VISUAL NAVIGATION AND CONTROL FOR SPACECRAFT PROXIMITY OPERATIONS WITH UNKNOWN TARGETS

DISSERTATION

Wyatt J. Harris, Major, USAF

AFIT-ENY-21-DS-094

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

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

AFIT-ENY-21-DS-094

VISUAL NAVIGATION AND CONTROL FOR SPACECRAFT PROXIMITY OPERATIONS WITH UNKNOWN TARGETS

DISSERTATION

Presented to the Faculty

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 Doctor of Philosophy in Aerospace Engineering

Wyatt J. Harris, B.S.A.E., M.S.M.E.

Major, USAF

AFIT-ENY-21-DS-094

VISUAL NAVIGATION AND CONTROL FOR SPACECRAFT PROXIMITY OPERATIONS WITH UNKNOWN TARGETS

Wyatt J. Harris, B.S.A.E., M.S.M.E.
Major, USAF

COMMITTEE MEMBERSHIP:

Richard Cobb, PhD
Chairman

Lt Col Kirk Johnson, PhD
Member

Clark Taylor, PhD
Member

Ma Costantinos Zagaris, PhD
Member

Adedeji B. Badiru, PhD
Dean, Graduate School of Engineering and Management

Abstract

Many current and future spacecraft missions may conduct rendezvous and proximity operations (RPO) with resident space objects (RSOs). Historically, guidance, navigation, and control (GNC) methods for spacecraft performing RPO have relied heavily on available information about or from a target RSO in order to conduct maneuvers in close proximity. However, an important subset of spacecraft RPO that is yet to be demonstrated on-orbit involves final approach maneuvers with respect to RSOs where no information (such as geometry, inertia, relative velocity, etc.) is known about the target *a priori*, and no information is actively provided from the target during maneuvering. Such operation with respect to 'unknown' targets represents an important possible mission set for Department of Defense (DoD) spacecraft. This research presents two GNC frameworks capable of autonomously controlling complex RPO maneuvers with respect to moving unknown targets. Both methods are capable of being implemented in real time using low cost, size, weight, and power (SWAP) hardware and require minimal pre-maneuver inspection of the target.

The proposed GNC methods are intended to operate a fully actuated 'chaser' spacecraft with respect to an unknown 'target' in a regime where resolved imagery and depth maps of the target are available (the 'final approach' phase of RPO). For both methods, the only information required prior to initiating a final approach maneuver is a single camera image and dense stereo depth map captured from the initial relative pose between chaser and target, and a stereo camera sensor onboard the chaser is assumed to be the only sensor available for relative navigation during maneuvering. Computer vision pipelines for each GNC method are developed and demonstrated.

The first GNC method implements an image-based visual servo (IBVS) control law augmented with procedures to enable its use with unknown targets. The method

computes coupled six-degree-of-freedom (6DOF) maneuvers for the chaser with respect to an operator-defined region of interest (ROI) on the target RSO. In order to utilize the IBVS control law with unknown targets, two procedures are proposed to generate a required 'goal' image, which implicitly defines the desired relative pose between chaser and target. A noted benefit of the method pertains to the fact that control error is computed in the image domain rather than Cartesian space, meaning estimation of a full relative pose estimate is not required.

The second proposed GNC method represents an alternative visual servoing scheme designed specifically for use with unknown targets, termed Surface Normal Visual Servoing (SNVS). Surface Normal Visual Servoing (SNVS) computes a novel navigational vector (the 'Predominant Surface Normal (PSN)' vector) associated with an ROI on the target using stereo depth information. Two variants of SNVS are compared which each use the estimated target PSN vector in different manners to compute control commands. Several benefits of SNVS over the first GNC method are noted, including its effectiveness with target ROIs containing significant nonplanar features, and lack of dependence on computer vision feature point tracking methods.

Results from a realistic six degree of freedom simulation and using the Air Force Institute of Technology's robotic RPO simulator Control and Autonomy Space proximity Robot (CASpR) demonstrate complex maneuvers with respect to moving unknown targets. The methods presented in this work represent compelling alternatives to other proposed methods for RPO with unknown targets, as they do not require estimation of target inertia or rely on complex simultaneous localization and mapping (SLAM) algorithms to generate potentially brittle relative pose estimates.

Acknowledgments

I'd like to thank my advisor, Dr. Cobb, for his guidance throughout my time at AFIT. I truly appreciate all his advice pertaining to both my research and other topics. I'd also like to thank my other committee members for their assistance with my research, and Dr. Sinclair from AFRL for sponsoring my work.

There are no words to fully express my gratitude to my wife for everything she's done over the last few years to support me and our family. Thank you for raising our two amazing sons, and for all of your love and encouragement. I could not have done this without you.

Wyatt J. Harris

Table of Contents

Page

Page

Page

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
6DOF	six degree of freedom
AFIT	Air Force Institute of Technology
AFRL	Air Force Research Laboratory
APF	Artificial Potential Field
AVGS	Advanced Video Guidance Sensor
CAD	Computer Aided Design
CASpR	Control and Autonomy Space proximity Robot
COM	center of mass
COTS	commercial-off-the-shelf
CSRT	Channel and Spatial Reliability Tracker
DART	Demonstration for Autonomous Rendezvous Technology
DoD	Department of Defense
DCM	Direction Cosine Matrix
DLR	German Aerospace Center
DOF	degrees of freedom
ECI	Earth-Centered Inertial
EO	electro-optical
ETS-VII	Engineering Test Satellite
EKF	Extended Kalman Filter
FEM	Feature Extraction and Matching
FoV	Field of View
FLANN	Fast Library for Approximate Nearest Neighbors
FOV	field of view
FPS	frames per second
GNC	guidance, navigation, and control
GUI	graphical user interface
HITL	hardware-in-the-loop
HCW	Hill-Clohessy-Wiltshire
IBVS	Image-Based Visual Servoing
IEKF	Iterated Extended Kalman Filter
IR	infrared
JAXA	Japan Aerospace Exploration Agency
LIDAR	Light Detection and Ranging
LLC	Low Level Controller
LPF	low pass filter

Acronym	Definition
LKF	Linear Kalman Filter
LVLH	Local-Vertical-Local-Horizontal
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
KLT	Kanade-Lucas-Tomasi
NASDA	National Space Development Agency of Japan
NMC	natural motion circumnavigation
NERM	Nonlinear Equations of Relative Motion
OE	Orbital Express
OOS	on-orbit servicing
ORB	Oriented FAST and Rotated BRIEF
PBVS	Position-Based Visual Servoing
PI	proportional-integral
PID	proportional-integral-derivative
POI	Point of Interest
PRISMA	Prototype Research Instruments and Space Mission Technology Advancement
PSN	Predominant Surface Normal
RANSAC	RANdom Sample And Consensus
ROI	region of interest
RPO	rendezvous and proximity operations
RSO	resident space object
SE(3)	special Euclidean group of dimension 3
SFM	Structure From Motion
SIFT	Scale-Invariant Feature Transform
SLAM	simultaneous localization and mapping
SO(3)	special orthogonal group of dimension 3
SNVS	Surface Normal Visual Servoing
SSA	Space Situational Awareness
STM	State Transition Matrix
SURF	Speeded-Up Robust Features
SWAP	size, weight, and power
VNSS	Visual Navigation and Servoing Simulation

VISUAL NAVIGATION AND CONTROL FOR SPACECRAFT PROXIMITY OPERATIONS WITH UNKNOWN TARGETS

I. Introduction

1.1 Motivation

In 2009, satellites Iridium 33 and Cosmos-2251 collided in low Earth orbit, creating thousands of pieces of debris, much of which will remain in orbit well into the next century [1]. This collision was a stark reminder of how congested Earth orbit is, and how catastrophic the consequences will be if we do nothing about this fact. The threat of resident space object (RSO) collisions like this is one of many issues motivating development of spacecraft rendezvous technologies. If an uncontrollable RSO is deemed a collision hazard, having the capability to expeditiously rendezvous with it could avert a crisis [2]. Even more alluring than collision prevention though, RSOs could be prevented from ever becoming uncontrollable via on-orbit servicing (OOS) methods that could repair, refuel, or retrofit them to extend their life [3].

Many types of OOS missions will require autonomous operation, due to the fact that manned OOS missions are often impractical, and communication delays make teleoperating maneuvering spacecraft in close proximity infeasible. As a result, the subject of autonomous OOS has garnered significant attention in recent years. However, much work is yet to be done to demonstrate reliable methods for many types of potential OOS scenarios.

1.1.1 Spacecraft Rendezvous and Proximity Operations (RPO).

The term rendezvous and proximity operations (RPO) refers to operation of multiple spacecraft and/or RSOs in vicinity of one another such that a desired relative position and

orientation is achieved/maintained. There are many possible applications of RPO, which may or may not include docking/berthing between spacecraft. OOS is an application of RPO that pertains to spacecraft servicing and inspection operations, and is the primary focus of this research.

In the context of this work, RPO missions for OOS consist of two spacecraft: a chaser and a target. The chaser is assumed to be a maneuverable spacecraft being controlled to affect the goal of the OOS mission. The target is an RSO that is the subject of the maneuver. Missions of this type are typically partitioned into several operations or phases based on the relative range between spacecraft [4–7]. Delineation and definition of these operations varies throughout literature, but within the context of this work, the operations of a typical OOS mission include: phasing, homing, closing, final approach, and (possibly) docking/berthing. The phasing operation consists of a transfer maneuver by the chaser to reduce the phase angle from its current location to the target (Figure 1.1 graphically defines the phase angle). This operation only requires coarse knowledge of the orbit and location of the target. The next operation, homing (also commonly called far-range rendezvous), consists of a maneuver by the chaser from the phasing orbit to a point in vicinity of the target. This operation requires knowledge of the relative position between the chaser and target, though measurement accuracy can still be fairly coarse.

The next two operations, closing and final approach, consist of maneuvers to achieve the desired relative position and attitude between the chaser and target. Since this work examines camera sensors for relative navigation, these phases are characterized by what information is available about the target from the camera sensor (similar definitions can be found in [8]). The closing phase is defined herein to include any relative range when the target appears unresolved in the chaser's camera image. That is, the target is still far enough away that no discernible features can be identified. As relatively little information can be

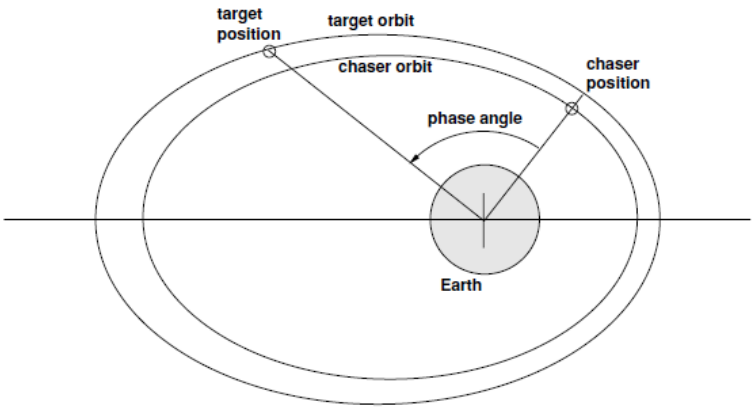

Figure 1.1: Definition of phase angle [7]

gleaned from camera sensors at this point, visual navigation with respect to the target is normally conducted with respect to the calculated centroid of the target in this phase.

The final approach phase is defined as the regime whereupon camera sensors onboard the chaser spacecraft are able to capture resolved imagery of the target—during this phase, camera sensors on the chaser can identify distinct features on the target and use knowledge of these features for relative navigation. The relative distances associated with each of these operations depends on characteristics of the camera sensor (such as focal length, resolution, focal plane array, etc.) and of the target itself. A rich set of information can be extracted from resolved imagery of the target, which can then be used to aid relative navigation with respect to the target in all six degree of freedom (6DOF).

Finally, if the mission requires physical contact between the chaser and target, an operation can be conducted to reduce the relative range to a distance sufficient for docking/berthing.

1.1.2 Types of RPO.

RPO missions for OOS are broadly divided into two categories, depending on the nature of the target RSO being serviced [9]. If the target can share information or otherwise operate to facilitate RPO with the servicing spacecraft (chaser), then the mission is termed

'cooperative.' If the target cannot actively communicate or maneuver to support RPO, then it is considered 'non-cooperative'. Additionally, the non-cooperative category is partitioned depending on what information is known about the target before the mission begins. If characteristics of the target are available, such as 3D geometry, moments of inertia, location of fiducial markers, etc., then the target is considered 'known.' Otherwise, if no information is available regarding the characteristics of the target, it is considered 'unknown.'

The engineering considerations and requirements associated with each of these categories varies greatly. For a given RPO scenario, conducting RPO with a non-cooperative target is generally considered more technically challenging than with a cooperative target, as spacecraft cannot coordinate with each other to affect a successful maneuver (such as sharing location or attitude information with each other). Moreover, conducting RPO with an unknown non-cooperative target presents an even greater challenge due to lack of knowledge about the target, which complicates relative navigation as many established methods for sensing and relative pose determination cannot be utilized.

The vast majority of RPO missions to date have been conducted with cooperative target spacecraft. For these operations, the fact that information is available about and from each spacecraft means that accurate relative navigation can be achieved via a variety of methods. As such, many processes and technologies have been proposed and demonstrated to facilitate close-in proximity operations between cooperative spacecraft. RPO with known non-cooperative targets are far less common, but have been investigated during several on orbit experiments (discussed in Section 2.2).

The only space missions pertaining to unknown targets have regarded exploration of near-Earth asteroids. While several of these missions demonstrated successful rendezvous with asteroids, the guidance, navigation, and control (GNC) methods implemented were largely tailored to this mission type, and are not directly transferable to the OOS-style

missions considered herein (these missions and their implications are discussed in Section 2.2.6).

To the knowledge of the author, no on orbit missions have demonstrated final approach phase RPO for OOS with unknown non-cooperative target RSO's. This type of mission is certainly plausible—for example if a 'dead' target spacecraft is physically damaged such that no geometric model is available, it would be both non-cooperative and unknown. There are many open questions about this subset of RPO that have not been answered in available literature, as discussed in the next subsection and Chapter II. Accordingly, the problem that this work primarily seeks to address is how final approach phase maneuvers with unknown non-cooperative targets can be autonomously conducted.

1.1.3 Relative Navigation Sensors for RPO with Unknown Targets.

This section provides a brief discussion of the types of relative navigation sensors available for final approach phase RPO with unknown targets, as this subject greatly influences the overall research effort.

For unknown non-cooperative targets, any sensor or navigation scheme which requires information from the target (such as GPS location, inertial attitude, fiducial markers, etc.) cannot be used. Consequently, research to date has almost exclusively considered electro-optical (EO) sensors for this type of mission [9]. Broadly speaking, EO sensors include devices capable of receiving radiation emitted in infrared, visible, or ultraviolet wavelengths.

EO sensors can be either active, meaning some wavelength of light is emitted from the sensor and the reflected returns of that light are collected as measurements, or passive, whereupon the sensor does not emit anything. Light Detection and Ranging (LIDAR), the most commonly proposed active sensor, works by casting infrared light on a scene via a laser source and analyzing the reflected returns. As opposed to a camera, LIDAR produces

a depth map (point cloud) of a scene, where each pixel conveys a measure of depth from the LIDAR sensor.

The most commonly cited passive sensor for relative navigation applications in space is a visible-spectrum camera. Passive infrared (IR) cameras have not been typically considered for relative navigation due to their comparatively poor image resolution (primarily as a result of IR's long wavelength), which makes identifying specific features difficult, and ultraviolet wavelengths are also typically not considered for any sort of relative navigation among spacecraft [9].

Compared to LIDAR, visible-spectrum cameras have relatively low size, weight, and power (SWAP) requirements, and cost less as well. Furthermore, visible-spectrum cameras have been demonstrated in the space environment and are already present on many current generation satellites. A main advantage of visible-spectrum cameras is their ability to capture visual information about a scene, which can be analyzed to identify and track features of interest over subsequent image frames. However, as discussed in Chapter II, relative navigation conducted using cameras is reliant on sufficient illumination conditions, which is not always guaranteed in space. Furthermore, monocular cameras do not readily provide depth in the same way as LIDAR, though this limitation can be relatively easily overcome by using a stereo camera, as discussed in Section 2.4.4.

Visible-spectrum cameras (specifically stereo cameras) are chosen for this work over LIDAR or other relative navigation sensors due to their ability to provide a rich set of information about a scene, and their demonstrated durability in the space environment. These images enable a range of computer vision methods to be used to aid relative navigation and enable one of the main methods examined in this work: visual servo control. A stereo camera is chosen for its ability to provide estimates of depth to objects in an image scene. Additionally, the lower SWAP and cost of cameras as compared to LIDAR make

them more attractive for a range of potential space missions, including small-sat missions, where space and power are constrained.

While visible-spectrum stereo cameras can provide a wealth of information, utilizing the information provided by these sensors for relative navigation and control with unknown targets is not a trivial task. There are available image processing methods from the field of computer vision that are applicable for target identification and tracking of unknown targets, however these methods have not been robustly examined for RPO applications (see Section 2.4 for a discussion of this topic). Additionally, established methods for 6DOF control using visual feedback (commonly refereed to as visual servo control) assume knowledge of the target (see Section 2.8 for an overview of several of these methods). Therefore, significant work remains to be done to apply these methods for unknown targets.

1.2 Research Questions and Tasks

This research effort seeks to advance the state of the art regarding autonomous navigation and control for the final approach phase of RPO with unknown non-cooperative targets.

Hypothesis: It is possible to autonomously perform complex 6DOF final approach maneuvers with respect to unknown non-cooperative target RSO's using a low SWAP stereo camera for relative navigation and robust control algorithms that do not require estimation of a geometric model of the target or its moments of inertia.

1.2.1 Research Questions.

The research questions that address this hypothesis are:

1. Is it possible to develop a visual navigation and control architecture for conducting the final approach phase of RPO with unknown non-cooperative targets that does not require a full relative pose estimate or estimating inertia/geometric properties of the target, and how robust is this architecture to measurement errors associated

with tracking feature points on the target? The method developed for this question is termed ‘GNC Method 1’.

2. Is it possible to develop a visual navigation and control architecture for conducting the final approach phase of RPO with unknown non-cooperative targets that does not require tracking target feature points or estimating inertia/geometric properties of the target? The method developed for this question is termed ‘GNC Method 2’.

3. Can methods developed for the previous two questions be implemented in real-time using feedback from a stereo camera sensor hardware, and are they effective when used in conjunction with available computer vision algorithms and hardware?

1.2.2 Research Tasks.

The over-arching goal of this work is to develop practical and robust navigation and control methods for spacecraft conducting final approach phase RPO with unknown targets. The methods presented herein are designed to use a stereo camera as the primary relative navigation sensor, which is capable of providing visual and depth information about the target in its field of view.

The proposed methods will be implemented in simulation and well as physical experimentation in a hardware-in-the-loop (HITL) enviroment. The simulation will facilitate initial analysis and improvement of the methods, while HITL experimentation will demonstrate the methods using actual camera sensor feedback in a realistic setting. The specific tasks that will be accomplished in order to answer each research question are outlined below.

Question 1 tasks:

(1.1) Develop a 6DOF simulation for relative spacecraft motion of a chaser and target, including implementation of navigation/control architecture.

(1.2) Propose a method(s) for generating the goal image necessary for visual servo maneuvers that specifies the desired end state of maneuver.

(1.3) Assess in simulation: (a) the architecture's ability to maneuver with respect to tumbling/moving targets and (b) the architecture's robustness to potential computer vision sources of error.

Question 2 tasks:

(2.1) Develop a method to estimate the predominant surface normal vector of a target region of interest, for the purpose of determining the desired chaser attitude for final approach maneuvers.

(2.2) Develop a method of visual servoing that utilizes this predominant surface normal vector in order to address the issue of maneuvering with respect to nonplanar regions of interest.

(2.3) Implement this new method in simulation and compare the effectiveness to architecture developed in answering Question 1.

Question 3 tasks:

(3.1) Develop practical versions of the navigation and control architectures developed for Questions 1 and 2, capable of being implemented in real-time using a stereo camera sensor and robust computer vision pipelines.

(3.2) Develop a failsafe mode that automatically initiates 6DOF relative pose-hold if maneuver termination criteria are met.

(3.3) Using hardware-in-the-loop, assess the ability of these architectures to maneuver a chaser in 4DOF (three in-plane dimensions and one rotational dimension) with respect to a moving non-cooperative unknown target.

1.3 Research Scope

For this work, a two spacecraft OOS-style mission is considered which includes a maneuverable chaser spacecraft and an unknown non-cooperative target RSO. The applicability of the research regarding relevant types and phases of RPO is shown in Figure 1.2. The proposed navigation and control methods primarily apply to maneuvers for the final approach phase of RPO, where the goal of maneuvering is to achieve a desired relative pose (position and attitude) between chaser and target. The proposed architecture can certainly be used with known targets, and could have utility in the docking phase of RPO as well.

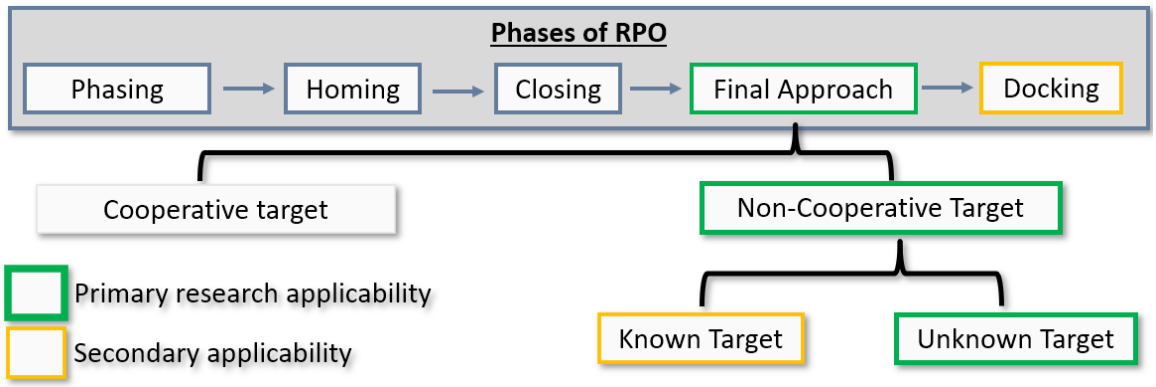

Figure 1.2: Scope of research

Figure 1.3 displays the proposed taxonomy of final approach maneuvers with unknown targets, including the primary methods this work presents. The figure refers to two GNC methods: Method 1 pertains to Research Question 1, and Method 2 pertains to Research Question 2, and describes an overall framework for how both methods can be combined in order to provide maximum utility for RPO with unknown targets.

1.4 Assumptions and Limitations

Assumptions and limitations relevant to this work are as follows:

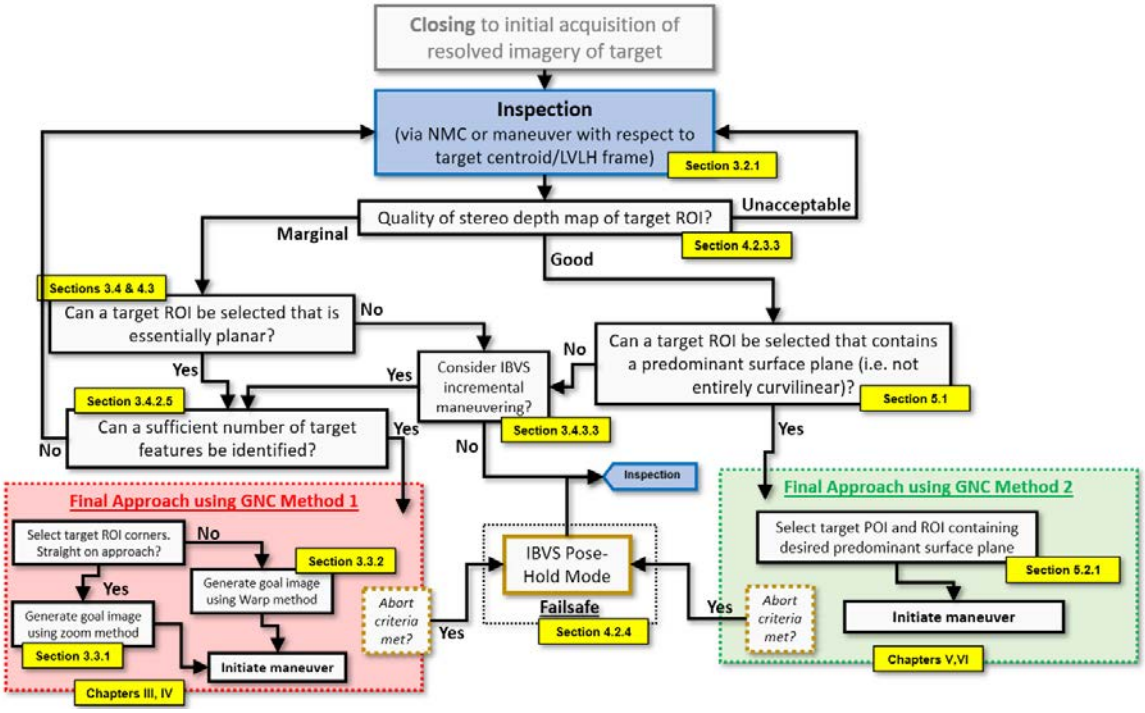

Figure 1.3: Proposed taxonomy of a final approach maneuver with an unknown target using visual navigation and control methods presented in this work.

1. Stereo camera sensor and real-time imagery: as discussed in Section 1.1.3, a visible-spectrum stereo camera is chosen to be the primary relative navigation sensor on the chaser spacecraft. Accordingly, it is assumed that an aligned dense depth map can be computed for each image captured by the maneuvering camera. Additionally, the camera sensor is capable of outputting images and depth maps at a rate sufficient for real-time maneuvering.

2. Initial relative pose: it is assumed that the chaser has completed a closing phase maneuver via other means and is initially positioned in close proximity to the target. There may be initial relative motion (translational or rotational) between the chaser and target (the direction and magnitude of which is initially unknown to the chaser). It is assumed the chaser has sufficient actuator authority to overcome this relative motion.

3. Resolved imagery: per the definition of final approach phase provided in Section 1.1.1, resolved imagery of the target RSO is assumed to be available. The imagery must be sufficiently resolved over the course of maneuvering such that features on the target surface can be tracked and a dense stereo depth map of the target region of interest (ROI) computed. It is assumed that there exists sufficient illumination on the target over the course of the maneuver to maintain resolved imagery.

4. Unknown target: Other than a single camera image and stereo depth map of the target captured from the initial relative pose, no other information about the target (physical geometry, mass, moments of inertia, etc.) is available *a priori*. The target ROI is assumed to be rigid in the sense that the physical configuration of the ROI does not change over the course of a relative maneuver.

5. Chaser actuation: chaser can maneuver in all 6DOF with respect to the target using continuous thrust/torque profiles. Applied forces and torques are decoupled.

6. Spacecraft dynamics: for the purposes of the simulated scenarios presented herein, both spacecraft are assumed to be rigid bodies, and orbital motion is Keplerian (no perturbations applied).

1.5 Expected Contributions

This work seeks to demonstrate that complex final approach RPO maneuvers with respect to an unknown non-cooperative target can be conducted using visual feedback from a stereo camera sensor. The expected contributions include:

1. A high-fidelity RPO simulation that models the 6DOF relative astrodynamics of a chaser and target spacecraft, as well as a stereo camera sensor onboard the chaser spacecraft. Significant capabilities of this simulation include:

   (a) Rendering of stereo camera gray-scale images and depth maps, useful for maneuver visualization and planning.

(b) Ability to synthesize stochastic image feature points on an imported target spacecraft model.

(c) A methodology for modeling the four main sources of error inherent with feature-based image processing and stereo depth estimation.

2. Development of visual navigation and control architecture based on a dynamic look-and-move Image-Based Visual Servoing (IBVS) control scheme capable of commanding complex maneuvers with respect to unknown targets (GNC Method 1). Demonstration of the architecture's ability to achieve desired pose with an unknown target in simulation. Assessment of the architecture's robustness to navigational error in simulation.

3. Development of two methods to generate goal images for previously unknown targets. These methods enable GNC method 1 to be used with unknown targets.

4. Development of a visual servo control law that uses a novel navigation vector, termed the Predominant Surface Normal (PSN) vector, to command complex maneuvers with moving unknown targets without requiring tracking of target feature points during maneuvering (GNC Method 2).

5. Development of computationally efficient method to estimate the PSN vector for a target ROI, as well as the predominant surface plane associated with this ROI.

6. Demonstration of practical implementations of both methods using feedback from a stereo camera sensor. Sub-contributions of this implementation will include:

   (a) Design, construction, and testing of a robotic RPO simulator, Control and Autonomy Space proximity Robot (CASpR), that can physically propagate movement of two independent bodies in 4DOF using linearized spacecraft equations of relative motion and feedback from a stereo camera sensor.

(b) Development of a feature-based image-processing pipeline capable of tracking natural point features identified on the target for use with GNC Method 1.

(c) Demonstration of a failsafe mode, capable of initiating a 6DOF relative pose-hold with a moving target object if criteria are met to terminate the current course of maneuver.

1.6 Document Preview

The rest of this document is organized as follows. Chapter II presents an overview of background material from the fields of astrodynamics, robotics, and computer vision, as well as a review of relevant literature. Chapter III presents GNC Method 1 and demonstrates the method in simulation. Chapter IV discusses additions to GNC Method 1 necessary for HITL implementation, and provides experimental results. Chapter V develops GNC Method 2, and includes simulation results demonstrating the method. Chapter VI provides experimental results of GNC Method 2 using HITL. Finally Chapter VII concludes the work by summarizing the proposed methods, contributions, and providing recommendations for future research.

II. Background and Literature Review

2.1 Chapter Overview

This chapter provides background information and a review of literature relevant to this research. Since the subject of this work is spacecraft rendezvous and proximity operations (RPO) using vision-based sensors, first a number of historical space missions incorporating vision-based experiments are summarized. The translational and rotational dynamics that govern relative spacecraft motion are then presented. Next, applicable topics from the fields of computer vision and visual navigation are detailed. Finally, control methods for rendezvous final approach are introduced, with a focus on visual servo control. Recently published works pertaining to these topics are referenced and discussed throughout. The chapter summary details the research gaps this work intends to fill.

2.2 Historical Autonomous RPO Missions

Several space experiments in recent decades have examined the final approach phase of autonomous spacecraft rendezvous using vision-based navigation methods. Each of these missions aimed to demonstrate various guidance, navigation, and control (GNC) techniques for autonomous spacecraft RPO using cameras as a primary relative navigation sensor. This section details a selection of experiments relevant to this research, in chronological order.

2.2.1 Engineering Test Satellite (ETS-VII).

The ETS-VII was a two-satellite formation launched in 1997 and managed by the National Space Development Agency of Japan (NASDA) [10]. While most of the mission's experiments centered around teleoperation of the chaser's robotic arm, autonomous rendezvous was demonstrated three times using a suite of relative navigation sensors which included a monocular camera. The target was highly cooperative though— data was shared

among satellites to perform differential GPS, and the target possessed retro-reflectors that allowed the camera to determine relative pose within 2 meters range. Although ETS-VII's visual navigation capabilities were limited, it is still noteworthy as it was likey the first on-orbit experiment to implement autonomous vision-based relative navigation [11].

2.2.2 XSS-10 & XSS-11.

The Air Force Research Laboratory (AFRL) conducted the XSS-10 [12] and XSS-11 [11] missions in 2003 and 2005 respectively. XSS-10 was a 28-kg micro-satellite deployed from the second stage of a Delta II booster that proceeded autonomously on a preplanned inspection route around the booster, using a combination of an IMU, GPS, star sensor, and a monocular camera for relative navigation. While the mission was successful, its visual navigation scheme was relatively basic by current standards, as it functioned by simply estimating the centroid of the target, so it therefore could not maneuver with respect to the 6DOF pose of the target. XSS-11 had a similar operational concept to XSS-10. It successfully conducted 75 circumnavigations of the rocket body it was deployed from, this time with the addition of a LIDAR sensor for range and angles measurements.

2.2.3 Demonstration for Autonomous Rendezvous Technology (DART).

DART was a NASA project launched in 2005. The objective of DART was to conduct a series of maneuvers in close vicinity to the target (down to 5 meters range) using its Advanced Video Guidance Sensor (AVGS). The target was passive (non-maneuvering) and non-cooperative, though it contained fiducial markers in the form of retro-reflectors that assisted the chaser in visually identifying it and estimating relative pose. The mission ended prematurely after the DART vehicle collided with the target [11].

