on such a small platform. Additionally, low power is obtained through the use of a passive imaging system as opposed to active LiDAR sensors. Leveraging pre-existing hardware that exists on star trackers minimizes the use of additional physical space on the spacecraft.

5.2 Key Limitations

Despite the list of advantages of the method developed, limitations exist. The method is designed to be robust to lighting changes; however, certain constraints of passive imaging systems cannot be ignored. One large issue is saturation if the sun is in the direct path of the camera or a specular reflection off of targets. Figure 59 shows this issue in another portion of the Orbital Express robotic arm self-inspection video. A method to mitigate this problem is to modify the maneuver strategy. Maintaining a pointing direction that is perpendicular to the sun vector when orbiting another space object yields a more consistent level of illumination and avoids placing the sun directly in the field of view. Applications in geosynchronous orbit also minimize abrupt lighting changes and Earth eclipse which in turn minimizes the possibility of excessively low light.

There are additional nuances that require understanding by the operator in or-

der to customize and test this method. Structure and motion initialization requires relative camera translation, not just rotation. Very reflective targets with very few additional features result in a difficult situation to maintain enough accurate point tracks for successful SLAM.

A cooperative target is not required but a rapidly spinning out-of-control target will likely cause difficulty in detecting corner features with the FAST algorithm due to motion blur. Any rapid motion with a significant amount of image blurring will likely cause failure of the algorithm. Changes in structure, such as a change in orientation of solar panels will cause issues as well, as this method assumes one fixed object with all parts following one motion model. Additional work is required to recognize and adapt to structural changes and separate motion of independent parts of the object.

Anomalies do exist as well. In general, the sensitivity of specific adjustable parameters has been tested with a series of training videos to converge on globally optimal values. However, some tweaking may be required to obtain application-specific results in challenging scenarios. Future work may implement additional methods to remove the requirement for adjusting certain parameters for successful results in anomalous cases.

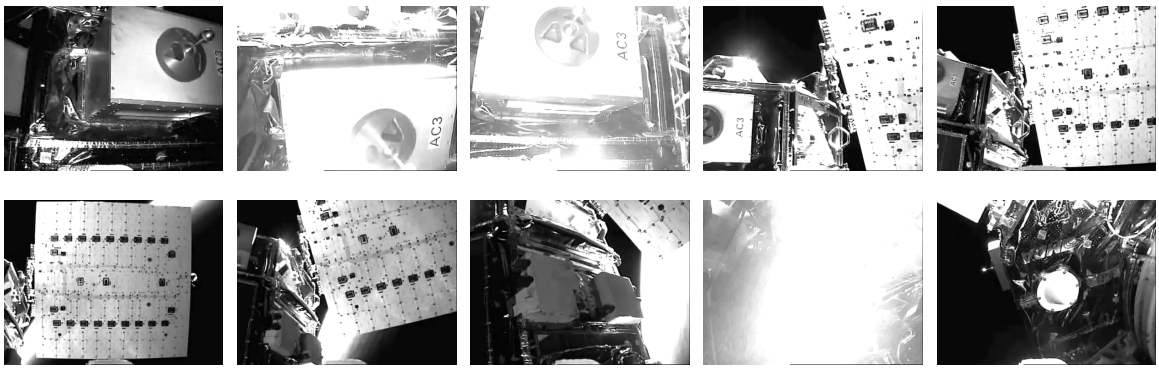


Figure 59. Orbital Express Self Inspection Video [33]: Saturation from Direct Sunlight in the FOV.

5.3 Future Work

The lowest hanging fruit for improvement is additional research to accurately extend point tracks to their entire time in view (from initial detection until they rotate out of view). Scale changes (motion toward or away from the target) and perspective change (relative target rotation) are challenges for point trackers, but this type of motion is exactly what is required to perform monocular structure and motion estimation. The KLT tracker with an affine consistency check on the feature window is a concept implemented during the testing phase of this research to accurately extend point tracks; however, the specific implementation slowed algorithm run-time to less than 1 Hz.

Additional research into typical relative motion rates during proxops is important as well. One of the goals of this research was to create a method that operates near real time. However, this rapid rate may not be necessary. An idea for future iterations may be more robust point tracking methods and additional error reduction through bundle adjustment schemes in exchange for a decrease in frame rate. This method includes the functionality to test various implementation techniques.