2.2.4 Orbital Express (OE).

OE, launched in 2007, was a two satellite formation managed by DARPA and NASA [13]. This mission represented the most advanced autonomous rendezvous endeavour yet undertaken, with a goal to conduct on-orbit servicing with the target spacecraft.

With a range of sensors including a narrow-field of view (FOV) camera, a wide-FOV camera, an infrared camera, a laser range-finder, and the same AVGS employed by DART, OE conducted a range of demonstration experiments, including autonomous capture and mating. As with DART, the target was 3-axis stabilized in attitude and contained retro-reflectors. According to [13], 100% of mission objectives were achieved, thus making it a successful demonstration of autonomous rendezvous with a non-cooperative and known target.

2.2.5 Prototype Research Instruments and Space Mission Technology Advancement (PRISMA).

PRISMA was a joint European venture headed by the Swedish National Space Board launched in 2010 [14, 15]. It was also a two-satellite formation, with a 6-DOF maneuverable chaser and passively stabilized target. The chaser was equipped with a suite of sensors including differential GPS, Formation-Flying Radio Frequency (FFRF), and a vision-based system (VBS) [16]. These sensors were used in various combinations over several experiments that each demonstrated a different relative GNC mode. Notably, some experiments were conducted using only the VBS, assuming the target to be non-cooperative (though the target did possess optical markers in the form of flashing LEDs at known locations) [17]. The chaser used visually identified markers and the known 3D model of the target to determine relative pose. This experiment demonstrated approach with a non-cooperative/known target down to 2 meters range with centimeter level accuracy.

2.2.6 Asteroid Rendezvous Missions.

It is worth noting that a number of missions have attempted rendezvous with asteroids, which could certainly be classified as RPO with an unknown target. Of these, the most notable are two missions conducted by the Japan Aerospace Exploration Agency (JAXA): Hyabusa (launched 2005) and Hyabusa 2 (launched 2014) [18–21]. These missions each approached, observed, and rendezvoused with a near-Earth asteroid. They used a

combination of sensors, including Light Detection and Ranging (LIDAR) and visible-spectrum cameras. For the final descent phase, both missions utilized a 'target marker,' which was dropped from the spacecraft beforehand to aid in conducting the maneuver. These markers contained a number of retro-reflectors that could be easily identified by the spacecraft, and were critical for the spacecraft to be able to zero out its relative horizontal velocity prior to touching down.

While these missions technically demonstrated rendezvous with an unknown resident space object (RSO), it is clear that the methods and requirements for touching down on an asteroid are generally different than those for an on-orbit servicing (OOS) mission. For example, the asteroids were large enough to have somewhat of a gravity well, which was used during these missions to affect the rendezvous (almost in a similar manner to planetary descent). Additionally, it can be assumed that the landing area of the asteroid was bigger and required less positioning accuracy than could be expected for an OOS mission. Also, the method of attaching a fiducial marker to the target ahead of time would be less plausible for an OOS mission as that operation would itself require a somewhat complex rendezvous maneuver in order to attach the marker to the target at a slow enough relative velocity to not damage or disrupt the target.

2.2.7 RPO Missions Summary.

While the fore-mentioned missions demonstrated the ability to autonomously dock with non-cooperative targets, they all relied on knowledge of the target, usually in the form of known fiducial markers and/or a geometric model of the target. Unfortunately the methods used for these experiments do not directly apply for the case of an unknown target, so the capability to conduct RPO with unknown non-cooperative targets is yet to be demonstrated.

2.3 Spacecraft Relative Motion Dynamics

In the final approach phase of RPO, both relative position and attitude between the chaser and target are of interest, so both translational and rotational dynamics must be treated. Herein it is assumed that both spacecraft are rigid bodies and that orbital motion is Keplarian (no perturbations applied). The chaser is capable of applying force and torque in any direction, and it is assumed that applied forces and torques are decoupled. Section 2.3.1 develops relative translational dynamics for the chaser and target, Section 2.3.2 describes the attitude dynamics/kinematics for each spacecraft, and Section 2.3.3 details how feature points on the target spacecraft are propagated.

2.3.1 Translational Dynamics.

Figure 2.1 depicts the relevant coordinate frames; $\{N\}$ is the Earth-Centered Inertial (ECI) frame, $\{L\}$ is the Local-Vertical-Local-Horizontal (LVLH) frame that rotates around $\{N\}$ along a reference orbital path, $\{T\}$ is the target frame, and $\{C\}$ is the chaser frame. For the $\{L\}$ frame, $\hat{X}_L$ (the 'radial' axis) points radially outward from the center of the Earth, $\hat{Z}_L$ (the 'cross-track' axis) is parallel to the reference orbit angular momentum vector, and $\hat{Y}_L$ (the 'in-track' axis) completes the triad. Without loss of generality, $\{T\}$ and $\{C\}$ are assumed to be at their respective spacecraft center of mass (COM) and aligned with principal moments of inertia. To avoid establishment of additional frames, the origin of the chaser and camera sensor coordinate frames are assumed to be coincident and aligned.

Typically, the origin of the $\{L\}$ frame would be chosen such that it is coincident with the origin of the $\{T\}$ frame (though not necessarily rotationally aligned). For the RPO scenarios considered herein with unknown targets, in practice this would be achieved *a priori* using an additional/separate Space Situational Awareness (SSA) capability to determine the orbit and location of the target. However, if the SSA solution is not completely accurate, or if the target maneuvers, then $\{L\}$ and $\{T\}$ frames will not be coincident. Therefore, the $\{L\}$ origin is assumed to be a virtual point located on a reference orbit in vicinity of the target frame

$\{T\}$. In this way both the chaser and target are free to maneuver (translate and rotate) with respect to the $\{L\}$ frame.

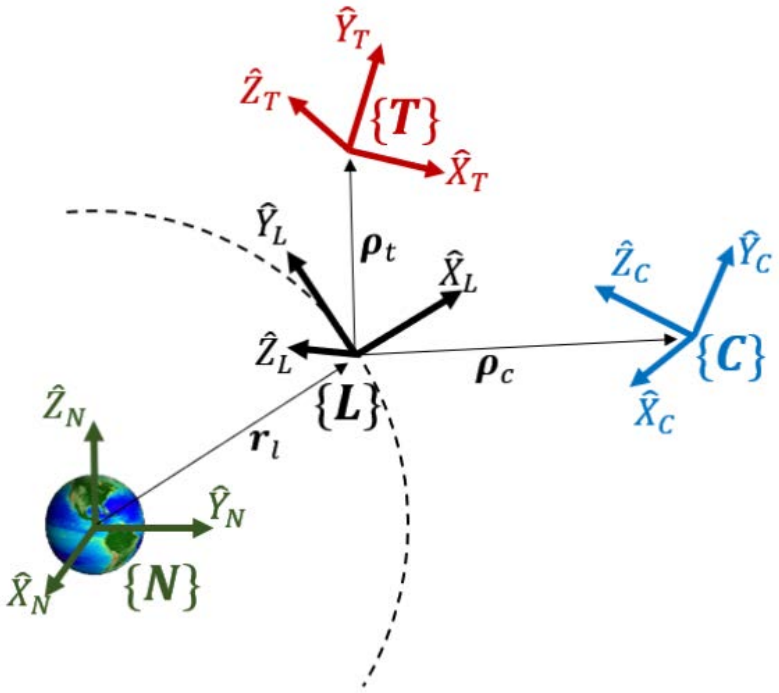

Figure 2.1: Relative motion coordinate frames: ECI $\{N\}$, LVLH $\{L\}$, chaser/camera $\{C\}$, and target $\{T\}$

The well known Nonlinear Equations of Relative Motion (NERM) [22] are used to model the translational states for the ith body ($\boldsymbol{\rho}_i = [x, y, z]^T$) with respect to the LVLH origin [22]

$$\ddot{x} - 2\dot{\theta}\dot{y} - \ddot{\theta}y - \dot{\theta}^2 x = \frac{-\mu}{r_i^3}(r_l + x) + \frac{\mu}{r_l^2} + u_x, \tag{2.1}$$

$$\ddot{y} + 2\dot{\theta}\dot{x} + \ddot{\theta}x - \dot{\theta}^2 y = \frac{-\mu}{r_i^3} y + u_y, \tag{2.2}$$

$$\ddot{z} = \frac{-\mu}{r_i^3} z + u_z, \tag{2.3}$$

where θ is the target orbit argument of latitude, $r_L = \|\mathbf{r_L}\|$, $r_i = \|\mathbf{r_i}\| = \left[(r_l + x)^2 + y^2 + z^2\right]^{1/2}$, and $\mathbf{u} = [u_x, u_y, u_z]^T$ is the control force acting on the spacecraft body. θ and r_L can be found using

$$\ddot{\theta} = \frac{-2\dot{r}_l\dot{\theta}}{r_l}, \tag{2.4}$$

$$\ddot{r}_l = r_l\dot{\theta}^2 - \frac{\mu}{r_l^2}. \tag{2.5}$$

The NERMs are used to model the individual translational motion of both the chaser and target with respect to the LVLH origin. Differential perturbations such as J2 effects or solar radiation pressure are not considered. This is reasonable in the context of this research as the relative effects of perturbations are generally insignificant for the ranges and time frames of the maneuvers considered.

2.3.1.1 Hill-Clohessy-Wiltshire Equations of Motion.

Under certain assumptions, the NERMs can be approximated by a set of second-order linear time-invariant differential equations known as the Hill-Clohessy-Wiltshire (HCW) equations of motion [23]. These linear equations are computationally efficient, and therefore ideal for onboard systems such as navigation filters or controllers. To derive the HCW equations, consider the following assumptions and their implications on the NERMs:

(i) Assume no control forces affect relative motion $\Rightarrow$ u_x, u_y, u_z terms in (2.1)-(2.3) are zero.

(ii) Assume no perturbations affect relative motion $\Rightarrow$ already assumed in the formulation of NERMs in (2.1)-(2.3).

(iii) Assume LVLH frame's orbit is circular $\Rightarrow$ $e = 0, \ddot{\theta} = 0, \dot{\theta} = constant = n = \sqrt{\frac{\mu}{a^3}}$

Applying these assumptions, (2.1)-(2.3) become:

$$\ddot{x} - 2n\dot{y} - n^2 x = -\frac{\mu(r_i + x)}{r_i^3} + \frac{\mu}{r_l^2}, \tag{2.6}$$

$$\ddot{y} + 2n\dot{x} - n^2 y = -\frac{\mu y}{r_i^3}, \tag{2.7}$$

$$\ddot{z} = -\frac{\mu z}{r_i^3}. \tag{2.8}$$

Next, assume that the distance from the LVLH frame to the spacecraft is small relative to the LVLH frame's orbital radius, and note that $r_l = a$ for a circular orbit. Then r_i is

$$\begin{aligned} r_i &= \sqrt{(a+x)^2 + y^2 + z^2} \\ &= \sqrt{x^2 + 2ax + a^2 + y^2 + z^2} \\ &= a\sqrt{\frac{x^2 + y^2 + z^2}{a^2}^{\,0} + \frac{2x}{a} + 1} \\ &= a\sqrt{1 + \frac{2x}{a}}. \end{aligned} \tag{2.9}$$

The term $1/r_d^3$ can be expanded using the binomial theorem and simplified using the closeness assumption:

$$\begin{aligned} a^{-3}(1 + \frac{2x}{a})^{-3/2} &= 1 - \frac{3}{2}\left(\frac{2x}{a}\right) - \frac{\frac{3}{2}\left(-\frac{3}{2} - 1\right)}{2!}\left(\frac{2x}{a}\right)^{2\,0} + H.O.T. \\ &= 1 - \frac{3x}{a}. \end{aligned} \tag{2.10}$$

Finally, (2.6)-(2.8) can be simplified to yield the HCW equations (note $\mu = n^2a^3$):

$$\begin{aligned}
\ddot{x} &= \frac{-n^2a^3}{a^3}\left(1 - \frac{3x}{a}\right)(a - x) + \frac{n^2a^3}{a^2} + 2n\dot{y} + n^2x \\
&= -n^2\left(a + x - 3x - \cancelto{0}{\frac{3x^2}{a}}\right) + n^2a + 2n\dot{y} + n^2x \\
&= 2n\dot{y} + 3n^2x \qquad (2.11)
\end{aligned}$$

$$\begin{aligned}
\ddot{y} &= \frac{-n^2a^3}{a^3}\left(1 - \cancelto{0}{\frac{3x}{a}}\right)y - 2n\dot{x} + n^2y \\
&= -2n\dot{x} \qquad (2.12)
\end{aligned}$$

$$\begin{aligned}
\ddot{z} &= \frac{-n^2a^3}{a^3}\left(1 - \cancelto{0}{\frac{3x}{a}}\right)z \\
&= -n^2z. \qquad (2.13)
\end{aligned}$$

Equations (2.12)-(2.13) represent the HCW equations of motion. Conveniently, these equations have a closed-form solution [24]:

$$x(t) = \left[4x_0 + \frac{2}{n}\dot{y}_0\right] + \frac{\dot{x}_0}{n}\sin(nt) - \left[3x_0 + \frac{2}{n}\dot{y}_0\right]\cos(nt), \qquad (2.14)$$

$$y(t) = -\left[6nx_0 + 3\dot{y}_0\right]t + \left[y_0 - \frac{2\dot{x}_0}{n}\right] + \left[6x_0 + \frac{4}{n}\dot{y}_0\right]\sin(nt) + \frac{2}{n}\dot{x}_0\cos(nt), \qquad (2.15)$$

$$z(t) = \frac{\dot{z}_0}{n}\sin(nt) + z_0\cos(nt), \qquad (2.16)$$

$$\dot{x}(t) = \dot{x}_0\cos(nt) + \left[3x_0n + 2\dot{y}_0\right]\sin(nt), \qquad (2.17)$$

$$\dot{y}(t) = -\left[6nx_0 + 3\dot{y}_0\right] + \left[6\dot{x}_0n + 4\dot{y}_0\right]\cos(nt) - 2\dot{x}_0\sin(nt), \qquad (2.18)$$

$$\dot{z}(t) = \dot{z}_0\cos(nt) - z_0n\sin(nt). \qquad (2.19)$$

Finally, these solutions can be used to form a State Transition Matrix (STM) for use with navigation filters and controllers:

$$\mathbf{\Phi}(t,0) = \begin{bmatrix} 4-3\cos(n\,t) & 0 & 0 & \frac{\sin(n\,t)}{n} & -\frac{2\,(\cos(n\,t)-1)}{n} & 0 \\ 6\,\sin(n\,t)-6\,n\,t & 1 & 0 & \frac{2\,(\cos(n\,t)-1)}{n} & \frac{4\sin(nt)}{n}-3t & 0 \\ 0 & 0 & \cos(nt) & 0 & 0 & \frac{\sin(nt)}{n} \\ 3n\sin(nt) & 0 & 0 & \cos(nt) & 2\sin(nt) & 0 \\ 6\,n\,(\cos(n\,t)-1) & 0 & 0 & -2\,\sin(n\,t) & 4\,\cos(n\,t)-3 & \\ 0 & 0 & -n\sin(nt) & 0 & 0 & \cos(nt) \end{bmatrix}. \tag{2.20}$$

2.3.2 Attitude Dynamics.

The attitude dynamics of the chaser and target can each be independently modeled using Euler's rotational equations of motion. The time derivative of the angular velocity of the ith body (where i is either the target or chaser) with respect to the inertial frame is [25]

$$\dot{\boldsymbol{\omega}}_{i/N} = \boldsymbol{J}_i^{-1}\left[-\boldsymbol{\omega}_{i/N}\times\boldsymbol{J}_i\boldsymbol{\omega}_{i/N}+\boldsymbol{\tau}_i\right], \tag{2.21}$$

where $\boldsymbol{J}_i$ is the moment of inertia for the ith body and $\boldsymbol{\tau}_i$ is the control torque acting on the ith body (other disturbances are not considered). Since attitude with respect to the LVLH frame is of interest it can be calculated by

$$\boldsymbol{\omega}_{i/L} = \boldsymbol{\omega}_{i/N} - \boldsymbol{\omega}_{L/N}, \tag{2.22}$$

where $\boldsymbol{\omega}_{L/N}$ is the time rate of change of true anomaly. For a circular orbit, $\boldsymbol{\omega}_{L/N}$ is constant.

Quarternions are chosen to parameterize the attitude of each spacecraft due to their lack of singularity. The quaternion kinematic equation used to propagate each spacecraft's attitude with respect to the LVLH frame can be described as in [26]

$$\dot{\boldsymbol{q}}_{i/L} = \frac{1}{2}\boldsymbol{Q}(\boldsymbol{q}_{i/L})\boldsymbol{\omega}_{i/L}, \tag{2.23}$$

where

$$\boldsymbol{Q}(\boldsymbol{q}) = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix}. \tag{2.24}$$

2.3.3 Calculation of Target Feature Point Vectors.

The visual servo control law detailed in Section 2.8 uses knowledge of a set of feature points located on the target body. An arbitrary feature point defined in the target body frame, $\boldsymbol{p}_j^T$, is not necessarily coincident with the target's COM. Therefore, when determining the position of this feature point, the equations above cannot be used alone, as a coupling between position and attitude occurs when a point is not at a body's COM. Figure 2.2 shows graphically how the *j*th feature point is related to the target body frame, the LVLH frame, and the chaser frame.

During camera image synthesis, in order to compute the pixel location of a feature point, $\boldsymbol{p}_j^T$ must be resolved in the camera frame $\{C\}$. To do this, a coordinate transformation is performed using a matrix of the special Euclidean group of dimension 3 (SE(3)) [27]. For example, the transformation matrix from the LVLH frame to the chaser frame is

$$\boldsymbol{T}_L^C = \begin{bmatrix} \boldsymbol{R}_L^C & -\boldsymbol{R}_L^C\boldsymbol{\rho}_c^L \\ \boldsymbol{0}_{1x3} & 1 \end{bmatrix}, \tag{2.25}$$

where $\boldsymbol{R}_L^C$ is a Direction Cosine Matrix (DCM) of special orthogonal group of dimension 3 (SO(3)) from the LVLH frame to the chaser frame, and $\boldsymbol{\rho}_c^L$ is the position vector of the chaser frame with respect to the LVLH frame, resolved in the LVLH frame.

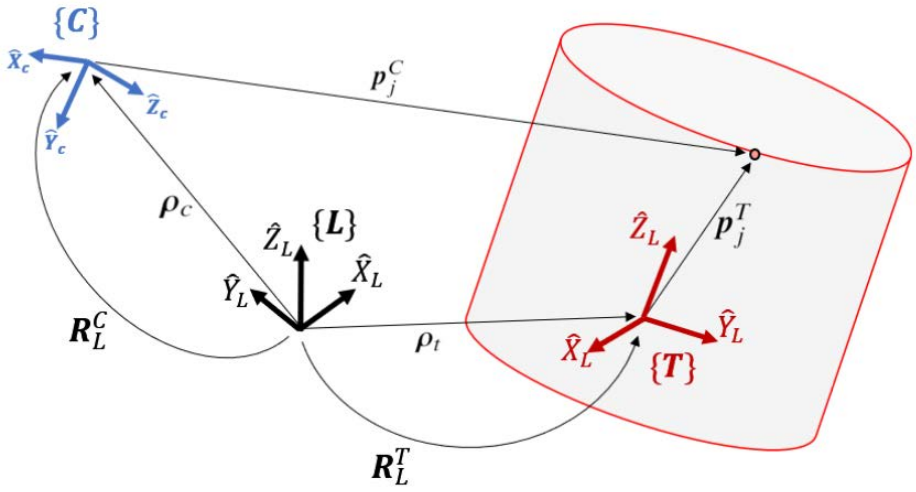

Figure 2.2: Target feature point geometry

Given knowledge of the position and attitude of the target frame and the chaser frame with respect to the LVLH frame, a set of feature points defined in the target body frame can be resolved in the chaser frame via successive multiplication by two transformation matrices

$$\begin{bmatrix} \boldsymbol{p}_j^C \\ 1 \end{bmatrix} = \boldsymbol{T}_L^C \boldsymbol{T}_T^L \begin{bmatrix} \boldsymbol{p}_j^T \\ 1 \end{bmatrix}. \tag{2.26}$$

2.4 Computer Vision

Camera sensors suitable for use with the visual navigation and control methods proposed in this work are assumed to be capable of providing video output at real-time frame rates (typically 30 frames per second or greater). Each image frame of the video sequence contains a wealth of information about the scene the camera is viewing. Several

methods from the field of computer vision can be used to exploit this information for navigation and control. This section provides a brief overview of relevant computer vision topics, starting with how a camera sensor is mathematically modeled.

2.4.1 Camera Modeling.

While there are many ways to model a camera sensor, it is common in the field of computer vision to use a pinhole camera model [28]. Much like the human eye, a pinhole camera creates a perspective projection of a 3D scene on a 2D image plane as depicted in Fig. 2.3. According to convention, the $\hat{Z}_c$ axis of the camera frame points along the optical axis, the $\hat{X}_c$ axis points to the right of the camera, and $\hat{Y}_c$ completes the triad. For a 3D point resolved in the camera frame, $\boldsymbol{p}^C = [X, Y, Z]^T$, the projection of that point onto the image plane in pixel coordinates is

$$\boldsymbol{p}^P = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & \gamma & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{2.27}$$

where f_u, f_v are the horizontal and vertical focal length (expressed in units of pixels):

$$\begin{aligned} f_x &= \frac{f\ (meters)}{pixel\ width\ (meters/pixel)}, \\ f_y &= \frac{f\ (meters)}{pixel\ height\ (meters/pixel)}, \end{aligned} \tag{2.28}$$

(c_u, c_v) is the principal point of the image frame, and $\boldsymbol{K}$ is the camera calibration/intrinsic matrix. γ is the skew coefficient representing the amount of shear distortion for each pixel of the sensor (typically 0 or nearly 0). The vector $[x, y, 1]^T$ represents the point expressed as a homogeneous vector using normalized image plane coordinates. In practice the camera intrinsic matrix is determined via a process called camera calibration.

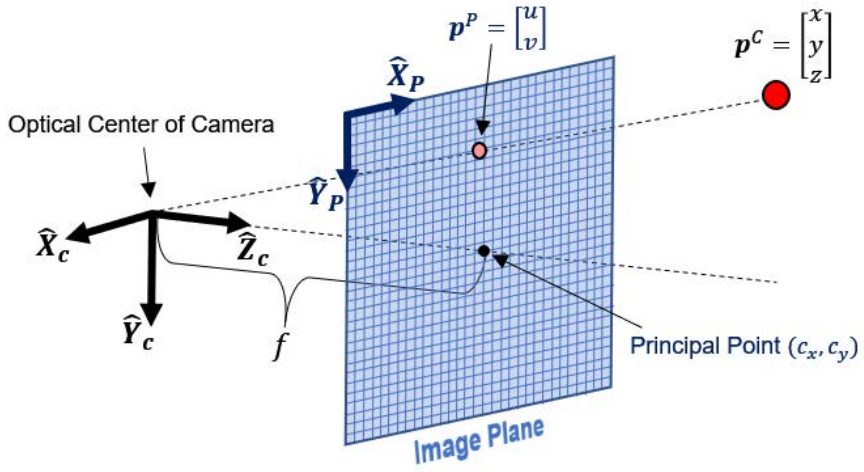

Figure 2.3: Pinhole camera model

Real camera images can exhibit other types of distortion, such as 'pincushion' distortion or 'barrel' (fisheye) distortion. These types of distortion are not considered in this work as they can generally be removed in software for calibrated cameras.

Note that there is an inherent scale ambiguity for a single image. Because each point coordinate is divided by z_c (the distance along the optical axis), information regarding depth is lost when the point is projected onto the image plane (i.e. there are an infinite number of 3D points that would project to the same pixel coordinates). One aspect of visual navigation requires dealing with this ambiguity, and will be discussed in Section 2.4.4.

Another important aspect of modeling a camera is understating where the camera is with respect to some reference frame (the 'world' frame denoted as $\{W\}$ in this section). The term 'pose' is used to describe the translation (location) and rotation (attitude) of a rigid body (in this case the camera, represented by the coordinate frame $\{C\}$). The pose of a camera in 3D space contains six degree of freedom (6DOF): three for translation, and three for rotation. The translation of the camera with respect to the world frame, resolved in the world frame, is represented by a vector, $\boldsymbol{t}^W_{W\to C}$. The rotation of the camera from the

world frame to the camera frame is represented by a DCM, $\boldsymbol{R}_W^C$. $\boldsymbol{R}_W^C$ and $\boldsymbol{t}_{W\to C}^W$ are depicted in Figure 2.4.

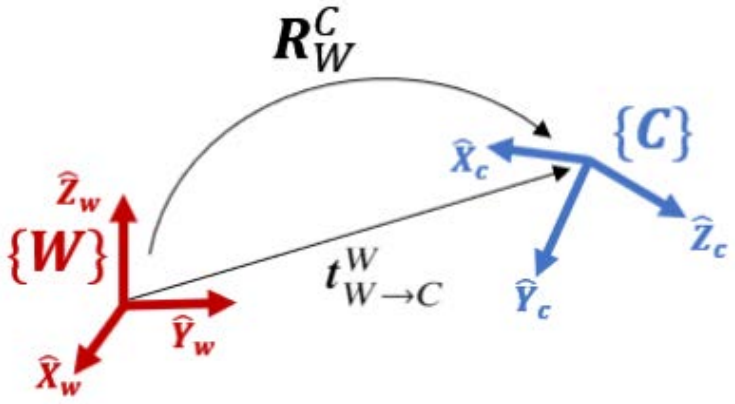

Figure 2.4: Camera extrinsic parameters

Together $\boldsymbol{R}_W^C$ and $\boldsymbol{t}_{W\to C}^W$ are known as a camera's extrinsic parameters. These parameters can be encapsulated in a matrix of SE(3), the camera extrinsic matrix

$$\boldsymbol{T}_W^C = \begin{bmatrix} \boldsymbol{R}_W^C & -\boldsymbol{R}_W^C \boldsymbol{t}_{W\to C}^W \\ \boldsymbol{0}_{1x3} & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_W^C & \boldsymbol{t}_{C\to W}^C \\ \boldsymbol{0}_{1x3} & 1 \end{bmatrix}. \tag{2.29}$$

Together the camera intrinsic and extrinsic matrices can be used to transform a 3D point expressed in the world frame, $\boldsymbol{p}^W$, into pixel coordinates in the image frame

$$\boldsymbol{p}^P = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{K}\, \boldsymbol{T}_W^C\ \boldsymbol{p}^W. \tag{2.30}$$

2.4.2 Epipolar Geometry.

If two pin-hole cameras observe the same scene from different angles (as is the case with a stereo camera or a single camera imaging an unchanging scene from two

perspectives), several observations can be made about the geometric relationships between the two images. Together these relationships are known as epipolar geometry. This Section provides a brief overview of epipolar geometry as it applies to stereo vision. [28] provides a thorough summary of concepts related to this topic and multiple view geometry in general.

Epipolar geometry describes the projective geometry between two images of a scene. Figure 2.5 depicts the relevant aspects of this geometry for two cameras C_L and C_R observing a point p. The perspective projection of p in C_L is p_L and p_R in C_R. The plane formed by the point p and the optical centers of C_L and C_R is known as the epipolar plane, and the points where the line connecting optical centers are projected onto each image plane are epipoles (e_L and e_R). The line drawn through a camera image projection of p and the same camera's epipole is known as an epipolar line.

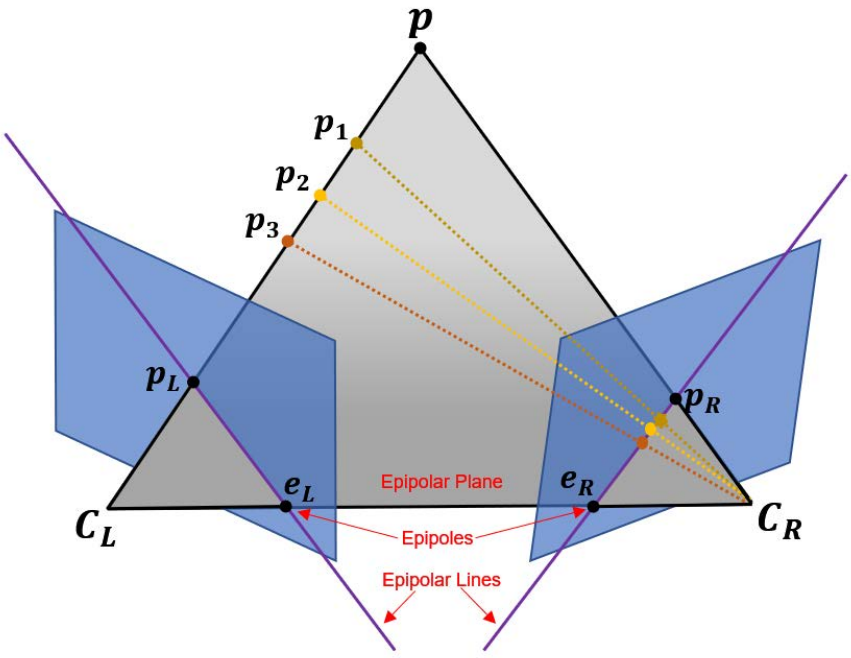

Figure 2.5: Epipolar geometry

It can be seen in Figure 2.5, that the distance to point $\boldsymbol{p}$ to C_L cannot be determined by its projection $\boldsymbol{p}_L$ alone, as the points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$ provide the same projection. Epipolar geometry can be exploited to calculate the depth to $\boldsymbol{p}$ given the additional view from the second camera though, along with a known relative pose (translation and rotation) between the two cameras. A basic method to do this involves finding the proper correspondence between projections of $\boldsymbol{p}$ in the left and right cameras. The problem of correspondence is simplified by the fact that the projection of $\boldsymbol{p}$ in the right image lines on the epipolar line on that image.

For a typical stereo camera, where the two image planes are parallel and vertically alligned, a calculation of depth to $\boldsymbol{p}$ can be made via triangulation by using the disparity between the horizontal pixel coordinates of the projections of $\boldsymbol{p}$ in each image, x_L and x_R

$$Z = \frac{fB}{x_l - x_r}, \tag{2.31}$$

where f is the focal length (assumed equal for both cameras) and B is the baseline between camera optical centers.

2.4.2.1 Essential Matrix.

Epipolar geometry imposes a constraint between projected (*normalized*) point.

$$\boldsymbol{p}_L^T(\boldsymbol{t} \times \boldsymbol{R}\boldsymbol{p}_R) = 0 \tag{2.32}$$

This is known as the epipolar constraint and can alternatively be written using the essential matrix $\boldsymbol{E} = [\boldsymbol{t}]^\times \boldsymbol{R}$ (where $[\boldsymbol{t}]^\times$ is a skew symmetric matrix),

$$\boldsymbol{p}_L^T \boldsymbol{E} \boldsymbol{p}_R = 0. \tag{2.33}$$

This relationship is very useful as it can be used to recover an estimate of the relative pose between two views up to a scale factor (given known image point

correspondences), though there are known problems of degeneracy under certain conditions [29]. Alternatively it can be used with outlier detection algorithms, as discussed in Section 2.4.3.3.

2.4.2.2 Homography Matrix.

The concept of a homography between two images is closely related to that of the essential matrix, though slightly less generalized. A homography relates the projections of a planer surface viewed in two different images. The projection of a point $\boldsymbol{p}_i$ on plane π in one image is related to its projection in another image ($\boldsymbol{p}_i^*$) by

$$\boldsymbol{p}_i^* = \boldsymbol{H}\boldsymbol{p}_i, \tag{2.34}$$

where $\boldsymbol{H}$ is a 3x3 homography matrix. Various methods exist to estimate the homography matrix for a planar surface viewed in two images as discussed in [30], and these methods can be made robust using an outlier rejection method such as RANdom Sample And Consensus (RANSAC) [31].

Relative pose (up to a scale factor) can be estimated using a calculated homography and the relationship:

$$\boldsymbol{H} = \boldsymbol{R} + \frac{\boldsymbol{t}}{d}\boldsymbol{n}^T \tag{2.35}$$

where d is the distance from the camera to the plane π and $\boldsymbol{n}$ is a unit vector normal to π for which the homography is defined. Several methods exist to decompose a homography matrix into a rotation matrix and translation/normal vectors. However, these methods provide a set of possible solutions, which require an additional method or information to determine which solution is correct [32].

2.4.3 Computer Vision For Relative Navigation.

Relative visual navigation for spacecraft RPO is possible via use of computer vision image processing techniques. Image processing refers to the "application of a set of techniques and algorithms to a digital image to analyze, enhance, or optimize image characteristics" [33]. In this context, image processing methods facilitate identification and tracking of the target by extracting information from camera sensor measurements that describes the general shape, size, and/or specific features of a target seen in the image.