Another creative area for future research is incorporating motion estimates based on sensor inputs, such as an inertial measurement unit with SLAM. In [21], Kim, Hwangbo, and Kanade use sensor inputs to assist in point tracking of a fixed scene. Implementing a Kalman filter is potentially useful as well to provide an initial guess and outlier recognition for subsequent pose estimates. However, replacing the Kalman filter for SLAM with a particle filter has been shown to increase resiliency to rapid camera motions [38].

5.4 Summary

The SLAM method presented accomplishes the goals of this research. This work is the first step toward controlling an indoor proxops simulator from one camera, independent of the target. Innovate methods are presented to handle the challenging demands of the space environment. Additionally, numerous lessons learned by those at the forefront of the computer vision field have been leveraged. The thoroughly commented MATLAB routines included provide the foundation for testing and customization of the SLAM method in an easily understandable format, enabling productivity for future work. This research paves the way for proxops using a platform as small as a CubeSat.

Appendix A. Dependencies

The software written in this thesis is dependent on the following pieces of software:

- MATLAB r2014b
- MATLAB Computer Vision Systems Toolbox
- OpenCV 2.4.10, the open source computer vision library [9]
- Mexopencv, a development kit of MATLAB MEX functions for OpenCV [48]
- SBA, a software package for generic sparse bundle adjustment [26]
- Additional acknowledgement: the MATLAB 3D plot camera model is used from the epipolar geometry toolbox [29]

Bibliography

1. Andrew C. M. Allen, Christopher Langley, Raja Mukherji, Allen B. Taylor, Manickam Umasuthan, and Timothy D. Barfoot. Rendezvous lidar sensor system for terminal rendezvous, capture, and berthing to the international space station. In *Proc. of SPIE*, volume 6958, pages 69580S–69580S–8, 2008.

2. Angels fact sheet. The Air Force Research Laboratory. 377 ABW/Public Affairs, 2013.

3. Adnan Ansar and Konstantinos Daniilidis. Linear pose estimation from points or lines. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *ECCV (4)*, volume 2353 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2002.

4. Jonathan Balzer and Stefano Soatto. Clam: Coupled localization and mapping with efficient outlier handling. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

5. Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

6. Blue canyon technologies. High Performance CubeSat Systems and Components. World Wide Web Page, accessed 6-December-2014. Available at `http://bluecanyontech.com/all_products/cubesats/`.

7. Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.

8. Jean-Yves Bouguet. Camera calibration toolbox for matlab. World Wide Web Page, accessed 8-December-2014. Available at `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`.

9. G. Bradski. The open source computer vision library, opencv. *Dr. Dobb's Journal of Software Tools*, 2000.

10. Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly, 2008.

11. T.C. Bryan, R. Howard, J.E. Johnson, J.E. Lee, L. Murphy, and S.H. Spencer. Next generation advanced video guidance sensor. In *Aerospace Conference, 2008 IEEE*, pages 1–8, March 2008.

12. Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag.

13. Daniel D. Doyle. *Real-Time, Multiple Pan/Tilt/Zoom Computer Vision Tracking and 3D Positioning System for Unmanned Aerial System Research*. PhD thesis, Air Force Institute of Technology, 2013.

14. Olivier D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In G. Sandini, editor, *Computer Vision ECCV'92*, volume 588 of *Lecture Notes in Computer Science*, pages 563–578. Springer Berlin Heidelberg, 1992.

15. B.C. Hacker and J.M. Grimwood. *On the Shoulders of Titans: A History of Project Gemini*. NASA History. U.S. Government Printing Office, 2010.

16. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

17. R.I. Hartley. In defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(6):580–593, Jun 1997.

18. Richard Hartley and Peter Sturm. Triangulation, 1996.

19. Joel A. Hesch and Stergios I. Roumeliotis. A direct least-squares (dls) method for pnp. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc J. Van Gool, editors, *ICCV*, pages 383–390. IEEE, 2011.

20. Homer: Texas a&m's proximity operations simulator robot. World Wide Web Page, accessed 12-December-2014. Available at `http://lasr.tamu.edu/research/proxops/`.

21. Jun-Sik Kim, Myung Hwangbo, and T. Kanade. Realtime affine-photometric klt feature tracker on gpu in cuda framework. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 886–893, Sept 2009.

22. Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–10, Washington, DC, USA, 2007. IEEE Computer Society.

23. Georg Klein and David Murray. Improving the agility of keyframe-based SLAM. In *Proc. 10th European Conference on Computer Vision (ECCV'08)*, pages 802–815, Marseille, October 2008.

24. V. Lepetit, F.Moreno-Noguer, and P.Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal Computer Vision*, 81(2), 2009.