The overall methodology for target tracking depends on the goal of the visual navigation scheme being utilized. This section summaries techniques for two general approaches to conduct relative navigation using imagery of a target: navigation based on a single point or region of interest (ROI) associated with the target (Sections 2.4.3.1 and 2.4.3.2), and full 6DOF navigation based on tracking of multiple features located on the target body (Section 2.4.3.3 and 2.4.3.4).

2.4.3.1 Target Tracking Using Image Segmentation.

For the first approach, navigation is performed using calculated centroid of the target, and possibly relative size or shape of the outline of the target in the camera image. This approach is applicable to angles-only navigation and is especially useful if the target only appears as a small group of pixels in an image frame (i.e. it is unresolved), or if a sufficient number of features on the target cannot be identified for full 6DOF maneuvering. Here the goal of image processing is to simply identify the target in the image (i.e. locate its centroid and possibly relative size/shape in the image).

A basic approach to locate the target in a camera image is to use a process known as image segmentation. Various techniques are available to accomplish this, but the ultimate goal is to segment the image into regions containing the target and other regions not containing the target. This is usually performed via a method of thresholding [34]. The

output of a thresholding process is a binary image, where each pixel value has a value of either 0 or 1, indicating the presence of the target or not.

A well-segmented image for RPO navigation should reveal one primary 'blob' that incorporates most or all of the target. Calculation of image moments [34] can then be used to determine characteristics of this blob useful for navigation and tracking, such as outline, size, and geometric centroid. Figure 2.6 shows a raw image taken on orbit during the PRSIMA experiment [14] on the left and a thresholded image on the right. Also shown are the highlighted outline of the target and the geometric centroid.

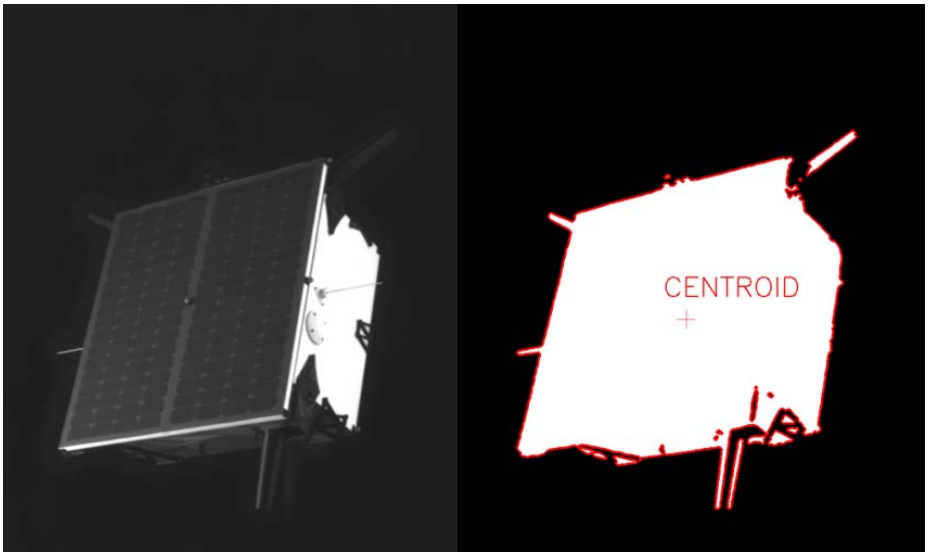

Figure 2.6: Example of image segmentation for target identification. Original image on left, thresholded image on right (original image from PRISMA data set [35])

2.4.3.2 Target Tracking Using Single Object Trackers.

While object tracking via image segmentation has utility for some applications such as the closing phase of RPO, it is a comparatively rudimentary method that will often break down as the target and tracking environment become more complex, such as in the case during the final approach phase when a full 6DOF relative pose is desired. To address the tracking problem for more challenging conditions, many research efforts have sought

to develop dedicated tracking algorithms, often called single object trackers, capable of robustly tracking objects given a single training image of the target object. As [36] states though, it is very difficult to develop a single algorithm capable of tracking objects under all conditions, due to the enormity of possible objects and environments. Accordingly, many proprietary tracking algorithms designed for particular applications exist, as well as a number of more general-use open-source algorithms. Several of the more common open-source algorithms are implemented in OpenCV [37], and are considered for this research effort. A number of surveys have attempted to quantify the effectiveness of these available algorithms, such as the recent work by Brdjanin and Dzigal [38]. As these authors suggest however, lack of standardization of evaluation metrics and the immense range of possible operating conditions make it difficult to comprehensively compare tracking algorithms.

An example of one of the more recent and effective open-source tracking algorithms is Channel and Spatial Reliability Tracker (CSRT) [39]. Like several previous algorithms that were widely considered effective, CSRT implements a tracking method based on the discriminative correlation filter (DCF); see [40] for an introduction to the use of correlation filters for object tracking. CSRT augments the DCF approach with spatial and channel reliability methods to overcome shortcomings of previous DCF-based tracking algorithms. While it is not the most computationally efficient open-source tracking algorithm, CSRT has been demonstrated to be effective at tracking objects given a single training image under a wide range of conditions.

2.4.3.3 Image Feature Tracking via Feature Extraction and Matching (FEM).

In order to maneuver the chaser both to a desired position and attitude relative to the target, additional information beyond the centroid and size of the target in each image frame is required. Here a different procedure is useful for image processing. The goal of this procedure is to identify and track unique features located on a target such that they can be used for visual navigation and control. In practice, features can take several different forms

such as points, lines, or other geometric shapes. This research considers point features as they are most readily compatible with the visual servo control laws discussed in Section 2.8.

Figure 2.7 displays the main procedures of a typical point feature-based image-processing pipeline applicable to this research. These steps include: reading an image frame from the camera sensor, detecting point features ('keypoints') in the image, determining suitable descriptors for detected features, and finally matching identified features to a previous 'goal' image in order to track them across video frames. The final step is made robust using additional outlier rejection methods.

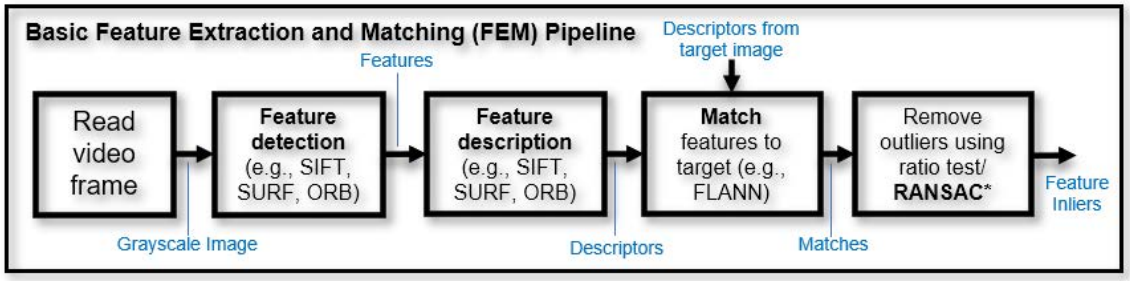

Figure 2.7: Basic FEM processing pipeline

Feature detection involves locating the pixel coordinates of a unique point feature in an image (for example, a corner). Many feature detector algorithms exist, with some of the more popular algorithms being Scale-Invariant Feature Transform (SIFT) [41], Speeded-Up Robust Features (SURF) [42], and Oriented FAST and Rotated BRIEF (ORB) [43].

Each feature detection algorithm locates features differently. SIFT, for example, detects features by first performing scale-space extrema detection where potential points that are invariant to scale and orientation are identified. Keypoints are then selected based on their measure of stability calculated using a 3D quadratic function. An example of SIFT features detected in an on-orbit image from the PRISMA dataset is shown in Figure 2.8.

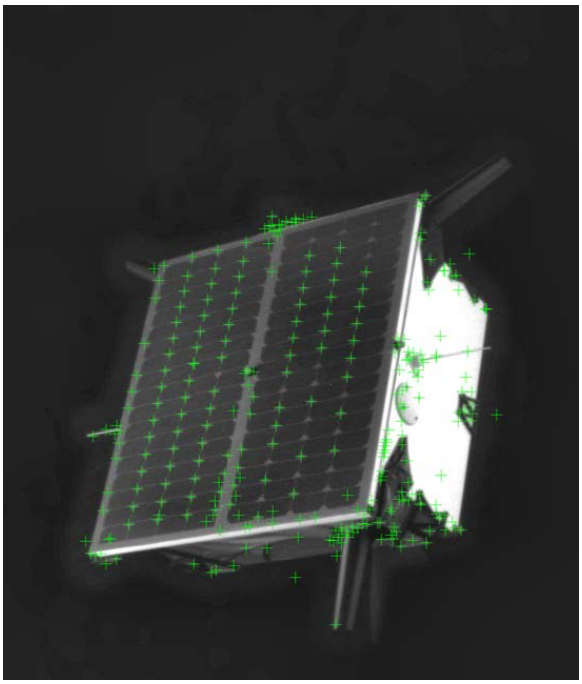

Figure 2.8: Example of SIFT features detected for on-orbit image (features indicated by green markers, original image from PRISMA data set [35])

The next step of a feature-based image processing pipeline is feature description. The goal of this procedure is to generate a descriptor for each detected feature, such that this descriptor can be compared against descriptors from other images to determine if the same feature has been identified in both image frames. As with detectors, there are many descriptor algorithms (SIFT, SURF, and ORB all implement descriptor algorithms in addition to detectors). Descriptor algorithms generate a vector for each feature that characterizes the feature based on other aspects of the image. SIFT does this by first assigning an orientation(s) to each keypoint based on local image gradient directions. Next, local image gradients are measured at various scales in sub-regions around the keypoint. These are rotated according to the keypoint orientation, and combined to form orientation histograms representing each sub-region. Figure 2.9 show an example transformation of image gradients from an 8×8 set of image gradients into a 2×2 descriptor array. In

practice, SIFT uses a 4×4 descriptor array, with each element containing 8 values, for a total of $4 \times 4 \times 8 = 128$ values in a SIFT descriptor vector.

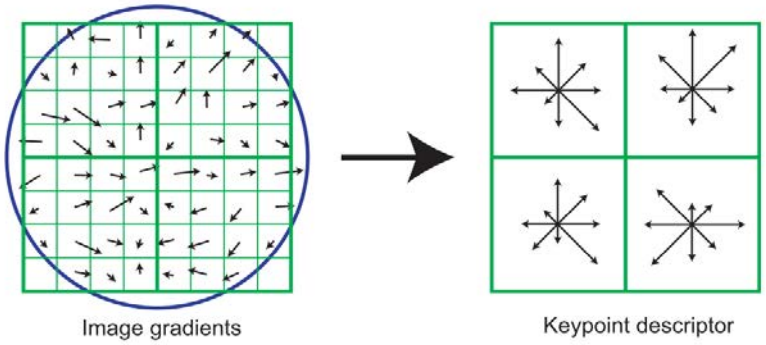

Figure 2.9: Transformation of image gradients to keypoint descriptors [41]

For the pipeline shown in Figure 2.7, feature tracking is achieved via a procedure called matching. Matching seeks to find unique correspondences between features across images, thus allow features to be repeatedly identified for tracking purposes. Once features are detected and described in two image frames, feature matches across the images can be found using a brute-force technique or via a dedicated matching algorithm such as Fast Library for Approximate Nearest Neighbors (FLANN) [44]. Feature matches are determined by comparing feature descriptors. Brute-force matching accomplishes this by comparing a descriptor in one image to all descriptors in the other image and calculating a distance value for each potential match (typically a Euclidean distance or Hamming distance depending on the type of descriptor). The feature with the lowest distance value is taken as a match for the feature in the first image. While this method works, it is not computationally efficient as it searches through every potential match for each feature.

FLANN is another method of feature matching that was designed to be a faster alternative to brute-force matching. It relies on a collection of algorithms utilizing kd-trees [45] to compute the approximate nearest neighbor solution for feature matches.

While the solution of matches provided by FLANN is approximate, it is significantly more computationally efficient than brute-forcing matching, making it more attractive for real-time applications such as those in this work.

The matching process is typically far from perfect —a significant portion of matches are usually incorrect. Lowe's ratio test, described in [41], is a method that can be used to remove a large portion of these outliers by comparing the best match for a given feature to the second best match. If the ratio of the distance values associated with these matches is greater than a pre-defined threshold (a value of 0.8 is cited in [41]), then the match is discarded. According to [41], this ratio test "eliminates 90% of the false matches while discarding less than 5% of the correct matches."

For many applications, eliminating only 90% of outliers is not enough to ensure success. Therefore, in order to robustly filter out the vast majority of incorrect matches, outlier rejection is usually implemented via a method such as RANSAC [31]. RANSAC is an iterative method designed to fit a model to experimental data sets that contain a high percentage of outliers. For the application of feature matching, RANSAC can be used to determine which feature matches are inliers and which are outliers.

To implement RANSAC for rejection of outlier matches, an additional method is required which is used as the model that RANSAC fits iterative solutions to. One option here is find the essential matrix relating two images using a method such as Nister's five-point method [46]. During each iteration of RANSAC, the essential matrix is re-computed using five randomly selected matches. All matches are then compared against this solution of the essential matrix to see how well they fit this model of the epipolar geometry relating the two images. If a match produces a distance to a corresponding epipolar line that is less than some pre-defined threshold, then it is considered an inlier. The number of inliers for a given iteration is recorded, and iteration continues until a max number of iterations is

reached, or the ratio of inliers to outliers exceeds a predefined threshold. The final set of inliers is taken from the iteration that contained the most inliers.

Figure 2.10 shows an example of the entire feature-based image processing pipeline applied to to images from the PRISMA data set. Green lines connect matched features between the two images. Close inspection shows that features appear properly matched between images.

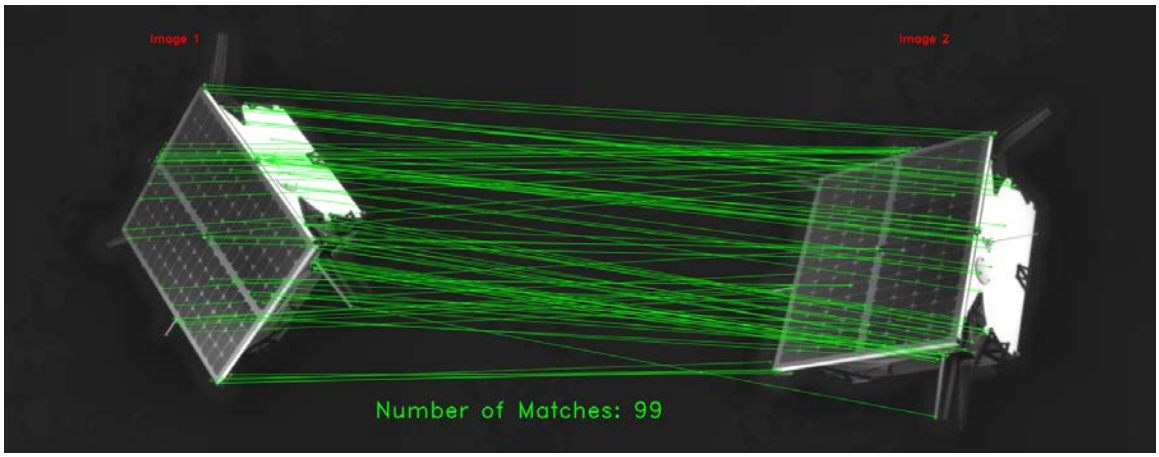

Figure 2.10: Example showing matched features across two images (original images from PRISMA data set [35])

2.4.3.4 Image Feature Tracking via Kanade-Lucas-Tomasi (KLT) Optical Flow.

Tracking features throughout a video sequence solely with FEM has several downsides: features may not be detected or matched in every frame, sometimes features are incorrectly matched (even with an outlier rejection method implemented), and the entire process is quite computationally expensive. An alternative that overcomes some of these downsides is to use a tracking method that only searches a local region around each feature in order to identify its new position in subsequent image frames. Here the sparse optical

flow method introduced by Kanade, Lucas, and Tomasi is a well-known feature tracking method.

In their 1981 work, Lucas and Kanade proposed a method for stereo image registration that takes advantage of the spatial intensity gradient between images to find point matches [47]. The generalized technique to register a point, $\boldsymbol{x}$, between two images $\boldsymbol{F}$ and $\boldsymbol{G}$, related by a translation, $\boldsymbol{h}$, and linear transformation (e.g. scale, rotation, shear, etc.), $\boldsymbol{A}$,

$$\boldsymbol{G}(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{h}), \tag{2.36}$$

involves minimizing the error associated with the relationship,

$$\boldsymbol{E} = \sum_{\boldsymbol{x}} (\boldsymbol{F}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{h}) - \boldsymbol{G}(\boldsymbol{x}))^2. \tag{2.37}$$

In practice this minimization problem is solved using a window of pixels around the point being tracked. A first order linearization assumption is made in order to compute an iterative least-squares solution for the transformation of points from the first to second frame. In a follow-on work, Tomasi and Kanade improved the method by proposing a method to choose feature windows [48].

Compared to FEM, tracking feature points via the KLT algorithm is far less computationally costly and generally produces a smooth trajectory for feature points throughout a sequence of images (due to the fact that tracked features are necessarily identified in every frame). However, if a KLT feature track is lost, the method has no way to reacquire that feature, making the number of features monotonically decrease as they are tracked though a sequence of images.

2.4.3.5 Target Tracking Using Machine Learning.

Finally, it is worth noting that another method used for identifying objects in a scene harnesses machine learning techniques [49]. These methods generally work by training

a classifier *a priori* that can locate a certain type of object in a scene based on common characteristics learned during training. For these methods to succeed, they usually require large training data sets (on the order of hundreds or thousands of images). Because this research assumes a unknown target however, there will be no data set available for training *a priori*. Even if a large data set of other spacecraft imagery is available, the large disparity among spacecraft designs suggests that a trained classifier might not work on an arbitrary unknown target. For these reasons, machine learning techniques are not considered for this research, but may warrant future consideration.

2.4.4 Review of Methods for Estimating Depth.

Achieving an accurate estimate of the pose of a camera with respect to some three-dimensional reference frame is a fundamental goal of a visual navigation system. To do this, information regarding range (depth) to objects in the camera's field of view is required. However, it can be seen from (2.27) that the depth (value of z) cannot be resolved from a single image (assuming no information regarding objects in the scene is available): proportionally changing the components of $\boldsymbol{p}^C$ will yield the same value for $\boldsymbol{p}^P$, regardless of scale. Fortunately, there are many methods to extract depth information; this section provides a brief review of several options for resolving depth using vision sensors, and an analysis of their potential utility for space-based applications.

One option to compute a depth map (i.e. a mapping of distance for every pixel in the image) is using an additional non-optical sensor such as LIDAR. However, the size, weight, and power (SWAP), cost, and complexity associated with these types of sensors often does not make sense for space applications. Alternatively, several methods to estimate depth using optical sensors exist, such as Structure From Motion (SFM) [50], structured light [51, 52], depth from focus/defocus [53], plenoptic (light field) cameras [54], and depth from stereo [55].

SFM algorithms exploit the epipolar geometry of a scene observed from multiple angles [56]. A known distance is required to compute scale in the scene, which may be the size of a known object in the image scene (unlikely to be the case with an unknown target) or knowledge of how far the camera moves between images (odometry). An advantage of this method is that it can be implemented with a monocular camera. However, while the amount of movement can be calculated using other sensors such as GPS, depth estimation is degraded by this distance measurement and the fact that the target may move in the time between successive image frames being captured.

Structured light refers to the use of a projector to cast a known pattern of light (usually in the infrared spectrum) on the image scene. Depth can be inferred by observing how the light pattern bends over objects in the scene. This type of depth estimation is not ideal as a stand-alone method for the space environment primarily because "strong ambient illumination severely degrades the performance of structured light based techniques" [57]. [57] proposes that performance of this method in direct sunlight can be improved by controlling the distribution of the structured light cast onto the scene, but this alone does not significantly increase its potential. Note that structured light can be combined with other methods such as stereo to improve overall depth estimation. This combination could be useful for space applications, especially in situations where direct sunlight is not present in the image scene.

Depth from focus/defocus methods exploit the limited depth of field that is inherent in real aperture camera models. Given two images of the same scene taken with different camera parameters, the depth in the scene can be computed using the relative focus of portions of each image. A benefit of this method is that is can be performed using a monocular camera. However, because this method relies on an accurate estimation of the amount of blur in an image, it is highly susceptible to error due to image noise and movement, which would likely make its utility in space limited. Additionally, the

usable range for the method is dependent of the size of the camera aperture, which further limits its applicability. [53] shows that the mathematical principles to obtain depth from focus/defocus are not so different from that of depth from stereo, and the method is therefore subject to many of the same limitations as stereo.

Plenoptic, or light field cameras, use the concept of a four-dimensional light field to extract depth information. To do this, a micro-lens array placed behind the main camera lens generates many images of the same scene, from which depth can be inferred in much the same way as depth from stereo. Processing the depth map of a single camera frame is very computationally expensive, however, so this method's potential for real-time use is currently limited. Furthermore, as the number of microlenses increases, spatial resolution decreases, which reduces functionality with feature extraction algorithms. Use of plenoptic cameras for spacecraft RPO was examined in [58], and while the results are noteworthy for the limited use case considered (less than 3 meters distance inspection using a robotic arm), the authors note that the accuracy of the method is highly degraded when the camera or objects in its field of view are moving. Given these limitations, currently available plenoptic camera technology does not represent a worthwhile option for conducting visual navigation during spacecraft RPO.

Finally, depth from stereo works by using the epipolar geometry of a scene and the known baseline distance between cameras to triangulate world points and thus provide full 3D information of a scene. In this way, it is similar to SFM, except the distance between camera viewpoints is known, and images are recorded at the same time, so target motion does not affect depth estimation. Stereo cameras represent a compelling class of sensor for space applications, especially considering many commercial-off-the-shelf (COTS) options now exist that provide aligned depth maps in real time (greater than 30 frames per second (FPS) or faster). An example color image and aligned depth map from a low-cost COTS camera is shown in Figure 2.11.

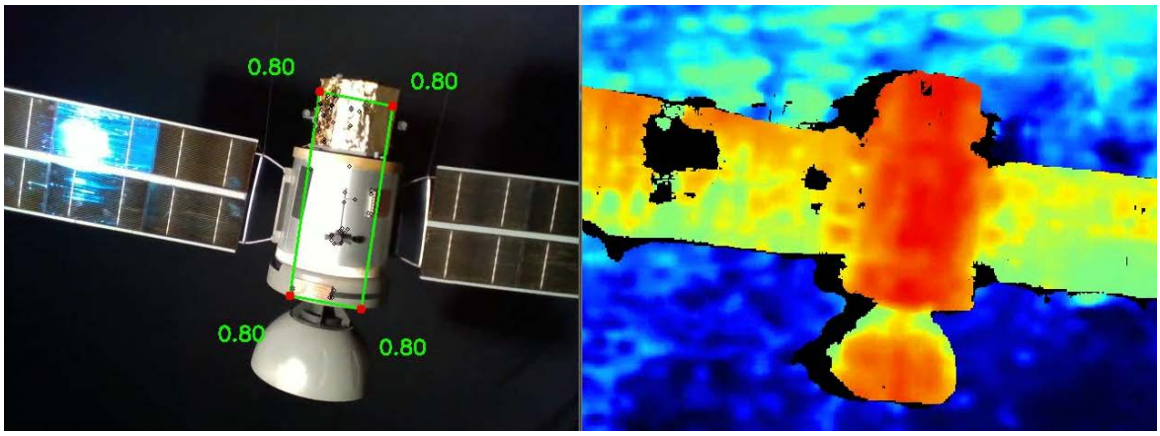

Figure 2.11: Example color image and aligned depth map of satellite mock-up using COTS stereo camera sensor (Intel RealSense d435i). Numbers in color image indicate depth to corresponding corner of region of interest.

2.4.5 Computer Vision in the Space Environment.

The orbital environment presents a number of challenges for the aforementioned image processing algorithms due to potentially large changes in many attributes of a given image scene. These include changes in the relative position (scale) and attitude (rotation) of the target, as well as changes in illumination that will manifest as varying levels of brightness and different shadows as images of the same scene are taken at different times. Therefore, it is important to analyze how the aforementioned image processing techniques will work in the space environment. Various research efforts have examined computer vision processes using imagery from actual space missions or space-like mock-ups in an effort to ascertain the utility of classical image processing algorithms in space. A brief summary of several of these studies is presented below.

Palmerini et al. [59] present an analysis of vision based techniques for close-in RPO. A simulation campaign is presented that utilizes an on-orbit image of the Progress spacecraft. The work primarily examines how SIFT features are identified as the image is rotated, scaled, and also warped via perspective transformations (the concept of a perspective warp is discussed in Section 3.3 of this work). Though the results show that

SIFT features can be impressively re-identified as the image is altered (for example as the scale of the image changes from 0.5x-2.0x), the overall value of the result is limited as only a single image was considered (importantly, the effect of changing lighting conditions was not examined).

Results of an experimental test campaign are presented in [60] which feature a scale mock-up of a satellite target placed in a space-like environment (dark room with single light source approximating the Sun). Image processing is composed of first identifying SIFT features in a reference image of the target (with the target set at a distance of 30 cm from the camera). Images are then taken of the target at various ranges and angles. Matched SIFT features in these subsequent images are used to estimate the scale change from the reference image, which can provide an estimate of distance from the target. Results show that a sufficient number of SIFT features are able to be continuously tracked, and the error of the estimated distance is kept to within 1 cm as the distance from the camera to the target increases to 90 cm. Additional experimental results are presented that demonstrate a basic visual navigation and control scheme. For navigation, a linear Kalman filter is used to estimate the relative position of the camera based on distance estimates from image processing step. A Linear Quadratic Regulator (LQR) is used to control the in-plane position of the camera. Results demonstrate a successful rendezvous maneuver using purely visual information for feedback. The authors conclude that an optical instrument can be considered as a standalone sensor for on-orbit applications, though it is important to note that this study appears to have not addressed the effect of changing lighting conditions as spacecraft orbit the Earth.

The German Aerospace Center (DLR) operates the European Proximity Operations Simulator (EPOS) 2.0 [61]. EPOS represents hardware-in-the-loop simulation facility designed for test and verification of RPO sensors and processes. The facility features high-fidelity approximation of the space environment via accurate lighting, background

and target spacecraft. Recent research using EPOS such as [62] shows that conducting close-in RPO using primarily vision sensors for relative navigation is certainly viable.

A number of studies have examined spacecraft visual navigation using 3D model based tracking methods either using actual space imagery [63–65] or space-like laboratory mock-ups [66]. The results of these studies show that the necessary computer vision methods can work well in the space environment. However, since these studies largely rely on different computer vision methods such as edge detection, they do not definitively demonstrate the efficacy of point feature-based methods.

Although not strictly space-related, it is worth noting that several research efforts in the field of computer vision have examined removing shadows from images [67–70], which would be a very useful capability in the space environment. The supposition of these methods is based on the idea that humans are very good at filtering out the effects of shadows on objects in a scene, therefore it should be possible for computer vision algorithms to be good at this as well. The shadow removal methods proposed generally work by first creating an illumination invariant gray-scale image from an input color image. This illumination invariant image does not contain the shadows present in the original image, so it can be used in conjunction with the original image to determine the location of shadow edges. These known locations can then be used to remove shadows in the original image via a variety of methods as proposed in [67–70]. While results of current state of the art shadow removal methods often look impressive to the human eye, and might be useful for thresholding techniques, they will likely not significantly increase performance of feature-based image processing methods. The investigation of this topic for space operations may be worthy of future research, however.

The studies referenced in this section all consider some (usually fixed) amount of illumination from the Sun on the target. Therefore, they do not fully quantify how the the inherent illumination changes of an arbitrary spacecraft orbit will affect the ability for

computer vision processes to identify and track target over the course of an entire orbit (it is unknown if Earth's albedo would be sufficient to properly illuminate a target when the Sun is eclipsed, as this has not been assessed in available literature). However, it is worth keeping in mind that the severity and frequency of illumination changes depends on the orbital regime for a given scenario. For example, a camera in a geostationary orbit will observe fewer changes in lighting over the same time frame as one in low Earth orbit, and an RPO maneuver conducted over a shorter time frame will experience less change in lighting than one over a longer time frame. When designing missions using visual sensors on-orbit, it is important to consider how these factors could affect the success of the mission.

In summary, these studies suggest that traditional image processing techniques function under space-like conditions in circumstances of sufficient illumination, though work remains to be done to more definitively show that feature-based navigation and control methods are effective for space applications.

2.5 Estimation Theory

A key requirement of a navigation system is to obtain an estimate of the relevant states of a system. In most cases, system states cannot be directly measured, and in all cases, sensor measurements contain noise. Therefore, estimation algorithms are used to produce a judicial guess of the actual state of a system.

In the context of state estimation, the full system model for a stochastic system is typically composed of a process model (2.38) and a measurement model (2.39), both of which may be nonlinear and are subject to process noise and measurement noise, respectively. Process noise captures effects of unmodeled dynamics and disturbances, and measurement noise occurs as a result of sensor imperfections. Here both the process and measurement noise are assumed to be additive white Gaussian. The full system model is given by

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{G}(t)\mathbf{w}(t), \tag{2.38}$$

$$\mathbf{z}(t) = \boldsymbol{h}(\mathbf{x}(t), t) + \mathbf{v}(t), \tag{2.39}$$

where $\boldsymbol{x}$ is the state vector, $f(\mathbf{x}(t), \mathbf{u}(t), t)$ is the process model (a function of the state, control effort $\boldsymbol{u}(t)$ and time), $\boldsymbol{G}(t)$ is the process noise influence matrix, $\boldsymbol{w}(t)$ is a white Gaussian noise disturbance, $\boldsymbol{h}(\boldsymbol{x}(t), t)$ is the measurement model, and $\boldsymbol{v}(t)$ is a white Gaussian noise process.

Most estimation methods either process measurement data as a batch, or recursively. The latter option, known commonly as filtering, is often used for real-time applications and generally operates by using the above model to perform two steps: a propagation step and an update step. The propagation step uses the previous estimate of the state, the applied control effort (if known), and the process model to propagate the state forward to the current time. This becomes the *a priori* state estimate for the current time, which is then used in the update step along with a sensor measurement and the measurement model to further refine the state estimate based on new data. A thorough development of stochastic systems and estimation theory can be found in [71, 72].

2.5.1 The Linear Kalman Filter (LKF).

The Kalman filter and its derivatives represent the canonical family of estimators for real-time applications. In its basic (linear) form, a Kalman filter is a recursive data processing algorithm that is optimal in minimizing mean square error (the difference between the true state and estimated state) [71]. For optimality to hold, however, the system model (process model and measurement model) must be linear and noise must be zero-mean white Gaussian with known variance. The general algorithm for a discrete Linear Kalman Filter (LKF) is shown in Algorithm 1.

The linearity assumption in a basic Kalman filter limits its applicability to nonlinear estimation. In certain cases however, a nonlinear system can be approximated by a similar linear system and a Kalman filter can still be used effectively.

Algorithm 1 Linear Kalman Filter

procedure LKF($\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{P}_{k-1}, \boldsymbol{z}_k, \boldsymbol{u}_k, \boldsymbol{\Phi}, \boldsymbol{H}, \boldsymbol{B}, \boldsymbol{Q}_k, \boldsymbol{R}_k$) ▷ Calculate one LKF iteration
  $\hat{\boldsymbol{x}}_k^- = \boldsymbol{\Phi}(k, k-1)\hat{\boldsymbol{x}}_{k-1}^+ + \boldsymbol{B}\boldsymbol{u}_k$ ▷ **Propagate** state
  $\boldsymbol{P}_k^- = \boldsymbol{\Phi}(k, k-1)\boldsymbol{P}_{k-1}^+\boldsymbol{\Phi}^T(k, k-1) + \boldsymbol{Q}$ ▷ **Propagate** covariance
  if new measurement available **then**
    $\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}^T\left[\boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^T + \boldsymbol{R}\right]^{-1}$ ▷ Calculate Kalman gain
    $\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\left(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-\right)$ ▷ **Update** state estimate
    $\boldsymbol{P}_k^+ = \boldsymbol{P}_k^- - \boldsymbol{K}_k\boldsymbol{H}\boldsymbol{P}_k^-$ ▷ **Update** covariance estimate
  end if
end procedure

$\hat{\boldsymbol{x}}$ - State estimate
$\boldsymbol{P}$ - Covariance estimate
$\boldsymbol{z}$ - Measurement
$\boldsymbol{\Phi}$ - State transition matrix
$\boldsymbol{H}$ - Measurement matrix
$\boldsymbol{B}$ - Control matrix
$\boldsymbol{Q}$ - Process covariance
$\boldsymbol{R}$ - Measurement covariance

2.5.2 The Extended Kalman Filter (EKF).

It is often not the case that both the dynamics model and measurement model are linear, or can be suitably approximated by linear systems. For these circumstances, the Extended Kalman Filter (EKF) is available. It does not require the process model or the measurement model to be linear. Rather, it implements a first-order approximation (by taking a Jacobian) of these nonlinear functions, using the current state estimate as an operating point for the linear approximation. It then performs propagation and update steps in a similar manner as standard Kalman filters, using these linearized approximations of the process and measurement model. The general algorithm for a discrete EKF is shown in Algorithm 2.

Importantly, because the EKF operates on approximations of the actual system, it is not necessarily optimal in the same sense that a standard Kalman filter is. Depending on how nonlinear a system is or how rapidly its dynamics change, the EKF's first-order approximation of the system may not accurately model the system. Furthermore, this approximation requires taking a Jacobian, which can often be difficult to calculate (either analytically or numerically).

Algorithm 2 Extended Kalman Filter

procedure EKF($\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{P}_{k-1}, \boldsymbol{z}_k, \boldsymbol{u}_k, \boldsymbol{B}, \boldsymbol{Q}_k, \boldsymbol{R}_k$) ▷ Calculate one EKF iteration

$\boldsymbol{F}_k = \left.\dfrac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_k, t)}{\partial \boldsymbol{x}(t)}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}^{+}}$

$\boldsymbol{\Phi}(k, k-1) = e^{\boldsymbol{F}_k}$

$\boldsymbol{H}_k = \left.\dfrac{\partial \boldsymbol{h}(\boldsymbol{x}, t)}{\partial \boldsymbol{x}(t)}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k}^{-}}$

$\hat{\boldsymbol{x}}_k^- = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}^{+}, \boldsymbol{u}_k, k)$ ▷ **Propagate** state

$\boldsymbol{P}_k^- = \boldsymbol{\Phi}(k, k-1)\boldsymbol{P}_{k-1}^{+}\boldsymbol{\Phi}^T(k, k-1) + \boldsymbol{Q}$ ▷ **Propagate** covariance

if new measurement available **then**

$\boldsymbol{K_k} = \boldsymbol{P}_k^- \boldsymbol{H}^T \left[\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^T + \boldsymbol{R}\right]^{-1}$ ▷ Calculate Kalman gain

$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\left(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-\right)$ ▷ **Update** state estimate

$\boldsymbol{P}_k^+ = \boldsymbol{P}_k^- - \boldsymbol{K}_k\boldsymbol{H}\boldsymbol{P}_k^-$ ▷ **Update** covariance estimate

end if

end procedure

$\hat{\boldsymbol{x}}$ - State estimate
$\boldsymbol{P}$ - Covariance estimate
$\boldsymbol{z}$ - Measurement
$\boldsymbol{\Phi}$ - State transition matrix
$\boldsymbol{H}$ - Measurement matrix
$\boldsymbol{B}$ - Control matrix
$\boldsymbol{Q}$ - Process covariance
$\boldsymbol{R}$ - Measurement covariance

The aforementioned filters are often a critical component of navigation systems. Not only are they useful for generating estimates of system states that are more accurate than raw measurements alone, they enable states that cannot be directly measured to be estimated as well. They are used herein for these reasons, and they are the foundation for many approaches taken e.g. to determine relative pose between spacecraft (the topic of the next section).

2.6 Visual Navigation for Unknown Targets - Relative Pose Determination

The ultimate goal of visual navigation schemes designed for use with traditional control methods is to determine the relative pose between the chaser spacecraft and its target (as the name implies, using some sort of vision sensor for measurement). This is a critical problem: as will be seen in Section 2.7, most previously proposed control methods require an accurate estimate of the full relative pose to function properly. Relative

pose determination for unknown targets is challenging however, so it has therefore been the subject of much research (a review of much the available research on this topic as of 2017 can be found in [73]). This section summarizes some common methods for pose determination of unknown space objects, and reviews several of the most relevant research efforts for the problem. The aim of this section is to show that relative pose determination for an unknown RSO is not a robustly solved problem, which will in turn motivate the examination of control methods that do not require full relative pose estimates (Section 2.8).

Pose determination of unknown targets is a simultaneous localization and mapping (SLAM) problem, complicated by the fact that the environment being mapped (the target spacecraft) may not be stationary relative to the camera sensor. Many different SLAM methods have been developed that could be applied to this type of problem. The classical method is to use an EKF that incorporates identified features in the state vector in addition to the dynamical states of interest [74]. Indeed, this is the approach that many of the reviewed studies below take. However, EKF-SLAM methods suffer from poor computational efficiency, especially as the number of features being tracked grows (computation increases quadratically with number of features [75]). Other more modern variants of SLAM exist that aim to be more efficient/effective than EKF-SLAM, such as FastSLAM [76, 77] and graph-based SLAM methods [78, 79].

FastSLAM utilizes a Rao-Blackwellized particle filter [80], which operates by partitioning the state vector into particle states and non-particle states, thereby reducing the dimensionality of the particle filter to make it more computationally feasible. The non-particle states (typically the map of features) are estimated via EKF.

The basic idea of graph-based methods is to represent that problem as a graph, where every node corresponds to a pose during the mapping process, and every edge a constraint imposed by measurements. The goal of graph-based SLAM is to build a map of nodes that

minimizes the error introduced by edge constraints (see Figure 2.12 for an example map). An optimization technique such as the Gauss-Newton algorithm is used to perform this minimization. A benefit of graph-based SLAM is that this method solves for the entire pose trajectory using all previous measurements to do so, as opposed to filter-based methods, which only use the current measurement to refine the estimate of pose.

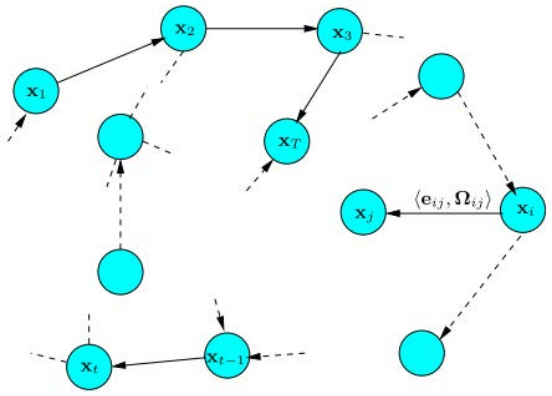

Figure 2.12: Example pose-graph depicting poses (nodes) and constraints (edges) [79]

Sonneburg et al. develop an EKF-SLAM algorithm for pose estimation of unknown RSO's [81]. The work is conducted in simulation and assumes a set of point features derived from a stereo camera are provided, it therefore does not consider the computer vision front-end algorithms necessary to generate these features (like many of the works reviewed in this section). The algorithm follows a traditional EKF-SLAM approach, with a state vector composed of relative position, velocity, quaternion, angular velocity, and 3D feature locations. Importantly, the filter does not estimate moments of inertia for the target (in fact, for one of cases in the results section involving a nutating target, the target's moment of inertia is assumed to be known). The numerical simulation results are sparse, and there is no discussion of why measurement noise was modeled as it was, or how realistic it might be. Furthermore, there is no discussion of how efficiently the algorithm

would work on board flight hardware. Consequently, it is difficult to assess the utility of this algorithm in a realistic environment using actual computer vision processing and hardware.

A hybrid SLAM/SFM method is developed in [82] using a monocular camera. For the SLAM approach, a variant of FastSLAM 2.0 [77] using a Rao-Blackwellised Particle Filter is implemented. A Bayesian filtering approach is used for the rotational states and a measurement inversion method is used for the translational states. Because a monocular camera is assumed as the relative navigation sensor, a SFM algorithm is required to recover relative pose measurements. There is no discussion of how scale is recovered using this method. Moments of inertia for the target are not estimated (the rotational process equations assume constant angular velocity of the target). The authors conclude that their hybrid approach is better than a purely Bayesian approach based on simulation and experimental results, however the experimental results from underwater testing using a remotely operated vehicle moving in a loop around an anchored target appear to show insufficient levels of accuracy needed for spacecraft RPO maneuvers. Given the assumptions (lack of proper dynamics model) and limitations (does not compute scale), this method is likely not mature enough for use on orbit.

In [83], the EKF-SLAM filter developed in [81] is augmented with a RANSAC-based algorithm to reconstruct the target's structure. The RANSAC-based method (from [84]) is used to reconstruct surfaces on the target from point clouds generated using a stereo camera. In this way, a 3D model of the target can be constructed for use with other navigation tasks (though the work does not discuss what other specific tasks this model could improve). Experimental results using a feature-based image processing pipeline (with SURF features) demonstrate the method's ability to recover a target's shape from point clouds, but computational time for one processing cycle is quoted as 1.45 seconds or greater—insufficient for real-time implementation.

Tweedle presents a novel SLAM implementation in his dissertation [85] and follow-on paper [86] designed for use with unknown spinning targets. His method is based on SLAM approach called incremental smoothing and mapping [87], and is formulated for rapidly spinning targets. The method was verified using a dataset from a robotic testbed onboard the International Space Station, and results show the algorithm's ability to build a geometric map of the target while simultaneously tracking the relative pose from camera to target. While Tweddle's work represents some of the most significant research regarding relative navigation with unknown targets, there are a number of notable assumptions and limitations. Namely, the proposed pose estimation algorithm was not run online—data from the experiment was processed afterwards using an off-board computer. It is noted that the smoothing approach implemented is computationally intensive, so its ability to run in real time is still unknown. Additionally, the chaser spacecraft was assumed to be static, and no external forces or torques on the spacecraft were considered.

An Iterated Extended Kalman Filter (IEKF) SLAM-based pose-estimation filtering scheme that utilizes a stereo camera is proposed in [88]. This work is notable for its method of estimating the target's inertia tensor through the use of a bank of IEKF's (a bank of five is considered in the work). Each filter is endowed with a different estimate of the target's moment of inertia and a maximum *a posteriori* method is used to select the filter with the most likely inertia matrix. Simulation results convey the method's robustness to inaccurate knowledge of the target's inertia, and the authors also claim that the algorithm is robust to measurement noise from the camera sensor (based on subsequent simulations with increasing levels of noise). However, it appears that the levels of measurement noise used (up to $5x10^{-4}$ radians) are unrealistically low, as noted in [9], and are not justified. Experimental results in a laboratory setting using the Israel Institute of Technology's Distributed Space Systems Laboratory (air bearing table with independent satellite models) are also presented. While results of a single test case appear to show the algorithm doing

a good job of estimating the relative pose as well as target inertia, there is no discussion of computational efficiency of the method (it is thought to be too high to implement on flight hardware according to other researchers [9, 89]). Additionally, the authors note that questions remain about how the method would scale up to orbital distances and actual space lighting conditions.

The development of an IEKF SLAM-based method in [89] is in many ways an extension of [88], and the authors go so far as to use similar tuning and noise parameters to facilitate direct comparison with [88]. The main difference between [89] and [88] is that the target's inertia tensor is directly estimated in [89], as opposed to [88] which uses a bank of filters each with a different inertia hypothesis. To estimate the target inertia (up to a scale factor), two scalars are appended to the filter state vector, representing ratios of the target's principal moments of inertia, as proposed in [85]. The method is examined in simulation using very similar measurement noise parameters as [88]. This is the main factor that makes applicability of this work in a realistic setting questionable: the maximum measurement noise assessed ($1x10^{-4}$ radians) seems unrealistically low. Although there is a brief discussion of the choice of this value (one rational being because [88] used it, though it was not justified in that paper), it remains unclear how noise with units of radians applies to measurements with units of pixels. Though the results presented show good estimation of all filter states, it is unknown how well this method would work in practice. The authors appear to acknowledge this fact by stating "the proposed estimation procedure could not be the best choice for missions with low cost hardware or without a very good computer vision architecture" [89].

Capuano et al. propose another variation of an EKF-SLAM filter utilizing a monocular camera and depth sensor [90]. The same state vector as [89] is used, and nonlinear relative motion dynamics are used to propagate the state estimate (as opposed to [82]). Additionally, a method is proposed to initialize new features and delete obsolete features.

The algorithm is assessed via a numerical simulation, using a Harris corner detector [91] on synthetic imagery. Results are only shown for estimation error of relative position and attitude—while attitude estimation is good, position error has a bias of approximately 2.5 meters due to depth estimation error (the paper does not say what the relative range between spacecraft was). Results for how well other parameters are estimated, such and the target's inertia tensor, are not presented. It is unclear how well this method would work in an experimental setting with flight hardware.

As [9] concludes and the results of these studies indicate, current methods proposed for relative pose estimation of unknown non-cooperative targets each have their own shortcomings, limiting their potential for use in actual space missions. The main implication of this is that at present many proposed control methods which require a full relative pose estimate are not viable for this type of RPO scenario. This is the basis for one of the main objectives of this work: to show that final approach maneuvers can be accomplished without first establishing a full relative pose estimate.

2.7 Control Methods for Final Approach Maneuvers

Many studies have examined control methods for close proximity 6DOF maneuvers. For example, several efforts have examined using Artificial Potential Field (APF) controllers for close-in maneuvers due to their ability to mitigate collisions (e.g. [92, 93]). Other control techniques have been examined as well, such as: LQR (e.g. [94, 95]), model predictive control (e.g. [96]), sliding mode control (e.g. [97, 98]), and adaptive backstepping control (e.g. [99]).

The intent of this section is not to review the relative merits or disadvantages of each proposed control method, but to convey that all the studies reviewed assume that the full relative pose between the chaser and target is provided. This is commonly stated as in [93]: "it is assumed that the motion information including relative position, velocity, attitude and angular velocity can be measured if the target is in the sensing field of the pursuer."

While this may be a fair assumption for cooperative RPO or even non-cooperative RPO with known targets, this is not necessarily a realistic assumption for RPO with unknown targets, as Section 2.6 showed. As a result of this assumption, these studies do not adequately demonstrate robustness to navigation errors likely to be present during non-cooperative/unknown RPO maneuvers. This again motivates the examination of control techniques that do not require full relative pose information between the chaser and target.

2.8 Visual Servo Control

Visual servo control is a general term that refers to the use of vision information in a feedback loop to control the motion of a robotic system. Using visual feedback for control was originally proposed in the 1970's [100, 101], and many of the core concepts of modern visual servo control were formalized in the 1990's [102–105]. Since then, visual servoing has become common practice in the field of robotics.

A visual servo system can either be configured as eye-in-hand, where the camera sensor is directly attached to the robotic effector (chaser spacecraft in the context of this work), or eye-to-hand, where the camera sensor is physically dislocated from the effector and observes movement of the effector with respect to the target [106]. While an eye-to-hand configuration is theoretically possible for the RPO scenarios considered in this work, it would require a third cooperative spacecraft, which would drastically increase mission requirements and therefore decrease feasibility. As such, visual servoing with eye-to-hand configuration is not considered in this work, though it may warrent future research.

Traditionally there are considered to be two main approaches to visual servo control: Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS) [27, 104, 107, 108]. This section describes describes the fundamentals of these two approaches, as well as two popular variations: 2-1/2D Visual Servoing and Partitioned Visual Servoing. The section concludes with a review of research that has examined visual servoing concepts for spacecraft RPO.

2.8.1 Position-Based Visual Servoing (PBVS).

PBVS computes control commands in the task space (i.e. in 6DOF for spacecraft RPO) [105]. These control commands are produced using the error between estimated relative pose ($\hat{\boldsymbol{T}}_G^C$) and desired relative pose ($\boldsymbol{T}_G^{C^*}$). A basic PBVS architecture is shown in Figure 2.13—image features are used in conjunction with other knowledge of the target to estimate the relative pose between the current camera pose and desired goal pose.

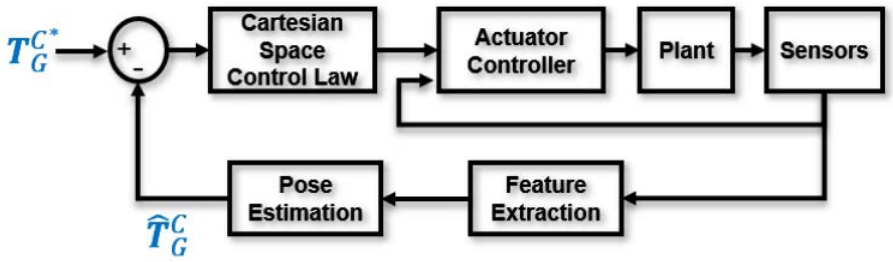

Figure 2.13: Position-Based Visual Servoing architecture

Fundamental to the success of a PBVS system is an accurate relative pose estimate, which requires a well-calibrated camera and known 3D/2D target feature correspondences [104]. Typically, feature correspondences are computed using a known 3D model of the target. For this application, since the target is unknown, a pose estimation scheme like those discussed in Section 2.6 would have to be used. Since obtaining an accurate relative pose estimation of an unknown target spacecraft is still an area of open research, it seems apparent that the pose estimation step of any PBVS for use with unknown targets would be the main limiting factor regarding overall effectiveness.

Additionally, because the control error is computed in Cartesian space, there is no way to ensure that the target image features will remain in the camera's field of view [109]. Furthermore, according to [110], "since the error made on the pose estimation

cannot be computed analytically as a function of the camera calibration errors, it seems to be impossible to analyze the stability of the system."

Due to the limitations discussed in this section, traditional PBVS is not considered in this work.

2.8.2 Image-Based Visual Servoing (IBVS).

IBVS is fundamentally different than PBVS in that it uses pixel locations of image features in the image plane to compute control commands. Consequently, it is generally acknowledged that IBVS is more robust to image noise and errors in camera calibration [27, 107]. Given a set of n point features observed in the current image frame, $\boldsymbol{s} = [u_1, v_1, ..., u_n, v_n]$, that can be matched to features from a goal image, $\boldsymbol{s}^*$, the controller seeks to drive the pixel locations of these features from their current locations to desired locations in the image. The goal image represents the view of the target from the desired pose. Since the pixel locations of features in the current image frame are a function of the camera pose in Cartesian space, if they are aligned with their location in the goal image, the camera has implicitly achieved its goal pose (a contribution of this research effort will be to propose methods to generate goal images for unknown targets, see Section 3.3). Fig. 3.1 depicts the basic concept of IBVS for an example square-shaped target.

2.8.2.1 The Interaction Matrix.

Central to the IBVS control law is the concept of the image interaction matrix (alternatively called the image Jacobian). This matrix relates the translational and angular velocity of a camera with respect to a world frame, $\boldsymbol{V} = [\boldsymbol{v}_c, \boldsymbol{\omega}_c]^T \in \mathbb{R}^6$, to the pixel velocities of features in the image plane, $\dot{\bar{\boldsymbol{s}}} = [\dot{x}_1, \dot{y}_1, ..., \dot{x}_n, \dot{y}_n]^T$ (overbar indicates vector expressed using normalized coordinates)

$$\dot{\bar{\boldsymbol{s}}} = \boldsymbol{L}_x(\bar{\boldsymbol{s}}, Z)\boldsymbol{V}. \tag{2.40}$$

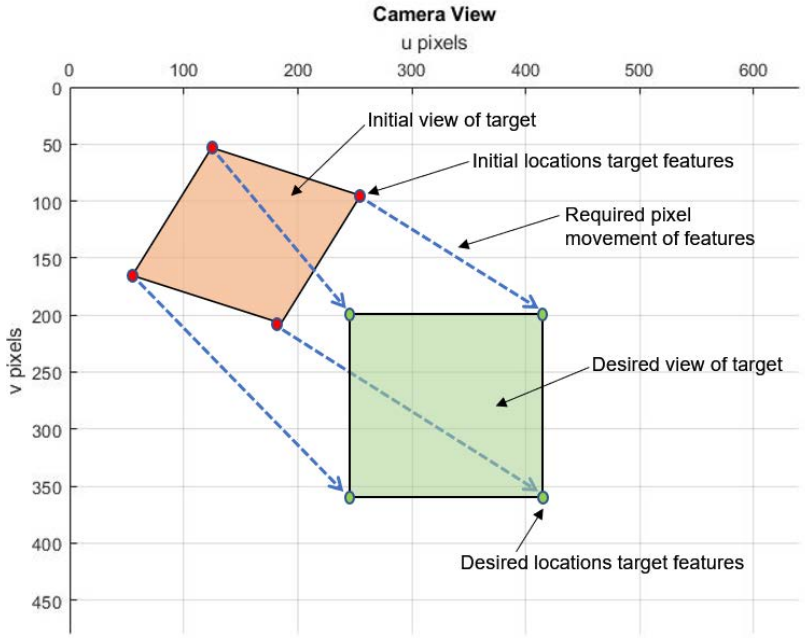

Figure 2.14: Basics of image-based control (adapted from [27])

To derive $\boldsymbol{L}_x$, consider a single point feature $\boldsymbol{p} = [X, Y, Z]^T$ resolved in the camera frame. Using the transport theorem, the time derivative of $\boldsymbol{p}$ in the camera frame is

$$\dot{\boldsymbol{p}} = -\boldsymbol{v}_c - \boldsymbol{\omega}_c \times \boldsymbol{p}. \tag{2.41}$$

The scalar components of this vector equation are

$$\begin{aligned} \dot{X} &= Y\omega_z - Z\omega_y - v_x \\ ,\dot{Y} &= Z\omega_x - X\omega_z - v_y \\ ,\dot{Z} &= X\omega_y - Y\omega_x - v_z. \end{aligned} \tag{2.42}$$

Recall the normalized image plane coordinates $x = X/Z,\ y = Y/Z$ from (2.27). The time derivatives of these coordinates are

$$\dot{x} = \frac{\dot{X}Z - X\dot{Z}}{Z^2}$$
$$, \dot{y} = \frac{\dot{Y}Z - Y\dot{Z}}{Z^2} \tag{2.43}$$
$$.$$

Using (2.43), (2.42) can be reformulated as a matrix equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & (1+y^2) & -xy & -x \end{bmatrix}}_{\boldsymbol{L}_x} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{2.44}$$

Given that pixel coordinates are provided as measurements, normalized coordinates $\bar{\boldsymbol{s}} = [x, y]^T$ and $\dot{\bar{\boldsymbol{s}}} = [\dot{x}, \dot{y}]^T$ can be generated from pixel coordinates ($\boldsymbol{s} = [u, v]^T$) using (2.27)

$$x = \frac{u - c_u}{f_u}, \quad \dot{x} = \frac{\dot{u}}{f_u}$$
$$, y = \frac{v - c_v}{f_v}, \quad \dot{y} = \frac{\dot{v}}{f_v}. \tag{2.45}$$

The $\boldsymbol{L}_x$ in (2.44) is a 2×6 matrix for a single point feature, making the equation underdetermined and rank deficient. This can be overcome by vertically concatenating the interaction matrix for multiple features

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \vdots \\ \dot{x}_n \\ \dot{y}_n \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{L}_{x_1}(\boldsymbol{p}_1, Z_1) \\ \boldsymbol{L}_{x_2}(\boldsymbol{p}_2, Z_2) \\ \vdots \\ \boldsymbol{L}_{x_n}(\boldsymbol{p}_n, Z_n) \end{bmatrix}}_{\boldsymbol{L}_x} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{2.46}$$

If three points are used and they are not coincident or colinear, the combined interaction matrix will in general be non-singular [27]. In practice, more than 3 points are normally used due to noise in the image feature locations.

Note that the depth to the feature, Z, is present in the interaction matrix. As a result, depth must be estimated to use this relationship (see Section 2.4.4 for a review of depth estimation methods).

Finally, it is also worth noting that interaction matrices can be derived for other types of image features such as lines, planes, circles, etc. as discussed in [27, 102]. Though analysis of IBVS using these types of image primitives is beyond the scope of this work, this subject is certainly worthy of future research, as it may be that these types of features can be more readily identified than point features in the space environment.

2.8.2.2 IBVS Control Law.

The implementation of IBVS considered herein is of the "dynamic look-and-move" type described in [104]. This type is depicted in Figure 2.15 characterized by an IBVS control law that computes setpoint inputs for a separate actuator controller, which are then used to produce control commands for the system.

To develop the IBVS control law, the error between matched features in the goal image and the current images is

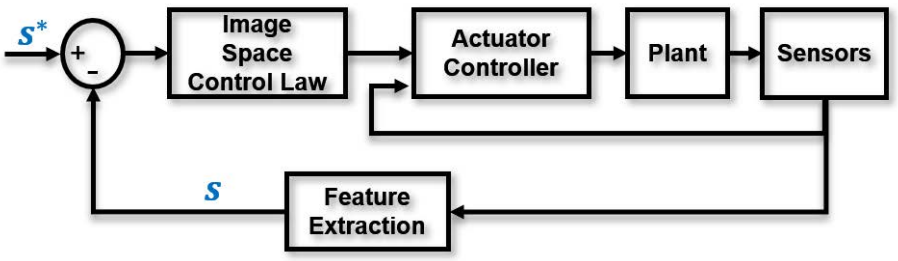

Figure 2.15: "dynamic look-and-move" image-based control (adapted from [104])

$$\boldsymbol{e}(t) = \bar{\boldsymbol{s}}(\boldsymbol{p}_j, \boldsymbol{C}) - \bar{\boldsymbol{s}}^*, \tag{2.47}$$

where the set of image features $\boldsymbol{s}$ is a function of 3D target features, $\boldsymbol{p}_j$ and intrinsic/extrinsic camera parameters, $\boldsymbol{C}$. Then, using (2.40), it follows that

$$\dot{\boldsymbol{e}} = \boldsymbol{L}_e \boldsymbol{V}. \tag{2.48}$$

Applying a linear controller, $\dot{\boldsymbol{e}} = -\lambda \boldsymbol{e}$, the velocity required to obtain an exponential decrease of $\boldsymbol{e}$ is

$$\boldsymbol{V}_{cmd} = -\lambda \widehat{\boldsymbol{L}_e^+} \boldsymbol{e}, \tag{2.49}$$

where $\widehat{\boldsymbol{L}_e}$ is an approximation of the interaction matrix, and λ is a tunable gain parameter (which can be implemented as a gain matrix to affect each velocity component differently). Assuming more than three features are considered, the Moore-Penrose pseudo-inverse of the interaction matrix approximation, $\widehat{\boldsymbol{L}_e^+}$, is used as a least-squares solution for the overdetermined equation.

Several options have been proposed in literature to generate the approximation of $\widehat{\boldsymbol{L}_e}$. The most straightforward and intuitive option is to set $\widehat{\boldsymbol{L}_e} = \widehat{\boldsymbol{L}_x}$. Using this approximation, the trajectories of feature points in the image plane are almost straight lines. While this

may be ideal in the image plane, this approximation generates a Cartesian trajectory that is often less than ideal.

Another option is to set $\widehat{\boldsymbol{L}_e} = \widehat{\boldsymbol{L}_{x^*}}$. That is, the approximate interaction matrix is set to the estimated interaction matrix for the goal pose. While this method displays asymptotic convergence when the camera is in the vicinity of the goal pose, feature point trajectories in the image plane and Cartesian camera trajectory are less than ideal when the initial camera displacement is large.

Chaumette introduced a degenerate case of IBVS in [109], where the camera makes non-intuitive and potentially unrealizable motion away from the target. This phenomenon is commonly known as camera retreat, and it occurs when a rotation of π rad (or nearly π rad) about the optical axis is required to achieve the goal pose. In this case, a large translation along the optical axis away from the target is commanded which is clearly suboptimal and sometimes unachievable (this translation approaches infinity as the required rotation approaches π radians). Malis proposes a simple method to reduce the amount of camera retreat that occurs as required optical axis rotation nears π rad in [111]. $\widehat{\boldsymbol{L}_e}$ is set to the average of the interaction matrix for the current frame and for the goal frame:

$$\widehat{\boldsymbol{L}_e} = \frac{1}{2}(\widehat{\boldsymbol{L}_x} + \widehat{\boldsymbol{L}_{x^*}}) \tag{2.50}$$

In addition to minimizing the effects of camera retreat, this method generally results in smooth trajectories in both the image plane and 3D space.

2.8.2.3 Stability of IBVS.

The stability of IBVS has been examined in numerous publications [107, 109, 112–114]. This section provides a brief analysis on the stability of IBVS using Lyapunov's direct method.

Consider the Lyapunov function

$$\mathcal{L} = \frac{1}{2}\|\boldsymbol{e}(t)\|^2. \tag{2.51}$$

The derivative of this function is (formed by inserting (2.49) into (2.48))

$$\dot{\mathcal{L}} = \boldsymbol{e}^T\dot{\boldsymbol{e}} \tag{2.52}$$

$$= -\lambda \boldsymbol{e}^T \boldsymbol{L}_e \widehat{\boldsymbol{L}_e^+}\boldsymbol{e}, \tag{2.53}$$

Thus, the system is globally asymptotically stable if

$$\boldsymbol{L}_e\widehat{\boldsymbol{L}_e^+} > 0. \tag{2.54}$$

However, in practice, the number of image features used to construct $\boldsymbol{L}_e$ will be greater than the number of degrees of freedom (six), meaning $\boldsymbol{L}_e\widehat{\boldsymbol{L}_e^+}$ will have a nontrivial nullspace, since the rank of $\boldsymbol{L}_e$ is at most six. Consequently, it is possible to reach a configuration such that $\boldsymbol{e} \in \ker \widehat{\boldsymbol{L}_e^+}$, which would correspond to a local minimum. Therefore, only local asymptotic stability can be assured.

As [112] states, even though it can be shown that IBVS is locally asymptotically stable in a neighborhood around $\boldsymbol{e} = \boldsymbol{e}* = 0$, "determining the size of the neighborhood in which stability and the convergence are ensured is still an open issue, even if this neighborhood is surprisingly quite large in practice."

These issues of convergence are discussed in [109], as well as conditions where $\boldsymbol{L}_e$ becomes singular. The latter issue of a singular interaction matrix, introduced in Section 2.8.2.2 ('camera retreat'), occurs when rotations in vicinity of π radians around the optical axis of the camera are required. The method discussed in Section 2.8.2.2 nearly eliminates this issue, though it can become possible with this method that image features will leave the camera's FOV.

2.8.3 Advanced Methods of Visual Servo Control.

In their performance analysis of several visual servoing methods, Gans et al. summarizes some of the main pros and cons of the classical IBVS method succinctly:

> *IBVS requires no 3D reconstruction of the environment, but does require an image taken from the goal position. IBVS performs optimally in the image space and it is a simple matter to regulate the trajectory of image features, for instance preventing them from leaving the field of view. However, certain control tasks can lead to singularities in the image Jacobian, resulting in system failure. Image-based systems also surrender control of the Cartesian velocities. Thus, while the task error may be quickly reduced to zero, complicated and unnecessary motions may be performed. This is particularly troublesome when operating in a physically limited or hazardous environment. Finally, while IBVS is robust in the face of signal noise and calibration errors, it requires an estimate of the feature point depth, which may be unavailable or difficult to estimate accurately.* [115]

These issues have motivated proposals of many variations of the basic IBVS control law that each seek to overcome noted shortcomings of the classical method (e.g. [110, 116–119], though many more exist). This section reviews two of the more highly cited variations: 2-1/2D visual servoing (also refereed to as hybrid visual servoing) [110], and a partitioned visual servoing approach by Corke and Hutchinson [116]. These variations of IBVS are both noted for their methods to decouple translational and rotational control in an effort to improve performance over the basic method.

2.8.3.1 2-1/2D Visual Servoing.

The method of 2-1/2D visual servoing, introduced by Malis et al. in [110], represents one of the first variations of visual servoing to exploit a decoupling between rotational and translational motion. The authors propose using a calculated homography (see Section

2.4.2.2) between the current image frame and the goal image in order to estimate a partial relative pose between the camera's current pose and goal pose. This partial pose is then used to drive rotation of the camera, while translational motion is controlled in the image space in a similar manner to traditional IBVS.