25. Longuet. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.

26. M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

27. David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

28. Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

29. Gian Luca Mariottini and Domenico Prattichizzo. The Epipolar Geometry Toolbox. *IEEE Robotics and Automation Magazine*, 1070-9930:2–16, 2005.

30. MathWorks. Single camera calibration using the camera calibrator app. World Wide Web Page, accessed 8-December-2014. Available at `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`.

31. NASA. International space station imagery. World Wide Web Page, accessed 6-December-2014. Available at `http://spaceflight.nasa.gov/gallery/images/station/crew-33/html/iss033e009286.html`.

32. David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):756–777, June 2004.

33. DARPA Tactical Technology Office. Orbital express data archive. World Wide Web Page, accessed 11-November-2014. Available at `http://archive.darpa.mil/orbitalexpress/index.html`.

34. Andrew Ogilvie, Justin Allport, Michael Hannah, and John Lymer. Autonomous robotic operations for on-orbit satellite servicing. *Proc. SPIE*, 6958:695809–695809–12, 2008.

35. Opencv 3.0.0 dev documentation: Camera calibration and 3d reconstruction. World Wide Web Page, accessed 10-December-2014. Available at `http://docs.opencv.org/trunk/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`.

36. A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer. Exhaustive linearization for robust camera pose and focal length estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10):2387–2400, 2013.

37. Prisma satellites: About prisma. World Wide Web Page, accessed 14-December-2014. Available at `http://www.lsespace.com/about-prisma.aspx`.

38. M. Pupilli and A. Calway. Real-time camera tracking using a particle filter. In *Britich Machine Vision Conference (BMVC'05)*, pages 519–528, Sept 2005.

39. Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

40. Brad Sease and Brien Flewelling. GEODETICA: A General Software Platform For Processing Continuous Space-based Imagery. *AAS (Preprint)*, 2014.

41. Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, Ithaca, NY, USA, 1993.

42. Space news. Cubist Movement, 2012.

43. Michael Tibbs. Conversation about laboratory setup and equipment, 2015.

44. Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

45. Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, London, UK, UK, 2000. Springer-Verlag.

46. Changchang Wu. Towards linear-time incremental structure from motion. In *Proceedings of the 2013 International Conference on 3D Vision*, 3DV '13, pages 127–134, Washington, DC, USA, 2013. IEEE Computer Society.

47. Changchang Wu. Visualsfm: A visual structure from motion system. World Wide Web Page, accessed 8-January-2015. Available at `http://ccwu.me/vsfm/`.

48. Kota Yamaguchi. mexopencv: Collection and a development kit of matlab mex functions for opencv library. World Wide Web Page, accessed 15-December-2014. Available at `https://github.com/kyamagu/mexopencv`.

49. Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666–673 vol.1, 1999.

50. Yinqiang Zheng, Yubin Kuang, S. Sugimoto, K. Astrom, and M. Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2344–2351, Dec 2013.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
6–03–2015	Master's Thesis	Oct 2013 — Mar 2015

4. TITLE AND SUBTITLE

A Monocular SLAM Method To Estimate Relative Pose During Satellite Proximity Operations

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Kelly, Scott J, Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

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

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENY-MS-15-M-219

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

Daniel D. Doyle, LtCol, USAF
Aerospace Vehicles Division
Air Force Research Laboratory
WPAFB OH 45433
Email: daniel.doyle@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RQV

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT Automated satellite proximity operations is an increasingly relevant area of mission operations for the US Air Force with potential to significantly enhance space situational awareness (SSA). Simultaneous localization and mapping (SLAM) is a computer vision method of constructing and updating a 3D map while keeping track of the location and orientation of the imaging agent inside the map. The main objective of this research effort is to design a monocular SLAM method customized for the space environment. The method developed in this research will be implemented in an indoor proximity operations simulation laboratory. A run-time analysis is performed, showing near real-time operation. The method is verified by comparing SLAM results to truth vertical rotation data from a CubeSat air bearing testbed. This work enables control and testing of simulated proximity operations hardware in a laboratory environment. Additionally, this research lays the foundation for autonomous satellite proximity operations with unknown targets and minimal additional size, weight, and power requirements, creating opportunities for numerous mission concepts not previously available.

15. SUBJECT TERMS

Computer vision, SLAM, proximity operations, pose estimation, rendezvous, controls, automation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Jonathan Black, AFIT/ENY
U	U	U	U	118	19b. TELEPHONE NUMBER *(include area code)* (540) 231-0037; jonathan.black@afit.edu