For this method, the control error to be minimized is defined as

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{m}_e - \boldsymbol{m}_e^* \\ \theta \boldsymbol{u} \end{bmatrix}, \tag{2.55}$$

where θ and $\boldsymbol{u}$ are the rotation angle and axis between the current image frame and goal image, derived from the partial pose estimation method. $\boldsymbol{m}_e$ is defined as the extended image coordinates of a single point on the target ($\boldsymbol{m}_e^*$ represents the same point in the goal image)

$$\boldsymbol{m}_e = \begin{bmatrix} x \\ y \\ \log(Z) \end{bmatrix}, \tag{2.56}$$

where x and y are normalized image coordinates of the point, and Z is the depth to the point. Here the logarithm is useful so that the third component of $\boldsymbol{e}$ becomes $\log\left(\frac{Z}{Z^*}\right) = \log(\rho_Z)$, where the parameter ρ_Z can be estimated using the homography-based partial pose estimation method. Note that $\boldsymbol{m}_e$ is defined by a single feature, so this method is potentially more susceptible to measurement noise than the least-squares solution of traditional IBVS.

The control law is then defined in a similar manner as in (2.49) for IBVS

$$\boldsymbol{V}_{cmd} = -\lambda \hat{\boldsymbol{L}}^{-1} \boldsymbol{e}. \tag{2.57}$$

$\boldsymbol{L}$ is an upper triangular matrix that is singularity free in the entire task space [110]

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_v & \boldsymbol{L}_{(v,\omega)} \\ \boldsymbol{0} & \boldsymbol{L}_\omega \end{bmatrix}, \tag{2.58}$$

where $\boldsymbol{L}_\omega = \boldsymbol{I}_{3x3}$,

$$\boldsymbol{L}_v = \frac{1}{Z^* \rho_Z} \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{bmatrix}, \tag{2.59}$$

and

$$\boldsymbol{L}_{(v,\omega)} = \begin{bmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \\ -y & x & 0 \end{bmatrix}. \tag{2.60}$$

It can be shown as in [110] that the only value of depth that must be estimated is the distance from the target reference plane to the camera in the goal pose. In practice this can be judiciously estimated *a priori*. Though as [110] states, its influence on the stability of the system is minimal, so the estimate need not be very precise. This fact makes 2-1/2D visual servoing more-suited to use with a monocular camera than traditional IBVS, a potentially big advantage for space applications.

Other notable pros and cons of 2-1/2D visual servoing as compared to IBVS include: the Cartesian motion of a camera using this method is in general more ideal, the method is singularity free, and an estimate of depth is not strictly needed. However, as shown in [115], the method is more susceptible to measurement noise as it relies on the homography-based pose estimation method and only uses a single feature for the translational velocity command. Furthermore, the pose estimation step adds quite a bit of complexity to the algorithm, including greater computational requirements.

2.8.3.2 Corke and Hutchinson (C&H) Partitioned Visual Servoing.

Corke and Hutchinson propose a variant of visual servoing in [116], herein referred to as C&H visual servoing. The essence of this method is to decouple control of z-axis motion (rotation and translation) of the camera from control of x and y motion. This method was primarily designed to solve the issue of interaction matrix sigularity (camera retreat) issue associated with IBVS, though it has some other useful characteristics as well.

To develop this method, the relationship of (2.40) is partitioned to separate the z-axis components from other degrees of freedom

$$\dot{\boldsymbol{s}} = \boldsymbol{L}_{xy}\boldsymbol{V}_{xy} + \boldsymbol{L}_z\boldsymbol{V}_z, \tag{2.61}$$

where $\boldsymbol{V}_{xy} = [v_x, v_y, \omega_x, \omega_y]^T$,$\boldsymbol{V}_z = [v_z, \omega_z]^T$, and $\boldsymbol{L}_{xy}$ and $\boldsymbol{L}_z$ are the $\{1, 2, 4, 5\}$ and $\{3, 6\}$ columns of the traditional IBVS interaction matrix from (2.44), respectively. Applying a linear controller as was previously done for IBVS, the control for the x and y degrees of freedom is

$$\boldsymbol{V}_{xy}^{cmd} = -\widehat{\boldsymbol{L}_{xy}^{+}}\left(\lambda\boldsymbol{e}(t) + \boldsymbol{L}_z\boldsymbol{V}_z^{cmd}\right), \tag{2.62}$$

where $\boldsymbol{e}(t)$ and λ are defined in the same manner as for traditional IBVS.

The commanded translational and rotational velocity of the z axis, $\boldsymbol{V}_z^{cmd} = [v_z^{cmd}, \omega_z^{cmd}]^T$, is defined separately. For this, Corke and Hutchinson propose two new image features that are easily computed and can be used to determine each component of $\boldsymbol{V}_z^{cmd}$.

The first feature proposed is used to control the angular velocity of the camera around the optical axis, ω_z^{cmd}. It is defined as the angle between the u axis of the image plane and a line segment connecting two target feature points: $0 \leq \theta < 2\pi$. Using this, ω_z^{cmd} is calculated by

$$\omega_z^{cmd} = \gamma_{\omega_z}(\theta^* - \theta), \tag{2.63}$$

where γ_{ω_z} is a scalar gain coefficient.

The second feature is used to control the translational velocity of the camera along the optical axis, v_z^{cmd}. It is defined as the area of a polygon formed by several target feature points. This is a useful feature as it is a scalar with magnitude clearly related to the distance along the optical access from the target, it is easy to compute, and it is rotation invariant. Corke and Hutchinson define this feature as the square root of the area of this polygon

$$\sigma = \sqrt{area}. \tag{2.64}$$

Then v_z^{cmd} can simply be defined by

$$v_z^{cmd} = \gamma_{v_z}(\sigma^* - \sigma), \tag{2.65}$$

where γ_{v_z} is a scalar gain that controls the rate of approach to the target.

2.8.4 Spacecraft Applications of Visual Servoing.

A number of studies have examined visual servo control for space applications. Several consider IBVS for space-based robotic arm manipulators [120–123]. As these are concerned with robotic arms, they do not consider dynamics associated with relative spacecraft motion. Additionally, all assume known feature correspondences on the target spacecraft, so they therefore do not address the challenges inherent with unknown targets. A few studies have considered visual servoing for control of the spacecraft bus [66, 124–127]. The remainder of this section provides a review of each of these studies.

Yanagi and Terui propose an IBVS controller for maneuvering a spacecraft with respect to an asteroid in [124]. Dynamics of the maneuvering spacecraft are considered and the asteroid is considered fixed. The study assumes exactly three features points on the

asteroid are perfectly known and are always visible. Simulation results are presented for one approach scenario, and show asymptotic decrease in control error as the maneuvering spacecraft approaches its goal pose. While the results validate the basic IBVS control scheme, the assumptions and simplifications inherent in the work provide little insight regarding the practical utility of visual servoing for this application.

Petit et al. develop a model-based visual servo control scheme intended for spacecraft rendezvous missions in [66]. The authors propose using a 2-1/2D visual servo control loop in conjunction with a 3D model-based pose estimation scheme. Importantly, because this method relies heavily on a known geometric model of the target to conduct pose estimation, the study is not specifically applicable to the case of RPO with unknown targets. However, the method is demonstrated using a realistic experimental setup. The setup is composed of an accurate scale mock-up of the Amazonas-2 spacecraft placed in a dark room with a point-light source acting as the Sun. Additionally, an image of the Earth is apparent in the background, which must be dealt with by the pose-estimation and tracking scheme. Experimental results using a camera attached to a 6DOF robot arm demonstrate the capability of the method to track the relative pose and command a rendezvous maneuver with respect to the known target. While this work demonstrates the effectiveness of the 2-1/2D control law, it is not applicable for the case on an unknown target.

An IBVS system for spacecraft docking using a known fiducial marker located on the target is presented in [125]. A two-step visual servoing control scheme is implemented: the first step aligns the camera with the estimated centroid of the fiducial marker, and the second step controls the approach to the target using traditional IBVS. The feature points used by the IBVS controller are the corners of the bounding box associated with the fiducial marker. To estimate the location of this bounding box, a tracking scheme based on a particle filter augmented with the CAMShift algorithm [128] is used. The proposed controller is implemented in simulation using quadcopter dynamics, and experimentally using an

off-the-shelf quadcopter platform. While this work is noteworthy in that it demonstrates experimental results of an IBVS controller using a flying platform, it cannot be concluded that the scheme would be effective for spacecraft RPO with an unknown target. Specifically, the CAMShift tracking algorithm would likely fail to produce smooth estimation of the bounding box for a region of interest when used to track an unknown target in the dynamic lighting environment of space. Furthermore, because the corners of the bounding box are always available as feature points, feature point non-identification is not addressed.

A modified IBVS control scheme intended for use with tumbling targets is proposed in [126]. It is claimed that traditional visual servo control exhibits oscillating feature error when feature points are located on a rotating target. To combat this, the path of each feature point is approximated with an ellipse, and a control law is then derived using feature ellipses instead of the points themselves. Simulation results assuming perfect feature correspondence show good performance, though dynamics and feature modeling error are not considered. Section 3.4 will show that modeling image features as ellipses is not necessary to maneuver with respect to a tumbling target.

Pomares et al. examine a variant of traditional IBVS for control of far and close range spacecraft RPO in [127, 129]. An IBVS control law is designed that is invariant to camera focal length, meaning it can be used with a zooming camera. This provides an expanded range of utility, as a single zoom-able camera sensor can be used for far and close approach phases of RPO. Actuators (thrusters and reaction wheels) are modeled, and simulations show that fuel use is commensurate with expected values for actual rendezvous maneuvers. Simulation results for three scenarios are presented, demonstrating the ability of the controller to maneuver a chaser during far approach, close approach, and with respect to a tumbling target. The results of this study show that visual servoing is a promising control scheme for spacecraft RPO. However, even though the study claims to be applicable to RPO with unknown targets, there is no discussion of how features are extracted, how

the goal image is generated, and it does not examine the robustness of the controller to modeling error (target features are are assumed to perfectly matched and observed in every frame). Additionally, there is no discussion of how depth to the target is estimated, or how any of these error sources could affect performance.

2.9 Summary

This chapter provided an overview of relevant background material as well as a review of related research. First, a number of historical on-orbit missions were discussed, and it was explained that these missions have not demonstrated solutions to the problem statement that this work seeks to address. Throughout the chapter, concepts from the fields of astrodynamics, computer vision, estimation theory, visual navigation, and robotics were covered in order to provide context and background for the methods developed in this work. Spacecraft dynamics were summarized, including relative translational dynamics and rigid body attitude dynamics. Several topics from the field of computer vision were introduced, including a summary of a feature-based image processing pipeline that will be used in this work. A review of literature relating to the use of computer vision methods in space was provided, and it was concluded that while it appears traditional computer vision techniques will work in space, work is yet to be done to conclusively demonstrate this. The last research question of this research effort will seek to investigate the potential effectiveness of feature-based computer vision techniques in the space environment.

The problem of relative pose determination for unknown targets was examined through a review of research applicable to this topic. It was found that while much research has considered this problem, no robust solution has been demonstrated in available literature. As a result, traditional control methods will not be effective for the case of an unknown target, since they all rely on an accurate relative pose estimate. This fact motivates the examination of visual servo control methods in this work, which do not require a full relative pose estimate.

The field of visual servo control was introduced, including a development of control laws for several common types of image-based visual servo methods. Some noted pros and cons of these methods were discussed.

Finally, this chapter reviewed a number of works that have examined visual servoing for spacecraft applications. While these works generally demonstrate the effectiveness of IBVS-style control laws for commanding close-in RPO maneuvers, none of them address the fundamental issues associated with unknown targets. Specifically, issues regarding generation of a goal image, detection and tracking of natural features on a target spacecraft, and robustness to measurement errors associated with the visual navigation system have not been addressed in available literature. As such, the present study seeks to more completely demonstrate the potential of visual servo control for spacecraft RPO and extend its utility to include use with unknown targets.

III. GNC Method 1: Image Based Visual Servoing for Final Approach Maneuvers with Unknown Targets

3.1 Overview

This chapter examines Research Question 1 (see Section 1.2): a complete visual navigation and control architecture for conducting final approach maneuvers with unknown targets is proposed and demonstrated in simulation. Here notional two-spacecraft final approach maneuvers are considered, as introduced in Section 1.3. The assumptions and limitations discussed in Section 1.4 apply for all scenarios presented.

Broadly, the proposed architecture utilizes a stereo camera sensor mounted on the chaser vehicle as its primary relative navigation sensor, combined with a feature-based image-processing method discussed in Chapter II. An Image-Based Visual Servoing (IBVS) controller is used to generate velocity commands for the chaser vehicle to achieve a desired goal pose. Issues unique to maneuvering with respect to unknown targets are addressed, such as estimating the relative velocity between chaser and target, and generating a goal image prior to conducting the maneuver. The architecture is analyzed in simulation, with a focus on assessing its robustness to measurement error resulting from the computer vision methods used to process video image frames from the camera sensor.

This chapter begins with a detailed description of the architecture, including how each of its subsystems function. Next, the simulation developed to test the architecture is introduced, and the methodology for modeling computer vision related sources of error is discussed. Section 3.4 presents simulation results and analysis for a number of final approach scenarios that seek to show the efficacy of the method under a range of realistic conditions. Finally, the chapter concludes with a summary of pros and cons of the architecture, and motivates Research Question 2 (to be examined in Chapter IV).

3.2 Method

This section outlines the proposed visual navigation and control architecture for controlling six degree of freedom (6DOF) maneuvers with respect to unknown targets. The architecture is designed around a traditional IBVS control law, and is augmented with other functions to enable implementation with unknown targets. Figure 3.1 shows the basic structure of the architecture, and divides the whole system into three subsystems: sensing, navigation, and control. The next subsection discusses necessary preconditions for conducting maneuvers using this architecture, then the remainder of this section presents details for each subsystem of the architecture.

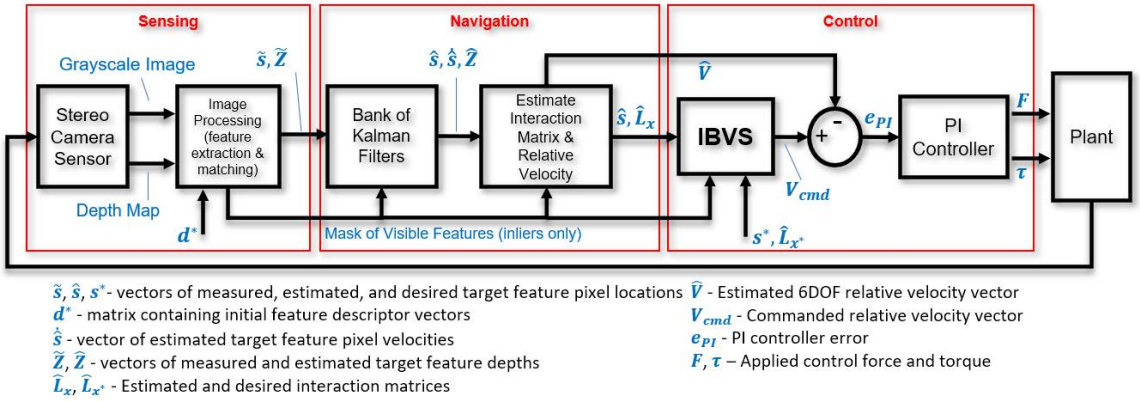

Figure 3.1: Proposed Visual Navigation and Control Architecture

3.2.1 Prior to Maneuvering.

With respect to the taxonomy of a final approach maneuver shown in Figure 1.3, this architecture represents the 'Method 1' block. That is, it is capable of controlling maneuvers assuming a goal image of the target is available, and a sufficient number of features on the target can be tracked.

As the taxonomy in Figure 1.3 shows, prior to maneuvering it is assumed that an inspection phase has been conducted and a goal image for the target has been generated (Section 3.3 will discuss how a goal image can be generated for unknown targets). This

phase would likely be composed of a number of potential relative maneuvers intended to ascertain a region of interest (ROI) on the target spacecraft (conducted at a safe range from the target).

A major advantage of the architecture proposed in this chapter is that a relative pose estimate is not required, so it is important to emphasize that in the inspection phase it is not necessary to gather significant information about the target, such as geometric shape, moments of inertia, relative pose, etc. Rather, the inspection phase is intended to allow a human operator to determine how they would like to maneuver the chaser spacecraft with respect to the target. Because the target is unknown, a human-in-the-loop is required prior to the visual servo maneuver to determine the goal of the maneuver (this is not a limitation specific to this architecture: it is the opinion of the author that a human operator will be required in some capacity for any architecture designed for final approach with unknown targets). In practice, this would be achieved by coarsely maneuvering the chaser to a location where a ROI on the target is visible. This ROI is then used to generate a goal image for the autonomous visual servo maneuver, which implicitly defines the goal relative pose of the chaser to target.

Conducting inspection phase maneuvers is not a primary focus of this research effort, however it is worth noting that natural motion circumnavigation (NMC) maneuvers could be used to view the target from a variety of relative poses in order to position the chaser such that a goal image for the visual servo maneuver could be generated. NMC's exploit the relative astrodynamics of two objects in orbit operating in vicinity of one another. It is possible to design a bounded relative 'orbit,' where upon the chaser circumnavigates the target (here assumed to be at the origin of the Local-Vertical-Local-Horizontal (LVLH) frame) as the target orbits the Earth. In this way, after the chaser is given an initial velocity to place it in an NMC, translational motion with respect to the LVLH origin is predictable and 'free' (in terms of fuel use). More information on NMC's can be found in [22], and

on design of inspection maneuvers in [130]. Figure 3.2 displays an example 3D trajectory for a chaser in an NMC as the target conducts one orbit of the Earth. As can be seen, the relative path traced by the chaser with respect to the LVLH origin is a bounded 2x1 ellipse, thus enabling a full 360 degree view of the target

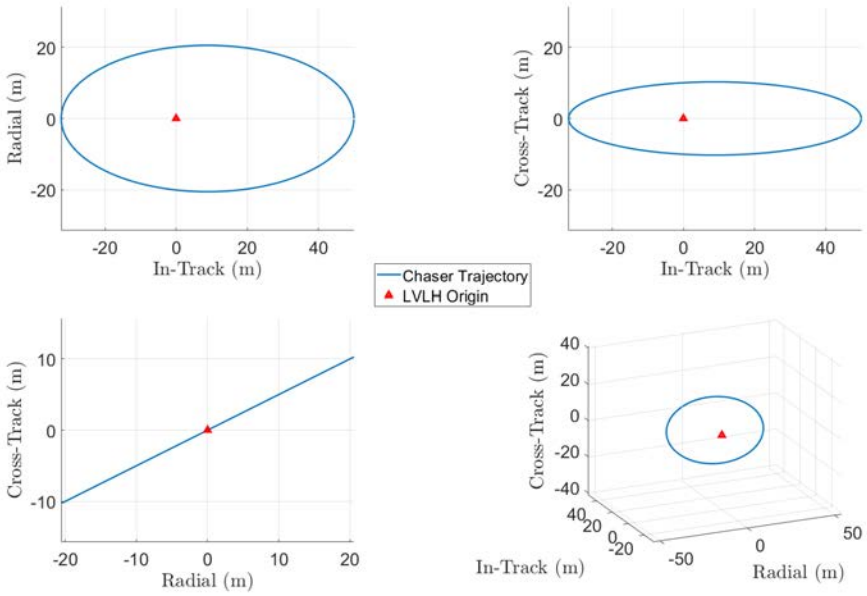

Figure 3.2: Example NMC for pre-manuever inspection of target

3.2.2 Sensing Subsystem.

The sensing subsystem is composed of a stereo camera sensor attached to the chaser spacecraft, and an image processing pipeline to process data from the camera. The stereo camera sensor is assumed to provide a grayscale image (i.e. a single image from one of the two cameras in the sensor) and a rectified depth map at frame rates capable for real-time use. The goal of the sensing subsystem is to identify and track natural feature points within the target ROI, and provide an image plane measurement (horizontal and vertical pixel location) and depth for each tracked feature point.

For this chapter, it is assumed the image processing pipeline operates essentially as described in Section 2.4.3.3 (see Figure 2.7). This pipeline identifies features in the current image frame (detects point features and generates descriptors for each feature), then uses a matching algorithm to compare the feature descriptors of the current image frame to those generated *a priori* for the goal image (denoted $\boldsymbol{d}^*$ in Figure 3.1). A robust outlier detection method is then used to remove most of the outliers from this set of matched features.

The subset of features identified in the current frame ('visible' features, denoted $\tilde{\boldsymbol{s}}$), i.e. those features that can be matched to features in the goal image, is an output of the sensing subsystem. These features represent unique points on the target that are tracked from frame to frame and are used by the navigation and control subsystems.

Importantly, a potentially large portion of features identified in the goal image may not be matched in the current image frame (for a variety of reasons, see Section A.1.2.2). As such, $\tilde{\boldsymbol{s}}$ represents a subset of all of the features $\boldsymbol{s}$ that are tracked by the navigation filters. To address the problem of bookkeeping, the sensing subsystem also provides a mask of visible features. This mask is a binary vector of length equal to the total number of features tracked, with each component having a value of 1 if the corresponding feature was matched in the current frame, else 0.

The final output of the sensing subsystem is a vector containing the estimated depth to each visible feature, $\tilde{\boldsymbol{Z}}$. Pixel locations of each visible feature are used in conjugation with the depth map to compute $\tilde{\boldsymbol{Z}}$. In reality there are parts of stereo depth maps that may not provide a value of depth (e.g. if a portion of the image is not seen by both cameras). In practice, values of depth for these regions may be interpolated, or features may be deemed not visible for the current frame if they do not have a valid value of depth.

Section 4.2.3 provides a more in-depth discussion of a practical implementation of the sensing subsystem for use with actual camera hardware and computer vision methods.

3.2.3 Navigation Subsystem.

The navigation subsystem has three functions: produce a filtered estimate of feature pixel locations ($\hat{\boldsymbol{s}} = [\hat{u}_1,\ \hat{v}_1, \ldots \hat{u}_N,\ \hat{v}_N]^T$), build the interaction matrix ($\widehat{\boldsymbol{L}_x}$), and estimate the relative velocity between chaser and target ($\widehat{\boldsymbol{V}}$).

A bank of linear Kalman filters is used to refine the measurements $\tilde{\boldsymbol{s}}$ and $\tilde{\boldsymbol{Z}}$. At the start of the maneuver, one Kalman filter is initiated for every feature in the goal image. The feature dynamics model does not incorporate any coupling between different features' states (i.e. each feature is considered independent), therefore if a single filter were to be used, there would be no cross-covariance generated between different features (and hence no benefit).

A constant velocity model is used to describe the dynamics of feature pixel locations and depths using the state vector for a single feature $\boldsymbol{x}_i = [u_i,\ v_i,\ Z_i,\ \dot{u}_i,\ \dot{v}_i,\ \dot{Z}_i]^T$, the discrete state transition matrix for this model is

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where dt is the filter time step. A Kalman filter update step is initiated for each visible feature according to the mask provided by the sensing subsystem. As elements of the state vector are directly measured ($\boldsymbol{z}_i = [u_i, v_i, Z_i]^T$), the measurement matrix is simply

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{3.2}$$

To deal with the possibility that the outlier rejection performed in the sensing subsystem fails to remove a false match, additional outlier rejection is implemented in each Kalman filter via monitoring of the Mahalanobis distance of each measurement (as known as chi-squared gating). The *i*th feature's Mahalonbis distance for the *k*th timestep is defined as

$$d_{i_k}^2 = (\boldsymbol{z}_{i_k} - \hat{\boldsymbol{z}}_{i_k})^T \boldsymbol{S}_{i_k}^{-1} (\boldsymbol{z}_{i_k} - \hat{\boldsymbol{z}}_{i_k}), \tag{3.3}$$

where $\boldsymbol{z}_{i_k}$ is the measurement vector for the *i*th feature at the *k*th time step, $\hat{\boldsymbol{z}}_{\boldsymbol{i_k}}$ is the filter predicted measurement, and $\boldsymbol{S}_{i_k} = \boldsymbol{H}\boldsymbol{P}_{i_k}\boldsymbol{H}^T + \boldsymbol{R}$ is the measurement prediction covariance. If the Mahalanobis distance for a measurement exceeds a predefined threshold, then the Kalman filter update step for that measurement is not performed. In practice this threshold can be chosen using a χ^2 distribution and desired confidence level, as in [54]. Furthermore, the mask of visible features is updated to remove this outlier, which ensures it is not used in calculation of the interaction matrix and IBVS control command during the current time step.

The mask of visible features is used to produce trimmed vectors $\boldsymbol{s}_t$, $\dot{\boldsymbol{s}}_t$, and $\boldsymbol{Z}_t$, which contain only inliers for the current time step. An interaction matrix, $\widehat{\boldsymbol{L}}_x$, is constructed using these trimmed vectors, the camera intrinsic parameters, and relationships introduced in Section 2.8.2.1

$$\begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \vdots \\ \dot{x}_n \\ \dot{y}_n \end{bmatrix} = \underbrace{\begin{bmatrix} \widehat{\boldsymbol{L}}_{x_1}(\bar{\boldsymbol{s}}_1, Z_1) \\ \widehat{\boldsymbol{L}}_{x_2}(\bar{\boldsymbol{s}}_2, Z_2) \\ \vdots \\ \widehat{\boldsymbol{L}}_{x_n}(\bar{\boldsymbol{s}}_n, Z_n) \end{bmatrix}}_{\widehat{\boldsymbol{L}}_x} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \tag{3.4}$$

where the interaction matrix for the *i*th feature is

$$\widehat{\boldsymbol{L}}_{x_i} = \begin{bmatrix} \frac{-1}{Z_i} & 0 & \frac{x_i}{Z_i} & x_i y_i & -(1+x_i^2) & y_i \\ 0 & \frac{-1}{Z_i} & \frac{y_i}{Z_i} & (1+y_i^2) & -x_i y_i & -x_i \end{bmatrix}, \tag{3.5}$$

and normalized image feature coordinates ($\bar{\boldsymbol{s}}_i = [x_i, y_i]^T$) are computed using the pixel coordinates and

$$\begin{aligned} x_i &= \frac{u_i - c_u}{f_u}, \quad \dot{x}_i = \frac{\dot{u}_i}{f_u} \\ y_i &= \frac{v_i - c_v}{f_v}, \quad \dot{y}_i = \frac{\dot{v}_i}{f_v} \end{aligned}. \tag{3.6}$$

The estimate of relative velocity between camera and target is calculated using a pseudo-inverse of the interaction matrix relationship discussed in Chapter II (2.46):

$$\widehat{\boldsymbol{V}} = \widehat{\boldsymbol{L}}_x^{+} \dot{\bar{\boldsymbol{s}}}_t. \tag{3.7}$$

This velocity estimate represents the velocity of the chaser with respect to the target, expressed in the chaser camera frame. As it is calculated in the least-squares sense, robustness to image measurement noise improves as the number of features increases.

3.2.4 Control Subsystem.

The control subsystem is a 'dynamic look-and-move' system utilizing traditional IBVS. It is composed of two components: an IBVS controller to generate relative velocity command setpoints and a proportional-integral (PI) controller to generate acceleration commands for the spacecraft actuators. Low-level logic for control of actuators is not considered in this architecture, as this is something specific to particular spacecraft.

The relative velocity command given by the IBVS control law is

$$\boldsymbol{V}_{cmd} = -\boldsymbol{\lambda}\widehat{\boldsymbol{L}_e^+}\boldsymbol{e}_{IBVS}, \tag{3.8}$$

where $\boldsymbol{\lambda}$ is a gain matrix, $\widehat{\boldsymbol{L}_e^+}$ is the pseudo-inverse of the interaction matrix approximation, and $\boldsymbol{e}$ is the feature servo error defined by

$$\boldsymbol{e}_{IBVS} = \bar{\boldsymbol{s}}_t - \bar{\boldsymbol{s}}_t^*, \tag{3.9}$$

where $\bar{\boldsymbol{s}}_t$ and $\bar{\boldsymbol{s}}_t^*$ are the trimmed feature vectors for the current frame and goal image frame, respectively (expressed in normalized coordinates).

Note that the vector containing the locations of all features in the goal image, $\boldsymbol{s}^*$, and the interaction matrix for the goal image, $\widehat{\boldsymbol{L}_{x^*}}$, can be calculated prior to the maneuver when the goal image is generated. These parameters are then trimmed every time step using the mask of visible features in the same manner as described previously.

$\boldsymbol{\lambda}$ is implemented as a gain matrix instead of a scalar to affect each component of velocity differently. Specifically, it was found that it is useful to set the gain for translational velocity commands differently from the gain for angular velocity commands, in order to shape the trajectory of feature points in the image plane such that they stay in the camera's field of view for the duration of the maneuver.

The approximation of the interaction matrix, $\widehat{\boldsymbol{L}_e^+}$, is selected in accordance with [111]

$$\widehat{\boldsymbol{L}_e} = \frac{1}{2}(\widehat{\boldsymbol{L}_x} + \widehat{\boldsymbol{L}_{x^*}}), \tag{3.10}$$

where $\widehat{\boldsymbol{L}_x}$ is provided by the navigation subsystem and $\widehat{\boldsymbol{L}_{x^*}}$ is an estimate for the interaction matrix associated with the goal pose (calculated *a priori*). Note that $\widehat{\boldsymbol{L}_{x^*}}$ requires an estimate of depth when the camera is in the goal pose. In practice, it was found that this estimate can be very coarse. The approximation is selected in part to minimize the potential for camera retreat (see Section 2.8.2.2), but also because it was observed in testing that this approximation provides a more ideal Cartesian trajectory than the other two approximations discussed in Section 2.8.2.2.

It will be shown in the results section of this chapter that this architecture performs well even when the target is moving. For a reader familiar with IBVS control as typically developed in literature, this might seem counter-intuitive, as it is often stated that the basic control law applies for non-moving targets (as in [107]). Indeed, several works propose adding an additional term to the control law for moving targets [108, 131]

$$\boldsymbol{V}_{cmd} = -\lambda\widehat{\boldsymbol{L}_e^+}\boldsymbol{e} - \widehat{\boldsymbol{L}_e^+}\widehat{\frac{\delta \boldsymbol{e}}{\delta t}}, \tag{3.11}$$

However, it is assumed in literature that the reference frame the velocity command is generated with respect to is a stationary frame, not attached to the target. Therefore, if the target is moving, it is necessary to account for the time variation of $\boldsymbol{e}$. This makes sense for typical applications of visual servoing. Considering a robotic arm, for example: sensors can measure joint velocities, which can be used to estimate the velocity of the camera with respect to some fixed reference point.

Because the relative velocity in this architecture is estimated with respect to the target (3.7), the reference frame for velocity commands is always static with respect to the target, so it is not necessary to account for the time variation of $\boldsymbol{e}$ as in (3.11).

Finally, a PI controller is implemented to produce acceleration commands for the chaser vehicle. This controller uses the error term

$$\boldsymbol{e}_{PI} = \boldsymbol{V}_{cmd} - \widehat{\boldsymbol{V}}, \tag{3.12}$$

where $\widehat{\boldsymbol{V}}$ is the velocity estimate provided by the navigation subsystem. A derivative term is not implemented with this controller as it was observed in testing that the error term tends to be quite noisy when sensor measurement noise is considered, which makes accurate calculation of the error derivative difficult. The output of the PI controller is a translational acceleration command and angular acceleration command, which is used by the low-level actuator logic to determine actuator commands.

3.3 Generating the Goal Image for Unknown Targets

A fundamental requirement of this architecture is that a goal image of the target be available, which implicitly defines the desired relative pose between the camera and target. Features in this goal image must be able to be matched to features in the current image frame in order to calculate the feature error for the IBVS control law. For a known target, a goal image could be selected from an independently generated database of images representing possible desired poses. Obtaining a goal image for an unknown target, however, represents a more challenging task that has not been addressed in available literature.

Unless the chaser has previously been positioned in the goal pose and was able to capture an image, there is no way to obtain an exact image representing the desired pose for a previously unknown target. This section proposes two options to generate an approximate goal image.

3.3.1 Option 1: Using a Zoomed-in Image.

The first option requires variable focal length (zooming) camera on the chaser. Prior to active maneuvering, a zoomed-in image of the target is captured at an instant when the optical axis of the camera is aligned with a vector orthogonal to goal image plane. This zoomed-in image then represents the goal image (note that this image can be rotated/shifted as necessary to properly represent the desired pose). Ostensibly this is a stringent requirement: the chaser must be partially aligned with the goal pose such that only a translation along and a rotation around the optical axis is required. This may not be unreasonable for many RPO scenarios though, as long as the target can be observed for a period of time prior to initiating the visual servo maneuver. During the observation time, judiciously designed natural-motion circumnavigation relative orbits could be used to observe the target from various angles as discussed in Section 3.2.1, including one sufficiently aligned with the desired goal pose.

In order to determine the amount of zoom required to achieve a desired (goal) depth at completion of the maneuver, the following relationship can be used (derived from (2.27))

$$f_{zoom} = \frac{(Z_{current})(f_{maneuver})}{Z_{goal}}, \tag{3.13}$$

where f_{zoom} is the required focal length for the zoomed image, $Z_{current}$ is the current depth from the camera to the target, $f_{maneuver}$ is the focal length of the camera used for maneuvering, and Z_{goal} is the desired depth at completion of the maneuver.

To demonstrate this option, two photos were taken of a satellite mock-up using the same camera at different zoom levels (1x and 2x zoom), shown in Figure 3.3 (a) and (b). A feature-based image processing pipeline was used to extract and match features between both images, as shown in Figure 3.3 (c). As can be seen, a large number of matches can be found between the scenes, even though they were taken using different focal lengths.

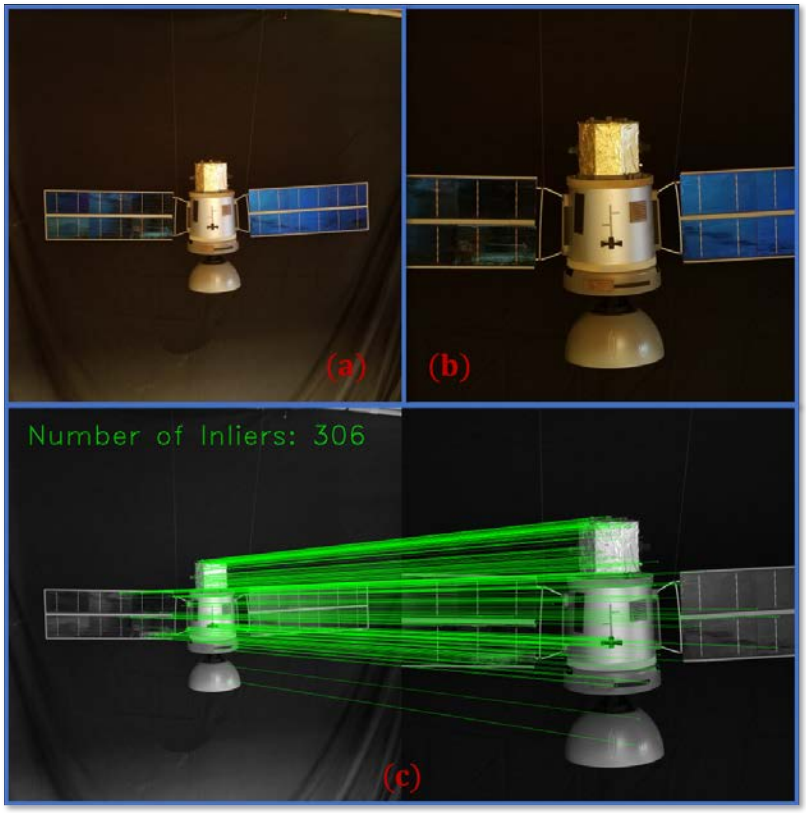

Figure 3.3: Goal image generation using zooming camera. (a) 1x zoom representing current view of target (b) 2x zoom representing goal image (c) matches between frames using SIFT detector/descriptor with RANSAC

It is important to note that the generated goal positions of features will not be perfect, due to the fact that a longer focal length image is used to generate them than is used for maneuvering. This is in general not a significant problem, as the IBVS control law computes control commands using a least-squares solution, meaning if enough features are tracked, the desired pose should still be achieved.

3.3.2 Option 2: Using a Perspective Warp.

The second option utilizes the concept that two images of a planar target taken from different perspectives are related by an isomorphic transformation called a homography (see Section 2.4.2.2). Using a homography, an image of a planar surface can be warped to create an approximate image of what the surface would look like from a different

perspective. Given four manually selected corners of a planar quadrangle present in a current image and the desired locations of these four corners in the goal image, a homography can be calculated that is then used to warp the current frame into a view approximating the desired perspective. This warped view is used as the goal image of the target.

To demonstrate the second option, a photo of a satellite mock-up was taken representing the view from the current pose of a notional chaser. A goal image was generated using OpenCV [37] functions *getPerspectiveTransform()*, which estimates the homography matrix from the current image to the desired image, and *warpPerspective()*, which produces a warped image using the estimated homography (shown in Fig. 3.4(b)). Fig. 3.4(c) shows that a large number of matches can be found between the original and warped views of the target. Note that the surface of the target is not completely planar, though relative to the distance the photo was taken from, it is planar enough to produce a reasonable result.

To implement the second option in practice, the entire current image does not need to be warped into the goal image, only the feature locations do. The position of the *i*th feature in the goal image can be generated using the estimated homography

$$\begin{bmatrix} \boldsymbol{s}_i^* \\ 1 \end{bmatrix} = \widehat{\boldsymbol{H}} \begin{bmatrix} \boldsymbol{s}_i \\ 1 \end{bmatrix}. \tag{3.14}$$

A third option could certainly be to combine these two options, as long as the limitations of each method are observed. It is important to note that for any method, the feature locations in the generated goal image will not precisely match their locations when the camera is actually in the goal pose. Given a sufficient number of features, this is in general not a problem however, as the IBVS control law seeks to minimize the error between current feature locations and goal locations using a least-squares solution.

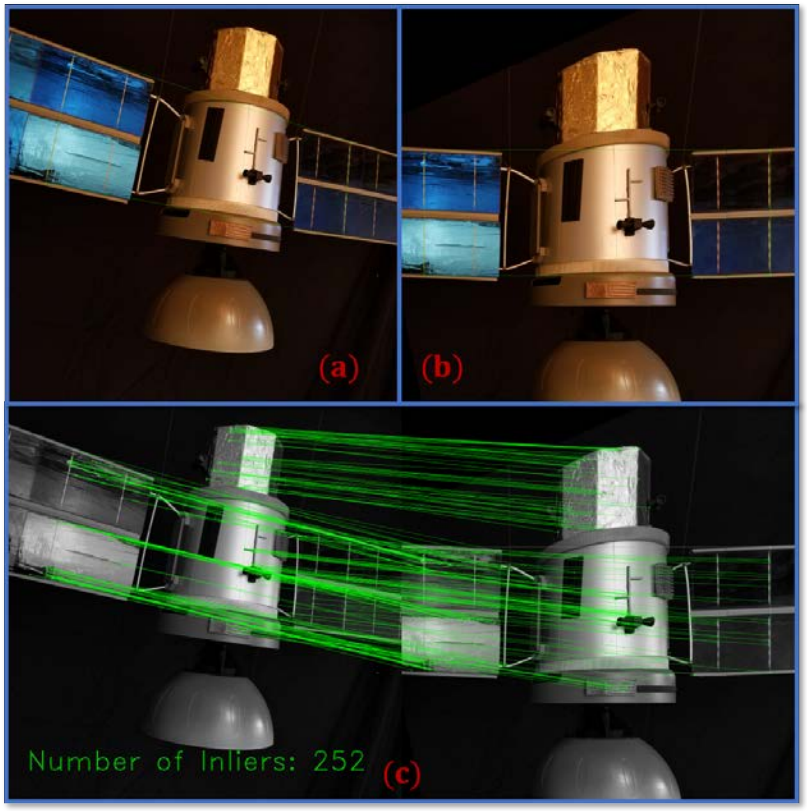

Figure 3.4: Goal image generation using perspective warp. (a) current view of target (b) warped view of target representing goal image (c) matches between frames using SIFT detector/descriptor with RANSAC

3.4 Results

This section presents simulation results and analysis for a range of final approach maneuvers using the architecture proposed in this chapter. Several increasingly complex scenarios are presented in order to examine how the architecture handles various possible initial conditions. The simulation utilized for these results is discussed in Appendix A. As the specific parameters of the spacecraft and scenarios are notional, the intent of this section is not to prove that the system works for a particular case, but instead to show that it performs well under a variety of conditions. The goal of the maneuver for each scenario is to approach the selected target ROI such that it takes up most of the camera's field of view, with the camera plane parallel to the ROI (for example, see Figure 3.6(b) for a camera view from goal pose).

The maneuver scenarios presented are broadly split into two subgroups: those where the target ROI associated with the goal pose is approximately a planar surface, and those where it is not. The results are divided in this way in order to examine how the surface structure of a target affects the effectiveness of the architecture. Specifically, because the option to generate a goal image via perspective warp relies on the assumption that the region being warped is planar, it is important to examine how warping a nonplanar surface can affect the quality of the maneuver.

The next subsection describes how measurement error parameters were selected for the camera sensor, and the following sections present results for targets with planar surfaces and targets with nonplanar surfaces, respectively.

3.4.1 Camera Sensor Modeling Parameters.

Table 3.1 summarizes the values used for each computer vision error parameter as well the total number of features tracked. According to [132], a standard deviation of 1 pixel for measurement location error is appropriate for most feature-tracking schemes. In order to examine worst-case conditions, the standard deviation of pixel location error was

chosen to be 2 pixels for this study. Experimental results in [133] yield a stereo disparity error standard deviation of 0.5, though a value of 1.0 is chosen here to examine robustness to greater depth estimation error. The percent of outliers present in any given image frame is dependent on a number of factors, but is typically very low if a scheme like RANSAC is implemented in the image processing pipeline. An outlier percentage (after RANSAC) of 3% is considered to be greater than would be expected in realty, but is selected to demonstrate the Kalman filter's ability to reject measurement outliers. The percentage of features not identified in any given frame is highly variable depending on scene conditions and image processing. A value of 50% is thought to be relatively conservative based on hardware experimentation with the satellite mock-up displayed in Figure 3.4.

Finally, for all scenarios presented except scenario 3P5, 100 target features are initiated and tracked by the bank of Kalman filters (of which 50% are selected to not be identified in each frame). This number was chosen based on hardware testing with the satellite mock-up shown in Figure 3.4, using a variety of feature detectors/descriptors, and is considered to be fairly conservative. Scenario 3P5 examines how the system performs for a reduced number of target features, and tracks only 20 target feature points total.

Table 3.1: Computer Vision Modeling Parameters

Error Mechanism	**Value**	**Units**	**Description**
Location Error	2.0	Pixels	σ of AWG applied to pixel location of image features
ϵ_d	1.0	Pixels	Stereo disparity matching error
Outliers	3	%	Percentage of features recast as outliers for each image frame
Non-IDs	50	%	Percentage of features not identified in each image frame
Target Features	100*	n/a	Total number of target point features tracked

* - For scenario 3P5 only, 20 target features are tracked

3.4.2 Maneuvering with Respect to a Planar (P) ROI.

This subsection considers maneuvers with respect to regions of interest on the target that are approximately planar. Figure 3.5 shows a Computer Aided Design (CAD) view of the target, with the ROI selected for maneuvering highlighted in red. The range of depth for protrusions and cavities within the ROI is 2 centimeters, which is considered planar in this context.

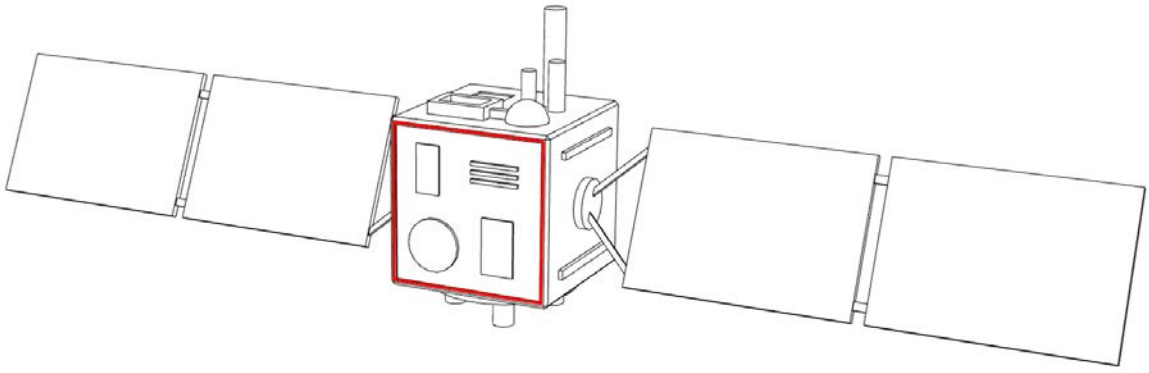

Figure 3.5: CAD model of target used in simulation, with approximately planar ROI highlighted in red

Table 3.2 contains a summary of the scenarios (3P1-3P5) presented in this subsection. The first two maneuvers are relatively basic, requiring only a corkscrew motion along the camera's optical axis to achieve the goal pose. Scenario 3P1 is presented without sensor noise applied, then scenario 3P2 applies sensor noise to show how it affects performance. The next three maneuvers, 3P3-3P5, consider a more complex maneuver (with sensor noise applied) requiring relative motion in all 6DOF. These scenarios all utilize the second method for goal image generation (perspective warp) proposed in Section 3.3. They variously examine a static (with respect to the LVLH frame) target, a moving/rotating target, and a target with significantly fewer feature points available for tracking.

Table 3.2: Overview of Maneuver Scenarios with Respect to Planar Target ROI

Scenario	Goal Image Generation	Goal Image Generation	Sensor Noise Modeling
3P1	Simple maneuver	Method 1 (zoom)	No
3P2	Simple maneuver	Method 1 (zoom)	Yes
3P3	Complex maneuver with static target	Method 2 (warp)	Yes
3P4	Complex maneuver with tumbling/drifting target	Method 2 (warp)	Yes
3P5	Complex maneuver with static target & reduced target features*	Method 2 (warp)	Yes

*- Only 20 target features are tracked for this scenario (100 for others)

3.4.2.1 Scenario 3P1: Simple Maneuver without Sensor Error Modeling.

The first scenario considered entails a relatively basic maneuver, without any sensor measurement error applied. The intent of this scenario is to present how the architecture performs under noise-free conditions.

For the maneuver, the chaser spacecraft is initially positioned on the positive in-track axis of the LVLH frame 10 meters away from the target, such that a translation directly along and a 90° rotation around the camera's optical axis is required to achieve the desired goal pose. The target is static at the LVLH origin with its planar region of interest facing in the direction of the positive in-track axis. Figure 3.6(a) shows the initial view of the target prior to maneuvering, and Figure 3.6(b) shows the view of the target from the chaser after the maneuver is complete.

For scenario 3P1 and 3P2, the goal image for maneuvering is generated using the first method described in Section 3.3 (i.e. using a zoomed-in view of the target), and implemented as discussed in Section A.1.3. A zoomed-in image is generated from the initial pose of the chaser using a 66 millimeter focal length camera image of the target (i.e. 6.6x zoom relative to the maneuvering camera). This focal length is chosen using (3.13)

and a desired goal depth of 1.5 meters. The image taken by this higher zoom camera is rotated 90° counter-clockwise to account for the desired rotation of the chaser relative to the target. This image is then used as the goal image for maneuvering. Given that the target bus is cube with 1 meter long sides and its geometric center is at the LVLH origin, the goal pose of the chaser equates to a position of $[0, 2, 0]$ meters in the LVLH frame for these scenarios.

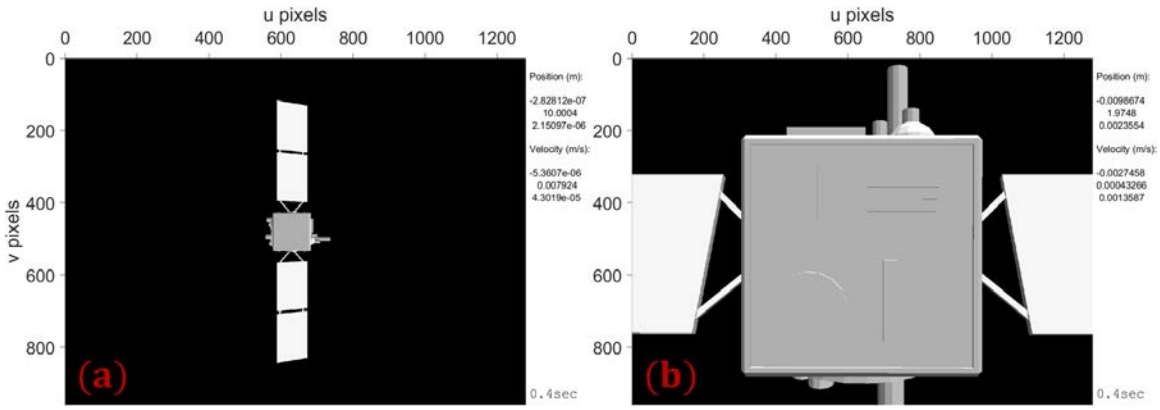

Figure 3.6: Scenario 3P1: camera views from chaser. (a) Initial view. (b) Final view.

Figure 3.7 shows the position of the chaser over the course of the 200 second maneuver. Maneuvering commences 5 seconds after simulation start to allow the filters to converge (this is the case for all results in this chapter). As can be seen, the chaser smoothly and asymptotically approaches the goal position of 2 meters from the LVLH origin in the in-track direction. The asymptotic approach of the goal pose is also apparent in the time history of translational and angular velocity (Figure 3.8). This is a very desirable characteristic of this architecture for space applications: collision avoidance is a primary concern for rendezvous and proximity operations (RPO), so it behooves a GNC system to increasingly slow the relative velocity as the relative distance decreases.

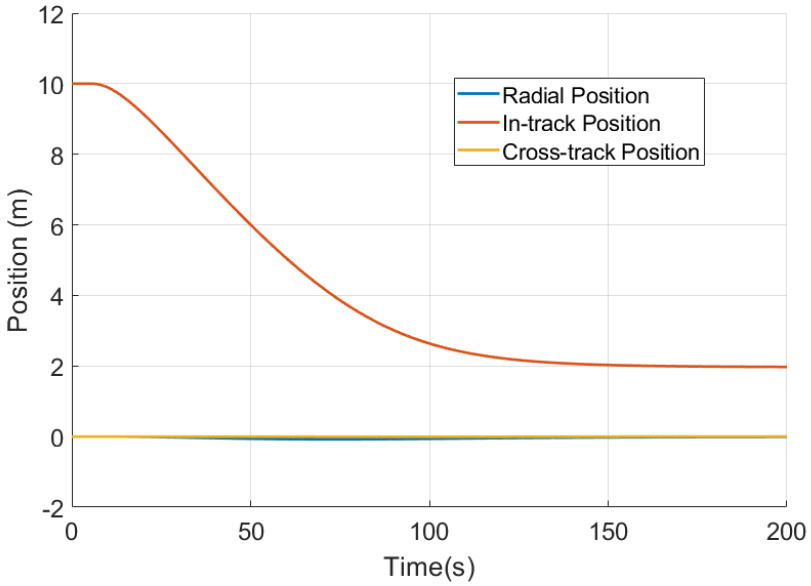

Figure 3.7: Scenario 3P1: position of chaser with respect to LVLH frame.

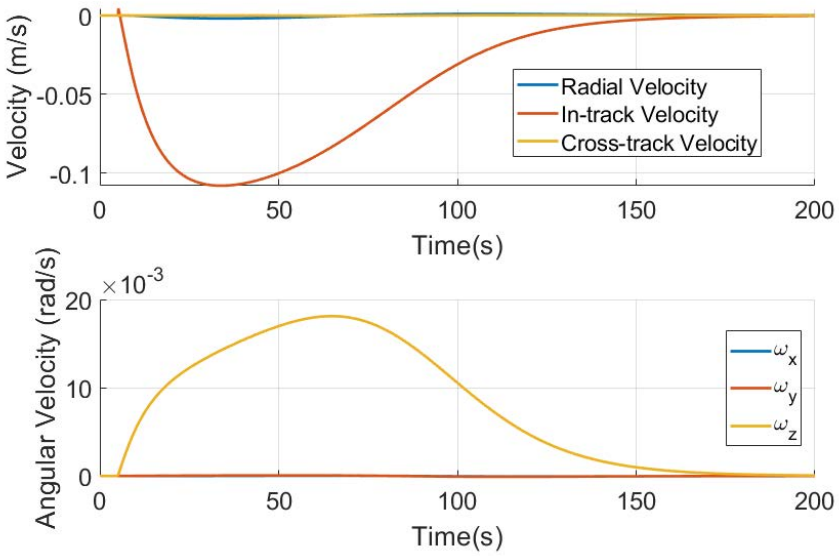

Figure 3.8: Scenario 3P1: velocity of chaser with respect to LVLH frame.

The chaser deviates very slightly (less than 0.07 meters) from a straight-line path along the in-track axis in the first 50 seconds of the maneuver. This deviation occurs in the negative radial direction, and is a product of the relative astrodynamics underpinning the maneuver. Indeed, this deviation complies with physical intuition for relative maneuvers in an LVLH frame: if a chaser located on the positive in-track axis is given a negative in-track velocity (as is the case in this scenario), its position will increase in the negative radial direction in addition to the negative in-track direction. The deviation is corrected to within a centimeter by 180 seconds into the maneuver.

To gain an understanding of how the target feature points behave in the image plane, Figure 3.9(a) displays the image plane trajectories for 10 target feature points. The feature trajectories follow a smooth curving trajectory to achieve their goal positions in the image plane. The IBVS control law gains for translational velocity and angular velocity components were differentially tuned to produce this desirable performance. The gain for the angular velocity components was selected to be slightly higher than that of the translational velocity components in an effort to ensure more of the necessary rotation of the chaser occurred earlier in the maneuver. This was done to prevent features from leaving the field of view of the camera, as can occur if the camera rotates when it is closer to the target. The gains selected for this scenario were used for the remainder of the scenarios presented in this chapter.

Figure 3.9(b) displays the servo error for all 100 features tracked. Of note, the error for all features does not exactly reach zero (and would not reach zero given longer run time of the simulation). This occurs because the features are trying to reach their goal positions as defined by the goal image, which in this case was taken with a higher focal length camera image. The perspective projection of the higher focal length places feature points at slightly different pixel locations as compared to the shorter focal length of the maneuvering camera. As can be seen, this is not a problem though as the architecture calculates navigation and

control solutions in the least-squares sense, which has the effect of minimizing error on the whole for the entire set of features.

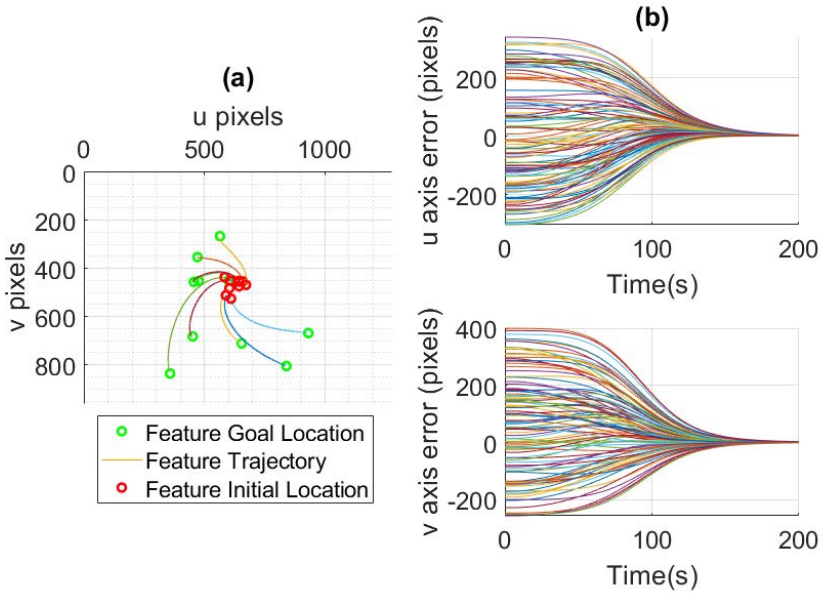

Figure 3.9: Scenario 3P1: (a) image plane trajectories of 10 target features. (b) Servo error for all target features.

3.4.2.2 Scenario 3P2: Simple Maneuver with Sensor Error Modeling.

Scenario 3P2 is identical to scenario 3P1 with the exception that it incorporates the sensor modeling error discussed at the start of this section. Together with scenario 3P1, the results of this scenario demonstrate how realistic sensor noise could be expected to change performance.

Figure 3.10 displays the position of the chaser relative to the target/LVLH origin over the 200 second maneuver. The trajectory is very similar to that of Scenario 3P1, except the deviation in the negative radial direction is magnified (peaks at about 0.5 meters). Additionally, there is a slight deviation in the cross-track axis that occurs early in the maneuver. Both of these deviations are corrected as the chaser reaches its goal pose. It is not surprising that larger deviations from the ideal straight-line trajectory occur at farther

relative distances: feature pixel error has a bigger effect at range and stereo depth estimation error increases with distance. As they occur farther out and are corrected as relative range decreases, these slight deviations in position do not affect the outcome of the maneuver.

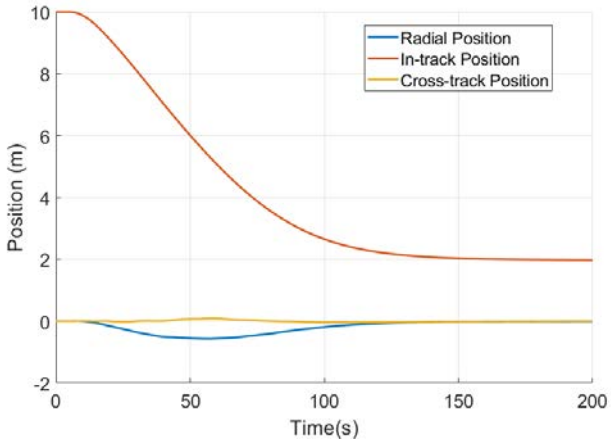

Figure 3.10: Scenario 3P2: position of chaser with respect to LVLH frame.

The velocity profile of the maneuver (Figure 3.11) is similar to that of scenario 3P1, with the main difference being the velocities in the radial/cross-track directions and around the body x and y directions vary more as a result of sensor noise. Notably, the velocity in the direction of and around the optical axis (in-track direction in this case) is smoother and very similar to that of scenario 3P1. Figure 3.12 shows the feature trajectories and servo error in the image domain, and it can be seen that these results are also similar to those of scenario 3P1.

As with any GNC system, it was found that the system's ability to reject sensor noise is influenced by tuning of parameters in the navigation and control subsystems. Since the results presented in this chapter were produced after tuning of the navigation filters and control gains, it is important to note that performance could be potentially degraded or enhanced depending on choice of parameters.

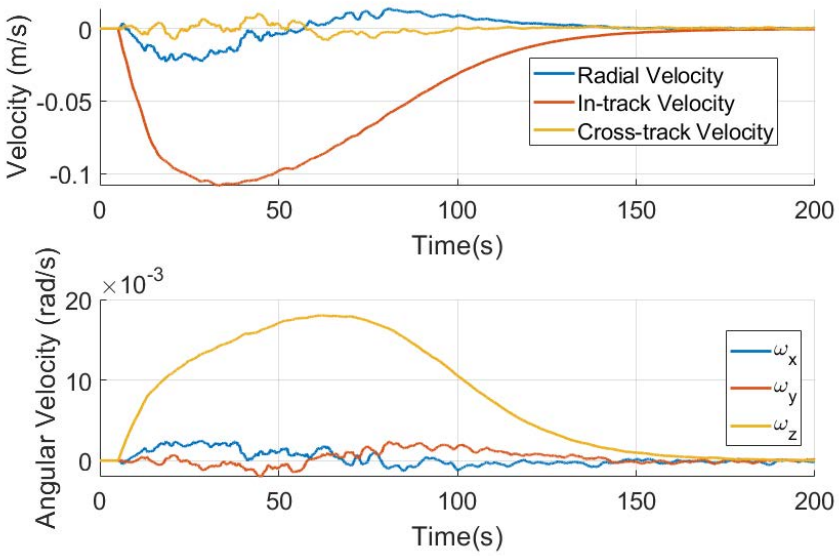

Figure 3.11: Scenario 3P2: velocity of chaser with respect to LVLH frame.

Regarding tuning for the results shown, control gains (for the IBVS controller and PI controllers) were selected to command relative velocities that were approximately commensurate with recommended velocities for established approach strategies (see [7] for example). Kalman filter process noise parameters were lowered to smooth the state estimates for features as much as possible without introducing too much lag in the estimate relative to the true state. Note that while the lower process noise helps smooth the state estimates, if control gains are increased too much, or if a large disturbance occurs during maneuvering (if the target maneuvers rapidly for example), it is possible that the system could enter an oscillation mode where the controller tries to respond to increasingly delayed navigation estimates.

Another important insight gleaned from the results of this chapter was the system's susceptibility to disruption as a result of measurement outliers. If Mahalonobis distance monitoring is not implemented (or a poor threshold is chosen), it was observed that non-rejected outliers often have a very detrimental affect on performance of the system,

even though there should be some inherent robustness as both the navigation and control solutions are computed in a least-squares sense. As could be expected, having a higher ratio of inliers to outliers made the system less susceptible to failure due to unrecognized outlier measurements. Properly tuning the Kalman filter parameters (especially the Mahalonobis distance parameter for outlier rejection) is important to ensure as few outliers are used in navigation and control calculations as possible. In practice, it is the opinion of this author that parameters for the outlier rejection method implemented in the sensing subsystem (typically RANSAC) should also be tuned to ensure minimal presence of outliers, even at the expense of inadvertently rejecting some inliers.

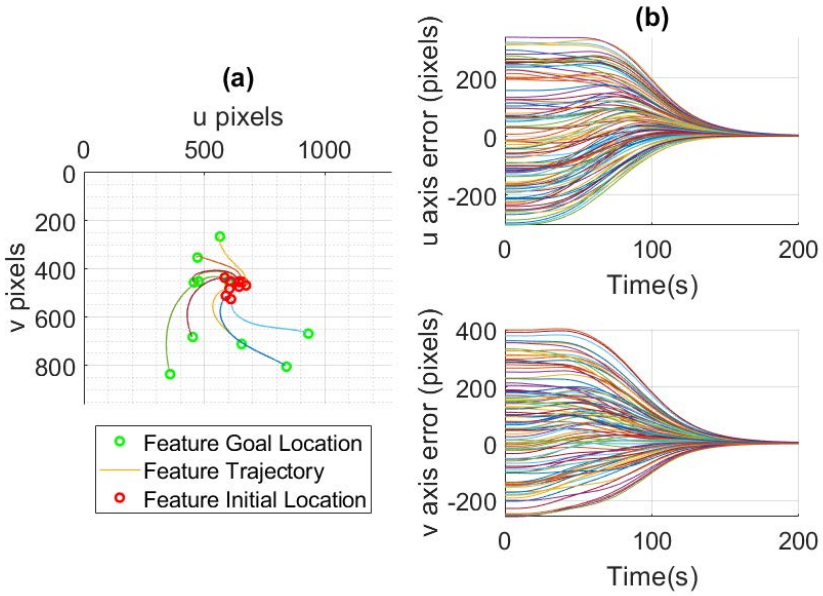

Figure 3.12: Scenario 3P2: (a) image plane trajectories of 10 target features. (b) Servo error for all target features.

Overall, the results of this scenario show that, given properly tuned filter parameters and control gains implemented to produce sufficiently slow relative motion, performance under realistically noisy conditions is comparable to noise-free conditions.

3.4.2.3 Scenario 3P3: 6DOF Maneuver with Static Target.

Scenarios 3P3, 3P4, and 3P5 examine various different initial conditions, requiring the chaser vehicle to conduct maneuvers in all 6DOF. The initial pose for both the chaser and target is the same for all three scenarios: the target is located at the LVLH origin and oriented with its planar ROI facing in the positive in-track direction. The chaser is initially located at $[5.0, 8.0, 3.0]$ meters in the LVLH frame, with an initial orientation as can be seen in Figure 3.14. Figure 3.13(a) shows the initial view from the chaser's camera, and Figure 3.13(b) shows the final view at completion of the maneuver.

Scenarios 3P3, 3P4, and 3P5 all utilize the second method for goal image generation discussed in Section 3.3 (the perspective warp method), which was implemented as discussed in Section A.1.3 (for these scenarios the planar ROI depicted in Figure 3.5 was selected). In implementing this method, the final size of the target ROI was selected to be the same as for previous scenarios, in order to make the goal pose once again equate to a position of $[0, 2, 0]$ meters in the LVLH frame (assuming the target remains static).

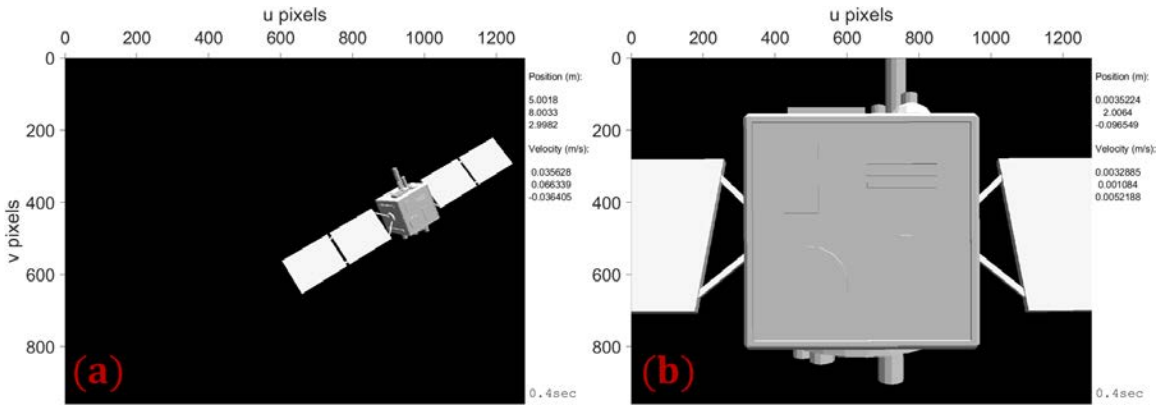

Figure 3.13: Scenario 3P3: camera views from chaser. (a) Initial view. (b) Final view.

For scenario 3P3, the target remains static at the LVLH origin. Figure 3.14 shows two views of the 3D trajectory of the chaser (depicted using the camera symbol) with

respect to the target. These results demonstrate the architecture's ability to smoothly and simultaneously change both orientation and position to achieve the goal pose.

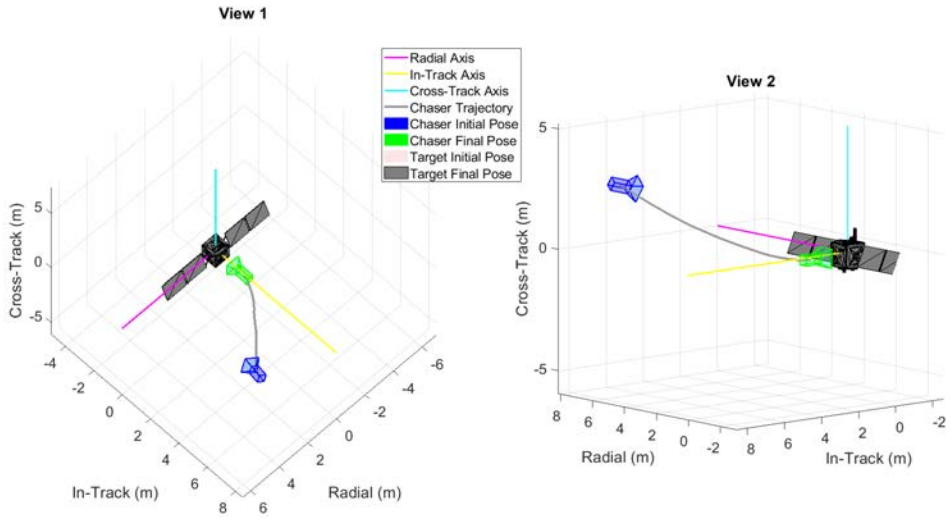

Figure 3.14: Scenario 3P3: 3D trajectory of chaser

Figure 3.15(a) shows the image plane trajectories for 10 target features. Note that the trajectories are well contained within the image plane due to the differential selection of IBVS control gains (i.e. they do not swing wide due to camera rotation late in the maneuver). Figure 3.15(b) shows the servo error for all features collapsing to nearly zero over the course of the maneuver. At the end of the maneuver, the servo error has a mean of 0.87 pixels, and standard deviation of 3.68 pixels. This spread of error is larger than than that of scenario P2 (which had a servo error standard deviation of 2.31 pixels), as a result of the perspective warp not accurately determining the true goal position of features. This variance of the servo error is not a problem in this case as it does not appreciably affect the final pose of the chaser.

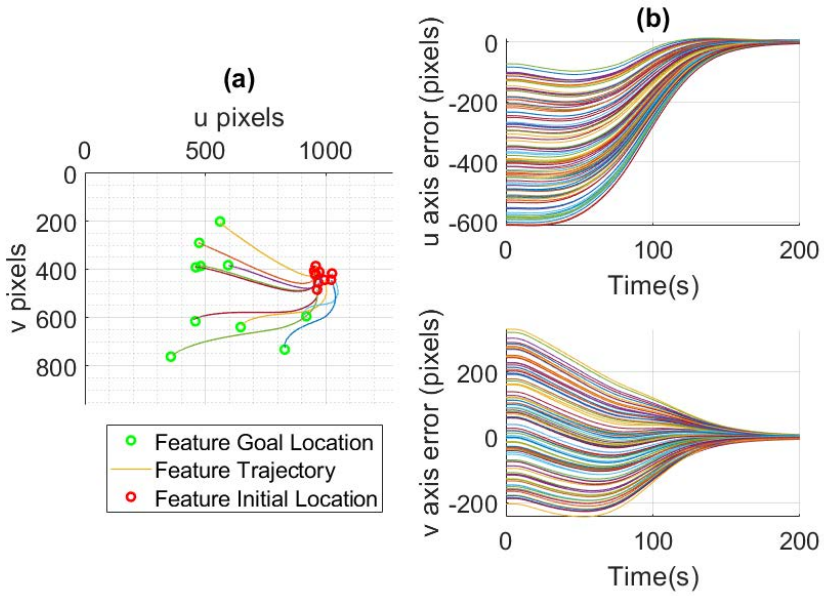

Figure 3.15: Scenario 3P3: (a) image plane trajectories of 10 target features. (b) Servo error for all target features.

Figure 3.16 shows the navigation estimate of relative velocity overlaid with the true relative velocity. The navigation estimate, using the interaction matrix relationship with all visible features for each image frame, is rather noisy, though it generally follows the true relative velocity. Implementing a low pass filter (LPF) was examined to smooth the relative velocity estimate. While the LPF did smooth the relative velocity estimate (and also introduce some lag in the estimate), it did not appreciably improve results for the scenarios tested.

The Kalman filter error for a single image feature is plotted in Figure 3.17. Additionally, the raw measurement error is shown in red, with large spikes in error corresponding to image frames where the feature was cast as an outlier. Small increases in filter variance occur as the feature is not identified in all frames or rejected as an outlier. The fact that the filter error and variance remains converged though for the duration of

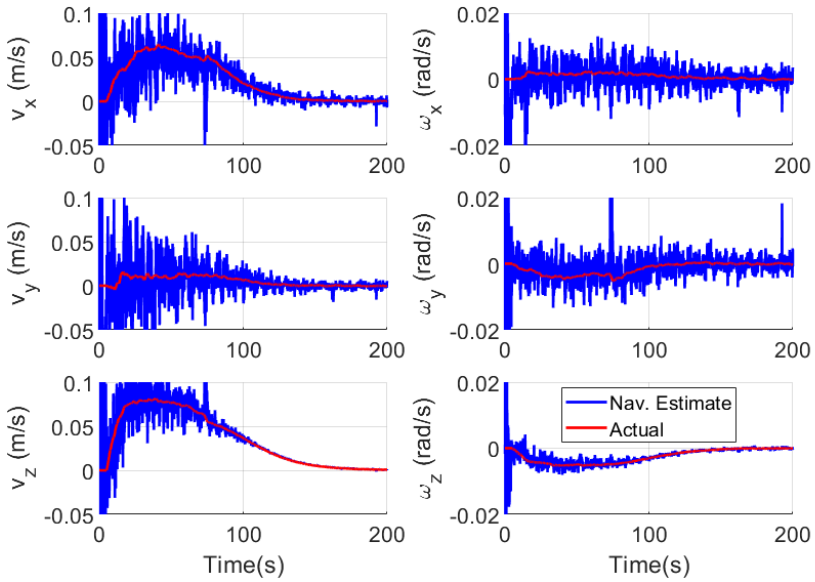

Figure 3.16: Scenario 3P3: estimated and actual relative velocity of chaser with respect to target.

the maneuver implies that the Mahalanobis residual monitoring method works to reject outliers.

Overall, this scenario demonstrates the ability for the architecture to command a coupled 6DOF maneuver using the warp method for goal image generation. The low variance of the servo error at the end of the maneuver indicates that the warp method does a good job approximating feature point locations as they occur in the actual image taken from the final pose of the chaser.

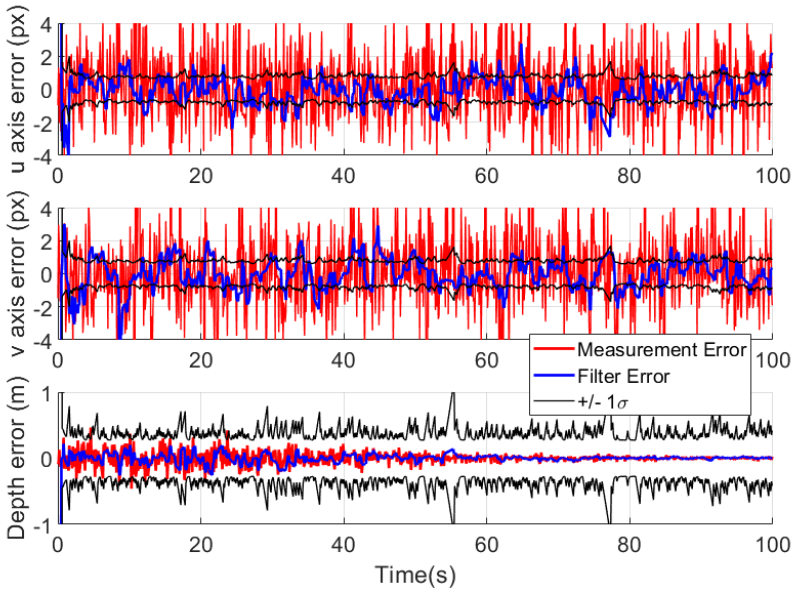

Figure 3.17: Scenario 3P3: Filter error for one target feature (first 100 seconds of simulation)

3.4.2.4 Scenario 3P4: 6DOF Maneuver with Tumbling and Drifting Target.

This scenario is identical to scenario 3P3 with the exception that the target is given an initial translational and angular velocity. The target's initial translational velocity of $[0.010; 0.005; 0.001]$ meters/second in the LVLH frame is intended to simulate either a maneuvering target, or imperfect knowledge of the LVLH frame origin. The target is also given an initial body frame angular velocity of $[0.000; -0.003; 0.005]$ radians/second with respect to the LVLH frame, as could occur with a damaged or maneuvering target.

Figure 3.18 shows two views of the chaser and target's 3D trajectories (with the initial and final pose for each spacecraft plotted). As can be seen, the chaser captures its goal pose even as the target changes position and orientation throughout the simulation. Once the goal pose is achieved, it is indefinitely maintained as any variation in visual servo error continues to be corrected.

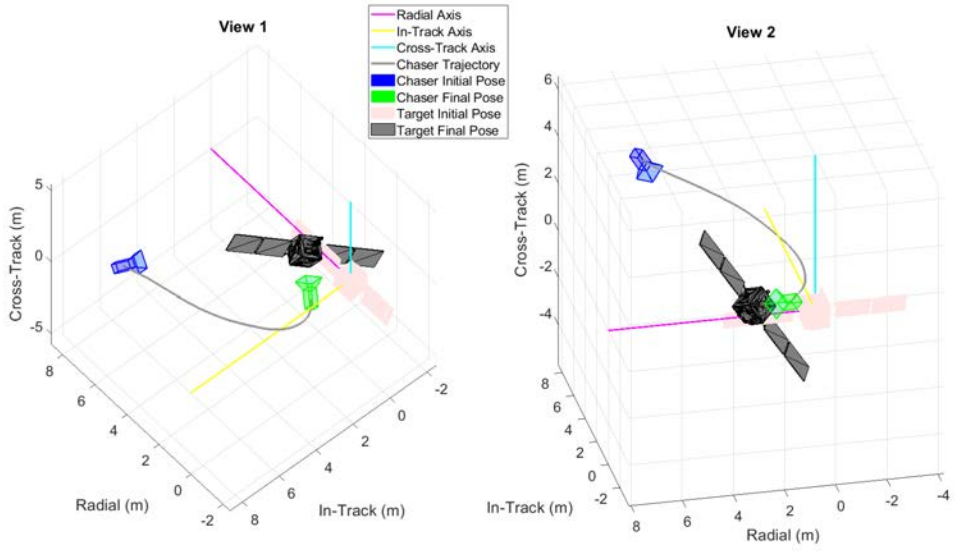

Figure 3.18: Scenario 3P4: 3D trajectory of chaser and target

Figure 3.19 shows feature trajectories and servo error. As with previous scenarios, feature trajectories are well contained and smooth. The servo error takes longer to collapse than in scenario P3, and the mean error does not make it as close to 0. Additionally, the standard deviation of error is larger (14.85 pixels). It is possible to improve the convergence rate and accuracy of servo error (and hence final pose) using higher control gains, though this was not done for these results in order to facilitate direct comparison among scenarios.

The estimated and actual relative velocities are shown in Figure 3.20. As with previous scenarios, the estimate is noisy, though it tracks the actual relative velocity well enough to affect a successful maneuver. The fact that this method can sufficiently estimate the relative velocity when both the chaser and target are moving is significant because it only requires information from the camera sensor, and does not require any specific knowledge of the target or its current state.

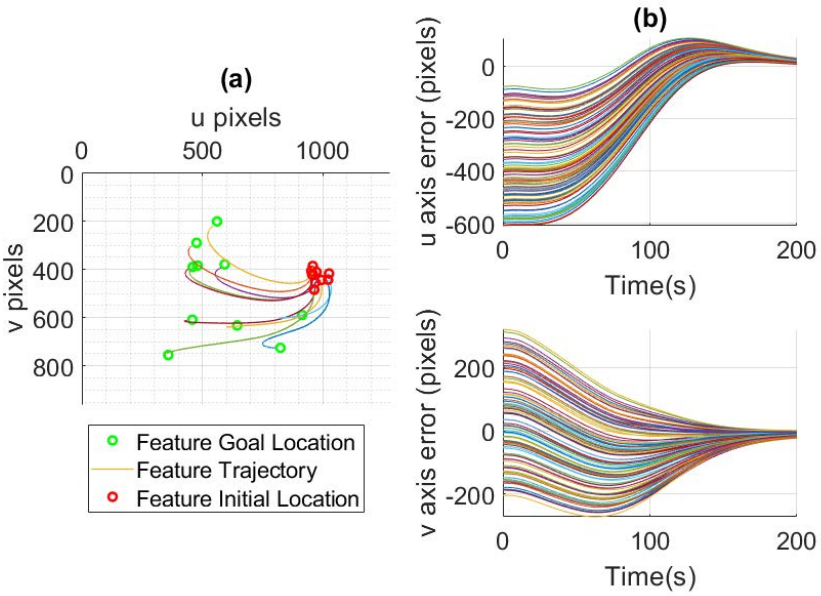

Figure 3.19: Scenario 3P4: (a) image plane trajectories of 10 target features. (b) Servo error for all target features.

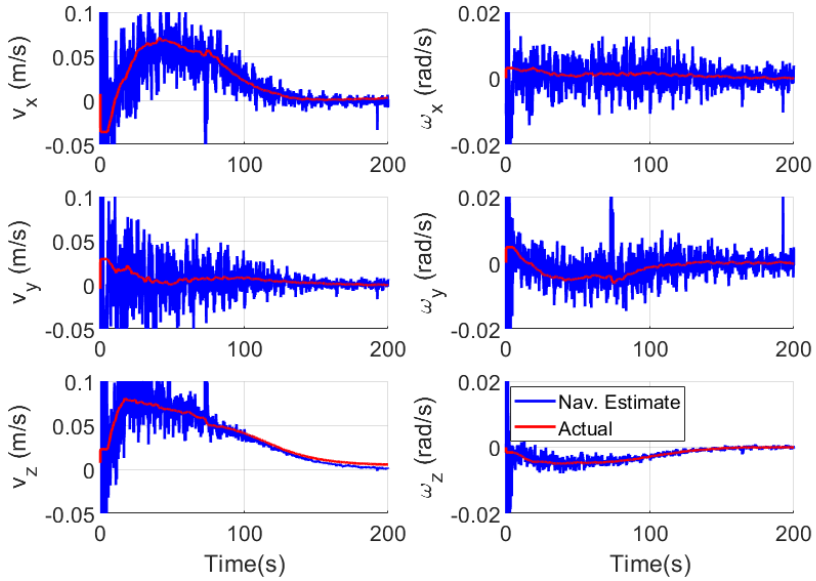

Figure 3.20: Scenario 3P4: estimated and actual relative velocity of chaser with respect to target.

This scenario demonstrates the architecture's ability to maneuver with respect to a target moving in 6DOF, without requiring any additional knowledge regarding the movement of the target other than what is available from the camera sensor. Additional testing with different initial conditions appears to indicate that the ability to maneuver with respect to faster moving targets is primarily limited by whether the chaser has control authority available to match the 6DOF velocity of the target. Assuming it does, this method can be applied for a wide range of target velocity conditions.

3.4.2.5 Scenario 3P5: 6DOF Maneuver with Reduced Target Features.

The final scenario using a planar target ROI examines how reducing the number of target features tracked affects performance. This is important to consider, as the image processing pipeline may not be able to extract as many features for bland or otherwise mostly uniform target ROI's. Additionally, it is reasonable to expect that fewer features will be detectable for low illumination conditions. Accordingly, this scenario assumes 20 target features are tracked (as opposed to 100 for previous scenarios). One feature is recast as an outlier for each image frame (5% of the total features), and 50% are still assumed to not be identified in each frame. This leaves up to 9 features available for navigation and control calculations. Other than the difference in features, this scenario is identical to scenario 3P3.

Figure 3.21 shows two views of the 3D trajectory of the chaser, and Figure 3.22 shows feature trajectories and servo error. As can be seen, the results are very similar to those of scenario 3P3. In fact, the difference in quality of the maneuver and how well the goal pose is achieved compared to scenario 3P3 is almost not discernible.

Figure 3.23 shows the estimated and actual relative velocities. The quality of the relative velocity estimate is worse than that of scenario 3P3, though not significantly enough to drastically affect the maneuver, even though 1/5th the number of features are used for the calculation.

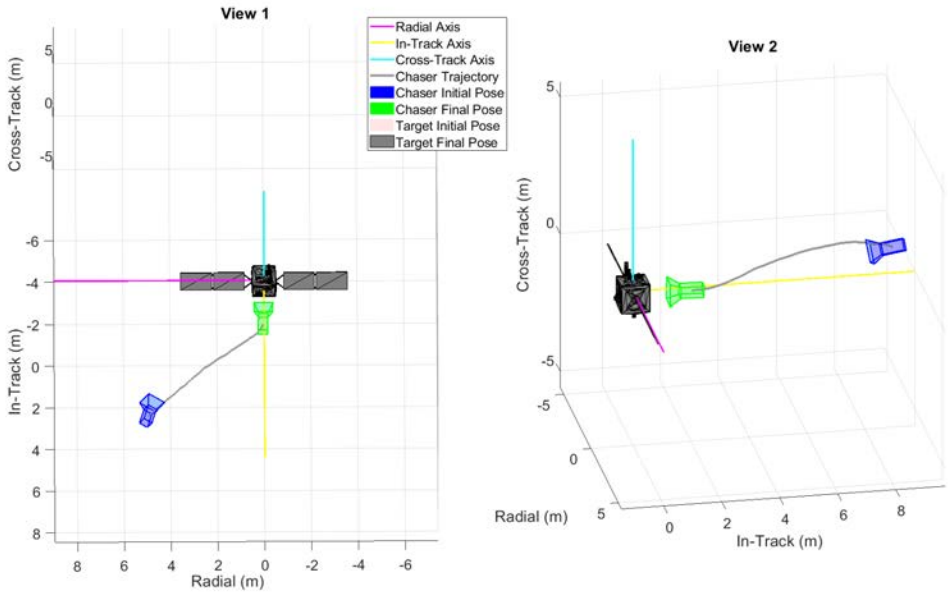

Figure 3.21: Scenario 3P5: 3D trajectory of chaser and target

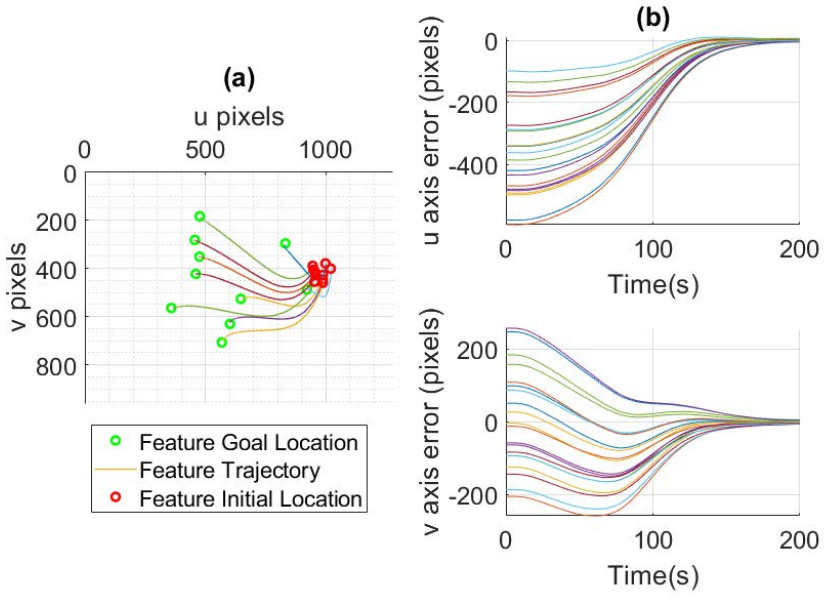

Figure 3.22: Scenario 3P5: (a) image plane trajectories of 10 target features. (b) Servo error for all target features.

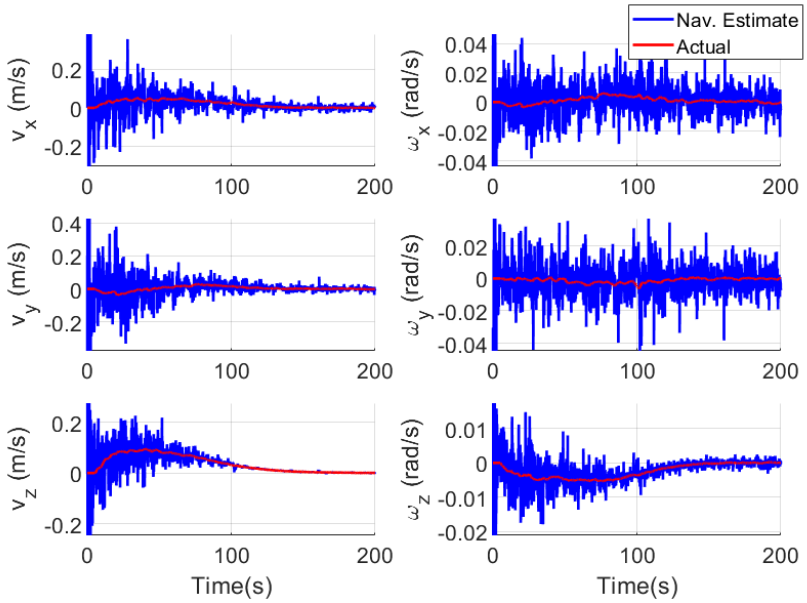

Figure 3.23: Scenario 3P5: estimated and actual relative velocity of chaser with respect to target.

It was observed in additional testing that as the total number of target features decreases, the affect of non-rejected outliers becomes more drastic (as could be expected). Consistently good results were achieved with as little as 16 features tracked, though it is apparent that even fewer features could be acceptable if outlier rejection is assured.

This scenario shows that the architecture is capable of maneuvering with a very small number of feature points available. As stated, this is a significant capability for bland target surfaces or low-light conditions.

3.4.3 Maneuvering with Respect to a Nonplanar (NP) ROI.

This subsection considers maneuvers with respect to regions of interest on the target that are nonplanar. Figure 3.24 shows a CAD view of the target, with the ROI selected for maneuvering highlighted in red. The range of depth for protrusions and cavities within the ROI is 67.5 centimeters.

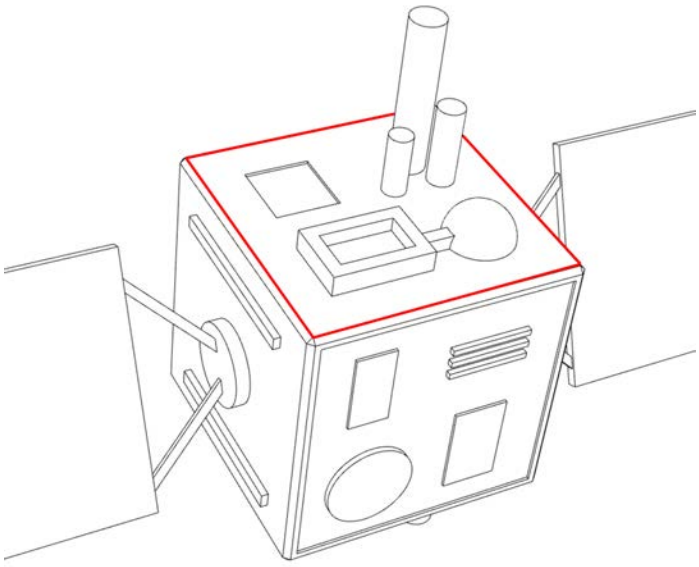

Figure 3.24: CAD model of target used in simulation, with nonplanar ROI for maneuvering highlighted in red

Results for three scenarios are presented in the following subsections (an overview is provided in Table 3.3) . The initial conditions for all three scenarios are the same, though each scenario uses a different method for either generating the goal image or conducting the maneuver. The first scenario uses the first method for generating a goal image (zooming camera), and the next two scenarios use the second method (perspective warp). The third scenario is similar to the second scenario, except it implements the overall maneuver as three separate incrementally implemented maneuvers.

Table 3.3: Overview of Maneuver Scenarios with Respect to Nonplanar Target ROI

Scenario	Maneuver Description	Goal Image Generation	Sensor Noise Modeling
3NP1	Complex maneuver with static target (single maneuver)	Method 1 (zoom)	Yes
3NP2	Complex maneuver with static target (single maneuver)	Method 2 (warp)	Yes
3NP3	Complex maneuver with static target, incrementally implemented as two maneuvers	Method 2 (warp)	Yes

3.4.3.1 Scenario 3NP1: Maneuver with Respect to Nonplanar ROI Using Zoom Method for Goal Image Generation.

For this scenario, the chaser is initially placed at $[5.0, 5.0, 10.0]$ meters in the LVLH frame, oriented facing the target as shown in Figures 3.25(a) and 3.26. The target remains static at the LVLH origin, with its nonplaner ROI facing in the positive cross-track direction. The goal of the maneuver is the similar to previous scenarios: approach the target ROI such that it takes up most of the camera's field of view (for these scenarios the goal pose equates to a position of $[0, 0, 2]$ meters in the LVLH frame), with the camera plane parallel to the ROI as shown in 3.25(b) (i.e. 0° tilt with respect to the radial and in-track axes).

The zoom method is used to generate the goal image. As the target ROI is not perpendicular to the optical axis of the camera at the start of the maneuver, it is assumed that it was perpendicular at some point prior to maneuvering (during an inspection NMC, for example), at which point the goal image was generated using a higher focal length camera image, as in scenarios 3P1 and 3P2.

Figures 3.26 and 3.27 show the trajectory of the chaser with respect to the target/LVLH frame. The chaser asymptotically approaches the goal pose, and completes the maneuver with the image plane parallel to the target ROI, as desired.

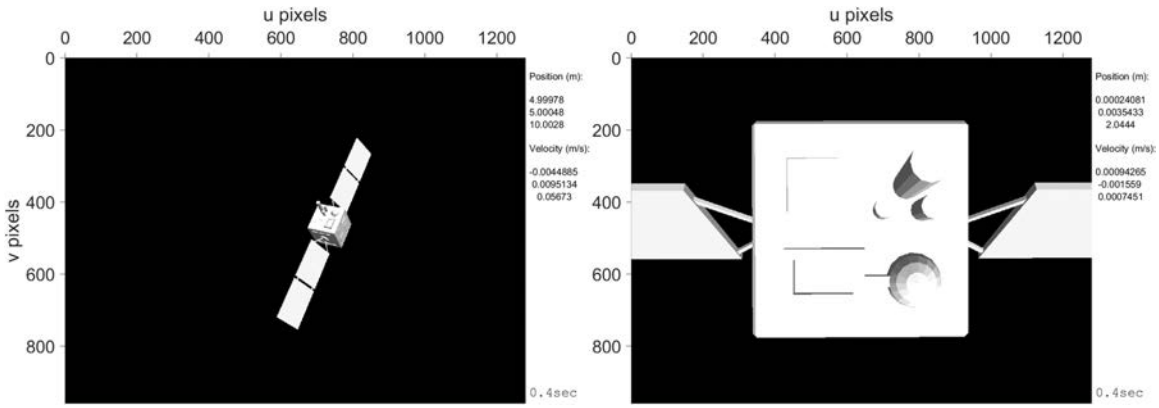

Figure 3.25: Scenario 3NP1: camera views from chaser. (a) Initial view. (b) Final view.

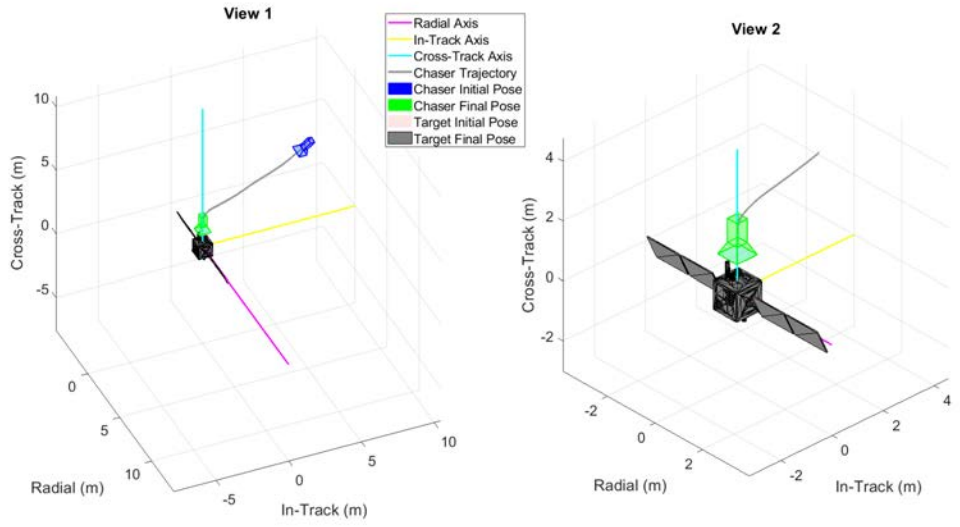

Figure 3.26: Scenario 3NP1: 3D trajectory of chaser and target

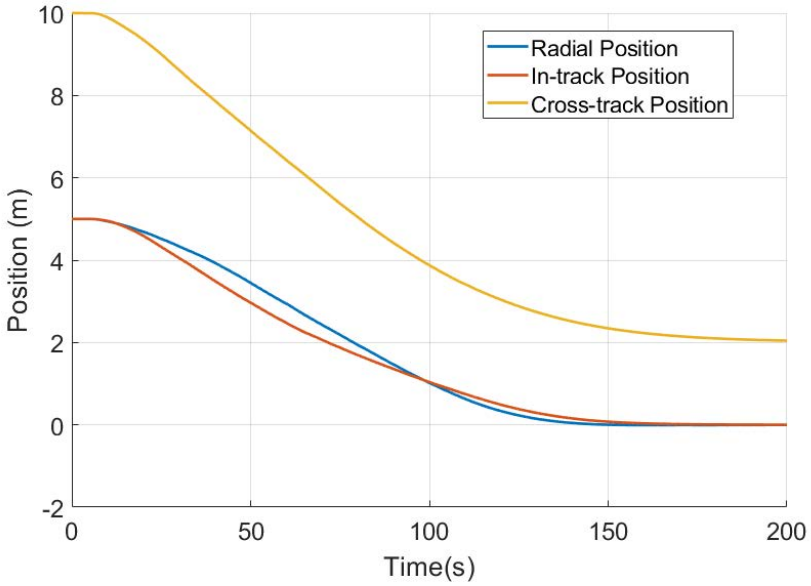

Figure 3.27: Scenario 3NP1: position of chaser with respect to LVLH frame.

Figure 3.28 displays the velocity of the chaser with respect to the LVLH frame. As with previous results, the velocity decreases smoothly during the later half of the maneuver. The magnitude of the velocity slows to less than 1 cm/s as the chaser approaches within 25 centimeters of its final pose.

Figure 3.29 displays the image plane trajectories and servo error, showing desirable results for the motion of the features. The standard deviation of the servo error is 4.88 pixels, larger than that of maneuvers with respect to static planar ROI's, but still good enough to have minimal effect on the chaser's pose at the end of the maneuver.

Overall, it is apparent that the zoom method works well to generate a goal image for the nonplanar surface examined, as the results for the final relative pose of the maneuver are very comparable to those previously presented for the planar ROI.

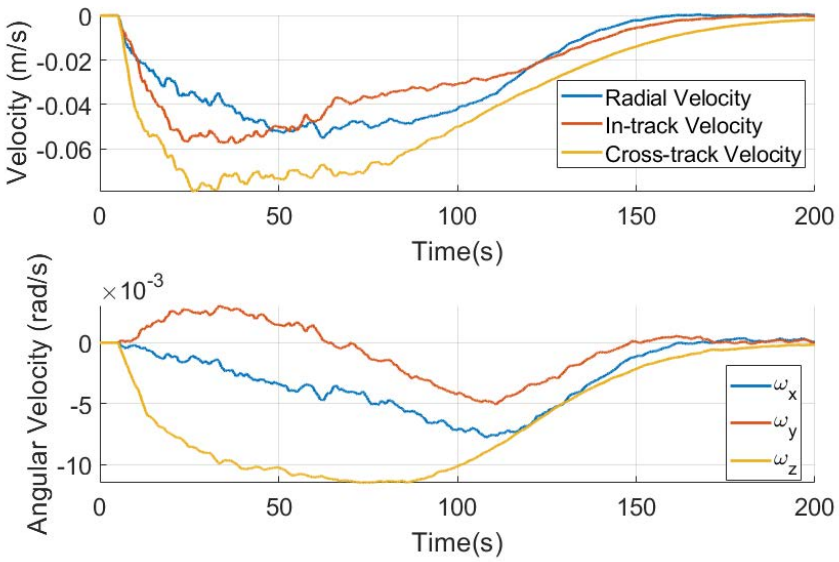

Figure 3.28: Scenario 3NP1: velocity of chaser with respect to LVLH frame.

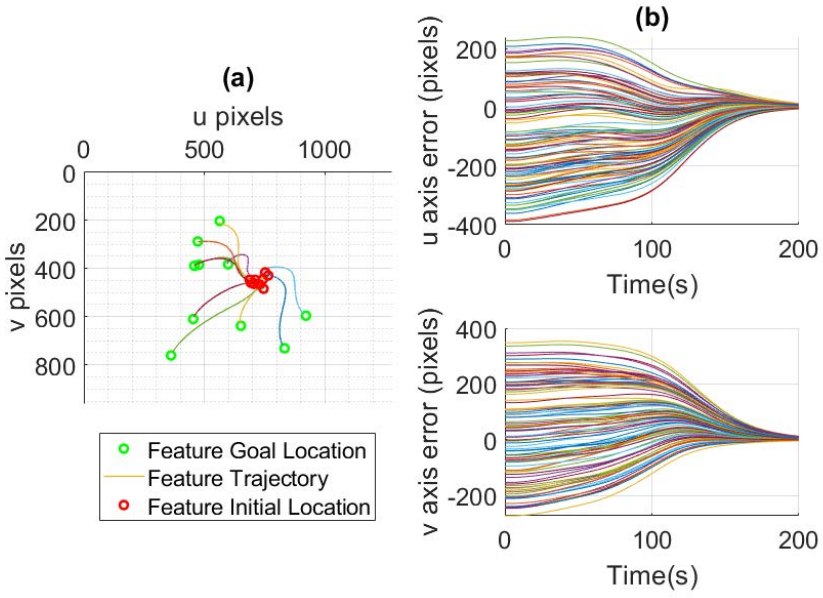

Figure 3.29: Scenario 3NP1: (a) image plane trajectories of 10 target features. (b) Servo error for all target features.

3.4.3.2 Scenario 3NP2: Maneuver with Respect to Nonplanar ROI Using Perspective Warp Method for Goal Image Generation.

Even after conducting inspection phase maneuvers, it is possible that the chaser camera could never have been positioned with the ROI sufficiently perpendicular to the optical axis such that the zoom method for goal image generation could be used. Accordingly, it is necessary to examine the warp method for goal image generation of nonplanar ROI's. This scenario is identical to 3NP1, except it uses the warp method to generate the goal image.

Figure 3.30 shows the 3D trajectory of the chaser, with an enlarged view of the final pose of the chaser at the end of the simulation, showing that it does not quite achieve an orientation where the camera plane is parallel to the target ROI. Figure 3.31 shows the view from the chaser camera at the end of the simulation. At completion of the maneuver, the camera axis is tilted $-11.3°$ around the radial axis and $14.0°$ about the in-track axis (all other scenarios with static targets achieved a tilt within $0.2°$ of each axis). The final position of the chaser is $[0.34, 0.31, 1.90]$ meters in the LVLH frame (as opposed to $[0.00, 0.00, 2.04]$ meters for scenario 3NP1).

These deviations from the results of previous scenarios are a function of the perspective warp inaccurately estimating to goal feature locations for the nonplanar target surface. The feature trajectories and servo error (Figure 3.32) convey this: the target features fail to reach their goal positions as well as in other maneuvers with static targets. The standard deviation of the servo error at the end of the maneuver, 12.50 pixels, is much larger than for previous scenarios.

The results of this scenario convey an important possible degeneracy of the warp method for goal image generation: as this method assumes a planar surface is being warped, if the target surface is not planar, the chaser may be less likely to achieve the desired goal

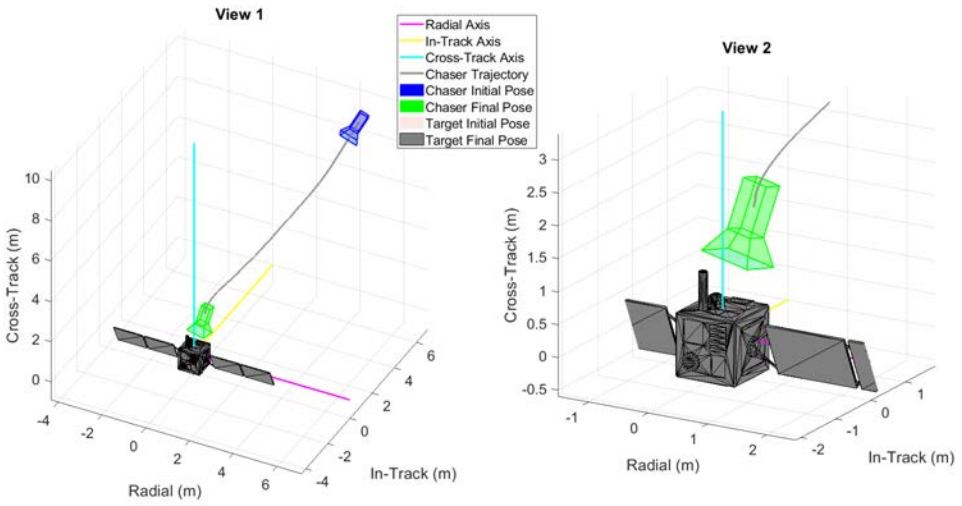

Figure 3.30: Scenario 3NP2: 3D trajectory of chaser and target

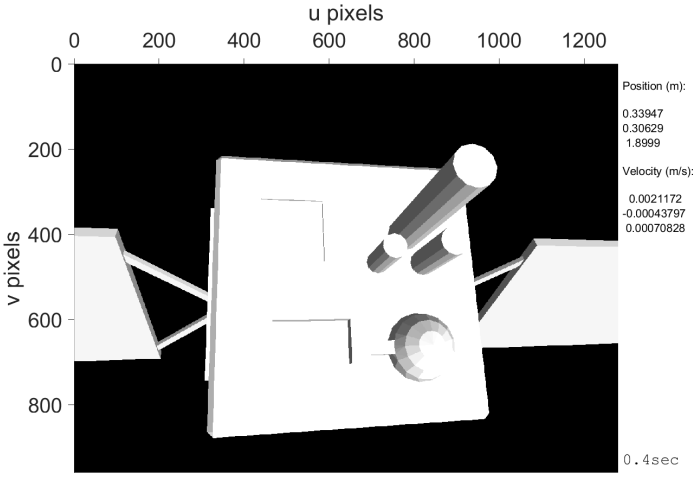

Figure 3.31: Scenario 3NP2: View from the chaser camera at end of simulation

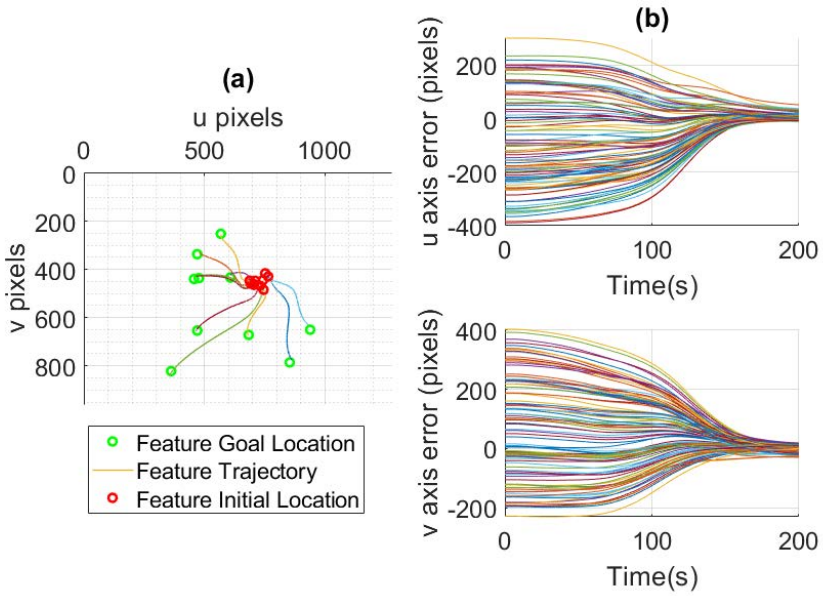

Figure 3.32: Scenario 3NP2: (a) image plane trajectories of 10 target features. (b) Servo error for all target features.

pose. One possible solution to this degeneracy is to manually select target features within the target ROI that appear to lie on the same plane (points on the surface of the case of the spacecraft bus, for example) prior to generating the goal image using the perspective warp (and subsequently only use these approximately planar features during maneuvering). This may not be feasible or practical for a remote operator though, so the next scenario examines a different option to improve the quality of the final relative pose.

3.4.3.3 Scenario 3NP3: Incremental Maneuvering with Respect to Nonplanar ROI.

This scenario examines one possible solution to the degeneracy presented in scenario 3NP2. Instead of conducting the entire maneuver all at once, it is proposed to break up the maneuver into multiple sub-maneuvers, each implemented back to back. The thought here is that after each maneuver, the camera gets incrementally closer to achieving the desired relative orientation, which makes the perspective warp for the next sub-maneuver's

goal image more accurate. This scenario breaks the whole maneuver up into three sub-maneuvers, though in practice more or less sub-maneuvers could be implemented.

For each sub-maneuver, a goal image is generated with the target ROI made incrementally larger in the camera field of view. Figure 3.33 depicts the trajectories of the three sub-maneuvers in shades of blue, and the trajectory from scenario 3NP2 is provided in red for reference. As can be seen, the chaser is closer to achieving the desired goal pose after the incremental maneuvers as compared to scenario 3NP2. Table 3.4 summarizes values for the final position and orientation for this scenario and previous two scenarios with nonplanar ROI's. Figure 3.34 shows the view from the chaser's camera at the end of the simulation.

Additional testing of this method confirms that it produces appreciable improvements in the quality of the final relative pose, as compared to conducting one maneuver (using perspective warp for goal image generation). However, the final relative pose provided by this method still may not be considered accurate enough, depending on the application. Consequently, additional methods for conducting complex maneuvers with respect to nonplanar ROI's will be examined in Chapter IV.

Table 3.4: Summary of Results for Scenarios with Nonplanar Target ROI

	Scenario		
	3NP1 (zoom)	**3NP2 (warp)**	**3NP3 (warp, 3x incremental)**
Position	[0.00, 0.00, 2.04] m	[0.34, 0.31, 1.90] m	[0.11, 0.14, 1.99] m
Tilt about radial axis	0.18°	−11.3°	−8.08°
Tilt about in-track axis	0.14°	14.0°	3.81°
Servo Error Standard Deviation	4.88 pixels	12.50 pixels	6.55 pixels

*All values are for end of maneuver

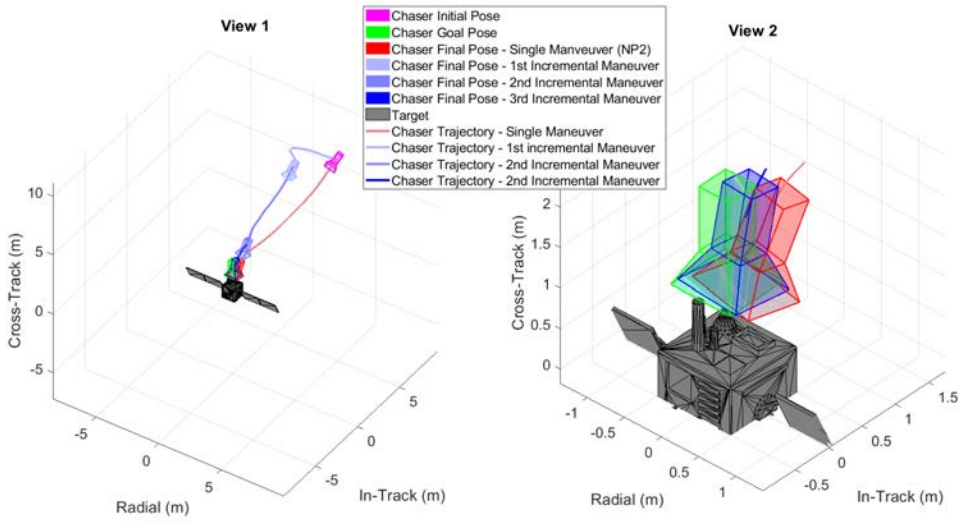

Figure 3.33: Scenario 3NP3: Comparison of 3D trajectories between scenario 3NP2 and 3NP3

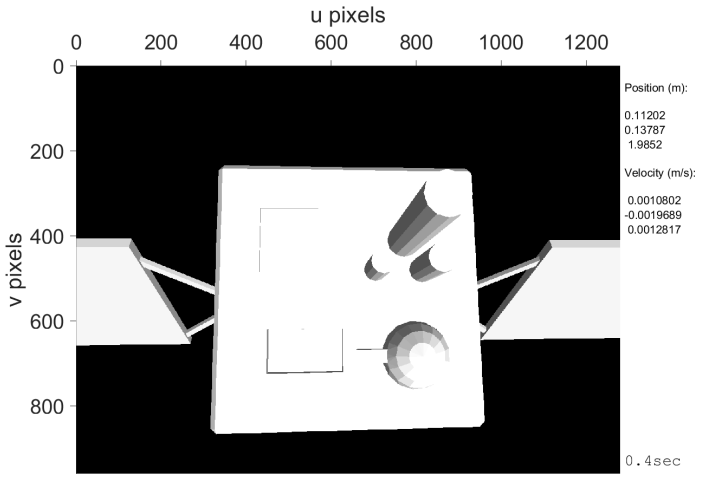

Figure 3.34: Scenario 3NP3: View from the chaser camera at end of maneuvering

3.4.4 Results Summary.

As stated in the introduction to this section, these results are not intended to represent any particular spacecraft mission or use case. They do however show the proposed architecture's ability to work for a variety of potential final approach RPO maneuvers.

Together these scenarios demonstrate that is possible to perform very complex RPO maneuvers using only information from a stereo camera sensor.

While the results presented generally depict the ability for the chaser to robustly capture a goal pose as defined by a generated goal image, it is noted that there is one possible degeneracy when the warp method is used to generate a goal image for highly nonplanar target ROI's. A method was examined to improve the quality of the final relative pose for this case by using multiple incrementally implemented maneuvers. It was shown that this does improve the convergence to the desired relative pose, but this method would impose an additional burden on remote operators in order to implement multiple maneuvers. Chapter IV of this work will examine a different type of visual servoing in an effort to improve relative maneuvers with respect to unknown nonplanar target ROI's.

3.5 Conclusion

This chapter proposed an architecture for autonomously conducting 6DOF final approach maneuvers with respect to unknown targets. Various aspects of sensing, navigation, and control were treated.

The architecture uses a low-cost, low-SWAP stereo camera as the primary relative navigation sensor, coupled with a feature-based computer vision pipeline to process stereo images. A bank of Kalman filters are used to estimate states for the tracked target features, which are in turn used to estimate the relative velocity between chaser and target. An image-based visual servo control law is implemented to generate control commands for 6DOF relative maneuvers.

Two methods were proposed to generate goal images for visual servo maneuvers with unknown targets. One method uses a zooming camera, and the other uses a perspective warp to generate an approximate image from the desired relative goal pose. Both of these methods were tested in simulation and found to work well with planar ROI's, including when the target was tumbling or drifting. Nonplanar target ROI's were also examined; while the zoom method worked well for this case, the warp method performed somewhat poorly in commanding the chaser to the desired relative pose. A method of incremental maneuvering using the warp method was proposed and was shown to provide better results as compared to conducting a single maneuver with the warp method. Improving maneuvering with respect to nonplanar ROI's will be a primary objective of Chapter IV of this work.

The proposed navigation and control methods were demonstrated using a 6DOF simulation. While image processing was not implemented, the stochastic nature of image feature tracking was modeled via four mechanisms. Results show that the architecture is robust to levels of camera sensor measurement error that are considered greater than what could be expected in reality. Furthermore, the results demonstrate the architecture's ability to command complex maneuvers with respect to tumbling/drifting targets.

Finally, it is worth reiterating that the proposed architecture not only does not require any knowledge of the target or its state, it does not require a relative pose estimate between the chaser and target. This is significant because computationally intensive and potentially brittle pose estimation algorithms are not required to perform these final approach maneuvers, even when the target is translating and rotating.

IV. Hardware-in-the-Loop Implementation of GNC Method 1

4.1 Overview

This chapter details work done to experimentally validate the framework proposed in Chapter III, with the intent to answer Research Question 3 as it applies to Research Question 1 (see Section 1.2). The chapter includes a description of the experimental test setup, modifications/additions made to the Chapter III architecture necessary to implement this method with hardware-in-the-loop (HITL), experimental results for various test cases representative of possible final approach maneuvers with unknown targets, and a discussion of these results including considerations for practical implementation of the method and recommendations for future work.

4.2 Method

The guidance, navigation, and control (GNC) method implemented in this chapter is fundamentally the same as that of Chapter III. Instead of examining the architecture in simulation as in the previous chapter however, this chapter utilizes HITL to assess the architecture under more realistic conditions than can be realized in simulation alone. As one of the main limitations of the simulation used in the previous chapter pertained to the ability to accurately model computer vision algorithms, a stereo camera sensor is used to provide real video imagery and depth measurements. These measurements are in turn used to control movement of a 4 degrees of freedom (DOF) robot with respect to a physical target (also capable of maneuvering in 4DOF). This section describes the experimental setup used for this chapter as well a several extensions to the method that enable its use with HITL.

4.2.1 Experimental Setup.

Physical motion of chaser and target bodies is actualized via the use of the Air Force Institute of Technology (AFIT) Control and Autonomy Space proximity Robot (CASpR). CASpR is a gantry style robotic machine capable of enacting 4DOF motion for two independent bodies (3 translational DOF and 1 rotational DOF). More information regarding the hardware/software structure of CASpR is available in Appendix B. The movement of each CASpR body represents relative motion with respect to the Local-Vertical-Local-Horizontal (LVLH) frame (for this work the origin of this frame is set as the centroid of the target object).

Figure 4.1 shows a view of CASpR setup for the experiments of this chapter. The chaser spacecraft is represented by the center assembly of the bottom gantry machine (specifically, the camera fixed to the top of this assembly), and the target spacecraft is represented by the object suspended from the center assembly of the top gantry machine.

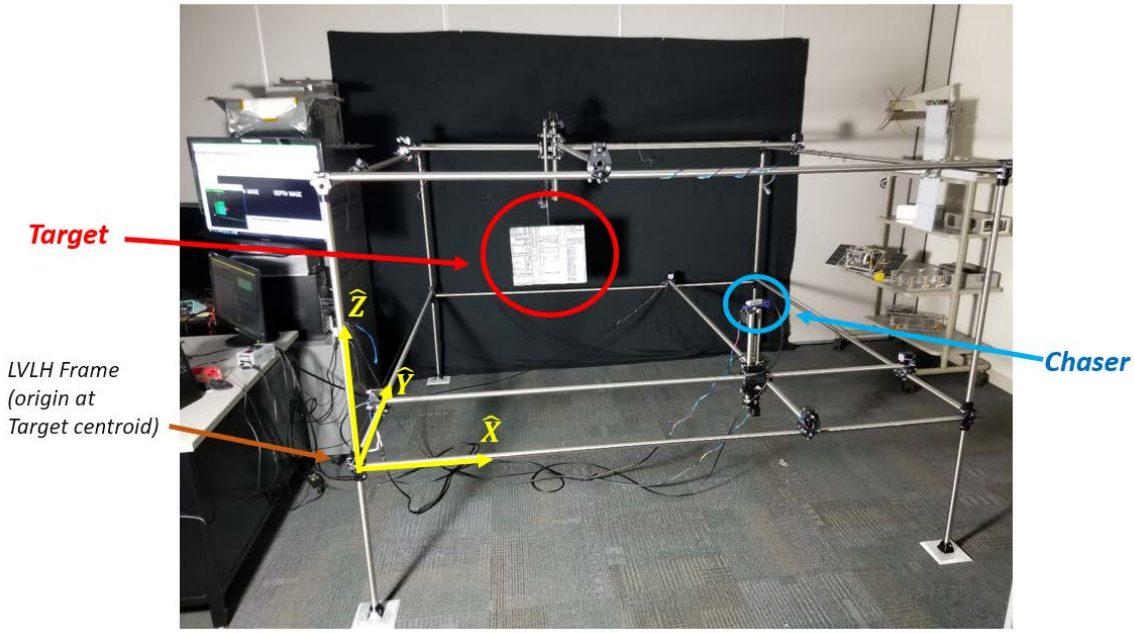

Figure 4.1: CASpR set up for experiments of this chapter with spacecraft bodies and LVLH frame highlighted.

As this work is founded on the use of camera sensors for relative navigation, a stereo camera is mounted on the chaser body (lower machine center assembly), and is the sole sensor used to affect relative maneuvering. The camera used for this work is an Intel® RealSense™ D435 [134, 135], shown in Figure 4.2. The D435 contains a stereo camera pair for measurement of depth, an additional color image sensor, and an onboard processor for calculation of depth maps. The camera also contains an active projector that optically superimposes patterned infrared light on the image scene in order to improve the quality of generated depth maps [136]. This patterned light aids in stereo rectification for texture-less surfaces and during low-light conditions. Not all stereo cameras have projectors however, so the D435's projector is disabled for all of the results contained within this work.

Figure 4.2: Intel® RealSense™ D435 stereo camera sensor [135].

The GNC code for this chapter was written in Python, in order take advantage of the extensive functionality of the OpenCV library. Python was chosen over C++ for ease-of-use. Though C++ generally produces more computationally efficient code, many Python libraries, including OpenCV, utilize compiled C code internally, making the difference in computational speed less drastic. A graphical user interface (GUI) was designed to

facilitate experimentation and observation of relevant metadata in real time. A screen capture of this GUI is shown in Figure 4.3.

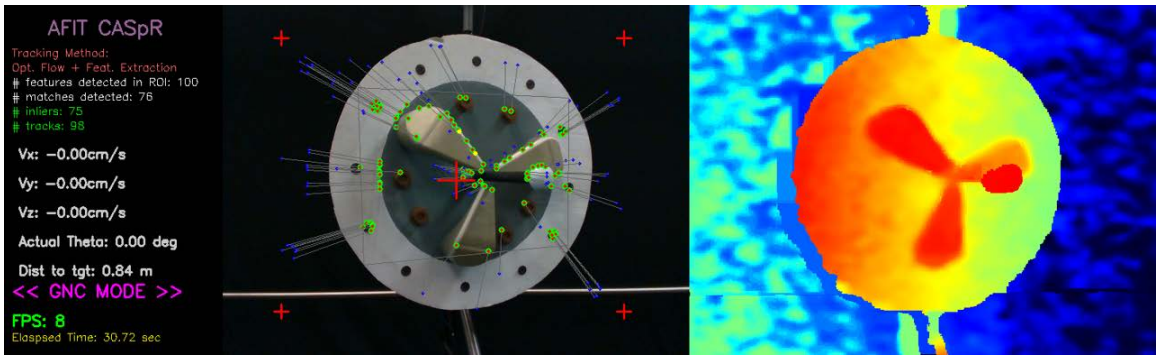

Figure 4.3: Example view of GUI developed to support this research. Camera view shown on left, aligned and filtered depth map on right. Green points in camera image indicate healthy target feature points, blue points indicate goal positions of features, and gray lines connect features to corresponding goal positions.

4.2.1.1 Approximating Truth Data with CASpR.

CASpR utilizes stepper motors to control motion of each body. These motors are operated in open-loop, and provide no feedback on their actual state to the low-level machine controllers. As motor position feedback is not available, nor is any other method of directly recording positional information for each machine (such as a Vicon motion capture system), information on the true state of each body must be inferred from the commands sent to each machine's stepper motors. This is a reasonable approach because stepper motors are extremely precise (they are commonly implemented open-loop in 3D printers and CNC machines, achieving sub-millimeter accuracy even after days of continuous operation). Thus it is assumed that stepper positions commanded by the low-level controller are equivalent to the actual position of the motors. During each scenario stepper motor commands are logged and used in post-processing to generate 'truth' data.

Several plots in Section 4.3 compare velocity estimated by the GNC system to the 'derived actual' velocity. Here, the derived actual velocity is an approximation of the true

velocity derived from the logged positional data of the stepper motors. To calculate this quantity, a discrete derivative of the positional data is taken and a second-order digital butterworth filter is applied to smooth the estimate. This filter is applied forwards and backwards to ensure zero phase lag.

4.2.1.2 Relative Motion Dynamics with CASpR.

As described in Appendix B, low-level microcontrollers are used to control the motion of each body. These microcontrollers receive acceleration commands from a separate 'flight' computer running the GNC algorithms developed herein, and in turn propagate spacecraft relative motion dynamics in order to determine the correct position and velocity to command each body to. As the microcontrollers only have 8-bit 16 MHz processors, it is necessary to implement linearized relative motion dynamics instead of the nonlinear equations of motion that were utilized in simulation. For translational motion, a discretized version of the Hill-Clohessy-Wiltshire (HCW) equations described in Section 2.3.1.1 are implemented in lieu of the Nonlinear Equations of Relative Motion (NERM). Discretized equations for the explicit solution of the HCW equations (equations (2.14)-(2.19)) including additional thruster acceleration terms are implemented on each microcontroller and are shown here for completeness:

$$x_k = \left[4x_{k-1} + \frac{2}{n}\dot{y}_{k-1}\right] + \frac{\dot{x}_{k-1}}{n}\sin(n\Delta t) - \left[3x_{k-1} + \frac{2}{n}\dot{y}_{k-1}\right]\cos(n\Delta t), \tag{4.1}$$

$$y_k = -\left[6nx_{k-1} + 3\dot{y}_{k-1}\right]t + \left[y_{k-1} - \frac{2\dot{x}_{k-1}}{n}\right] + \left[6x_{k-1} + \frac{4}{n}\dot{y}_{k-1}\right]\sin(n\Delta t) + \frac{2}{n}\dot{x}_{k-1}\cos(n\Delta t), \tag{4.2}$$

$$z_k = \frac{\dot{z}_{k-1}}{n}\sin(n\Delta t) + z_{k-1}\cos(n\Delta t), \tag{4.3}$$

$$\dot{x}_k = \dot{x}_{k-1}\cos(n\Delta t) + \left[3x_{k-1}n + 2\dot{y}_k\right]\sin(n\Delta t) + u_x\Delta t, \tag{4.4}$$

$$\dot{y}_k = -\left[6nx_{k-1} + 3\dot{y}_{k-1}\right] + \left[6\dot{x}_{k-1}n + 4\dot{y}_k\right]\cos(n\Delta t) - 2\dot{x}_{k-1}\sin(n\Delta t) + u_y\Delta t, \tag{4.5}$$

$$\dot{z}_k = \dot{z}_{k-1}\cos(n\Delta t) - z_{k-1}n\sin(n\Delta t) + u_z\Delta t, \tag{4.6}$$

where k is the timestep number, Δt is the time from the previous timestep to the current timestep, $\boldsymbol{u} = [u_x, u_y, u_z]^T$ is the translational acceleration vector imparted by control thrusters, resolved in the LVLH frame, and all other variables are as defined in Section 2.3.1.1.

CASpR has one rotational DOF for each body, about the LVLH Z-axis, termed θ. Discretized equations for this state are developed from (2.21),

$$\theta_k = \theta_{k-1} + \dot{\theta}_{k-1}\Delta t, \tag{4.7}$$

$$\dot{\theta}_k = \dot{\theta}_{k-1} + \alpha_c\Delta t, \tag{4.8}$$

where α_c is the angular acceleration imparted by attitude control actuators.

The control algorithm outputs commands in the body frame. It is therefore necessary to rotate these commands to the LVLH frame, as described in Appendix B. To add an additional layer of realism, control actuators are modeled by applying a single pole infinite

impulse response low-pass filter to the requested acceleration commands, $\boldsymbol{F}^{req}$,

$$\boldsymbol{F}_k^{act} = \alpha \boldsymbol{F}_k^{req} + (1-\alpha)\boldsymbol{F}_{k-1}^{act}, \tag{4.9}$$

where α is a decay parameter, and $0 \leq \alpha \leq 1$ for unity DC gain. α is related to the time constant, τ of the filter (the number of samples for the system to decay to 36.8% of its original value) [137],

$$\alpha = e^{-1/\tau}. \tag{4.10}$$

For a sample time of 0.1 seconds (the typical loop rate utilized herein), $\tau = 3$ is thought to conservatively model the delay associated with spacecraft thrusters. Thus, $\alpha = 0.7165$ is used for this work.

4.2.2 Control Law Modification for Implementation with CASpR.

The Image-Based Visual Servoing (IBVS) control law introduced in Chapter III was designed to control 6DOF maneuvers. As such, the interaction matrix, $\boldsymbol{L}_x$, used for that implementation (equation (2.44)) relates the 6DOF relative velocity of the camera and target to optical flow of target feature points. CASpR, however, is limited to 4DOF motion, so using $\boldsymbol{L}_x$ as previously developed will not yield velocity commands that completely drive the chaser to its goal pose, since two of the attitude states are uncontrollable. Therefore, it is necessary to truncate the interaction matrix for use with CASpR's 4DOF system. This is done by simply removing the columns of the $\boldsymbol{L}_x$ associated with the uncontrollable DOF (ω_x and ω_z in the camera frame). For a single image feature, the interaction matrix relationship

is then:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & -(1+x^2) \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & -xy \end{bmatrix}}_{\boldsymbol{L}_x} \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_y \end{bmatrix}. \tag{4.11}$$

4.2.3 Practical Implementation of the Sensing Subsystem.

As detailed in Section 3.2.2, the role of the sensing subsystem is to produce image and depth measurements for every target feature point identified in each video frame during maneuvering. Because the simulation used in the previous chapter did not implement actual computer vision algorithms (rather, it approximated the stochastic nature of feature tracking via a number of mechanisms, as discussed in Section A.1.2.2), it did not completely address in detail how feature tracking could be implemented in reality. This section proposes a practical image processing pipeline for identifying and tracking target feature points during maneuvering using a stereo camera. Results presented in this chapter demonstrate that this proposed method is effective for the laboratory experiments examined in this work.

Identifying and tracking image features over the course of a complex approach maneuver in an accurate, robust, and computationally efficient manner is a challenging computer vision task. Two separate feature point identification and tracking methods were introduced in Section 2.4.3: image Feature Extraction and Matching (FEM) (Section 2.4.3.3) and feature tracking using the Kanade-Lucas-Tomasi (KLT) optical flow algorithm (Section 2.4.3.4). This work proposes combining these two methods via a bank of Kalman filters in an effort to ensure both robustness and computational efficiency. Figure 4.4 shows a flowchart of the pipeline.

The pipeline is composed of several subroutines, indicated by the dashed rectangles. These are: maneuver initialization, feature tracking via sparse KLT Optical Flow, feature update/re-initialization via FEM, stereo depth measurement, and Kalman filtering. In the

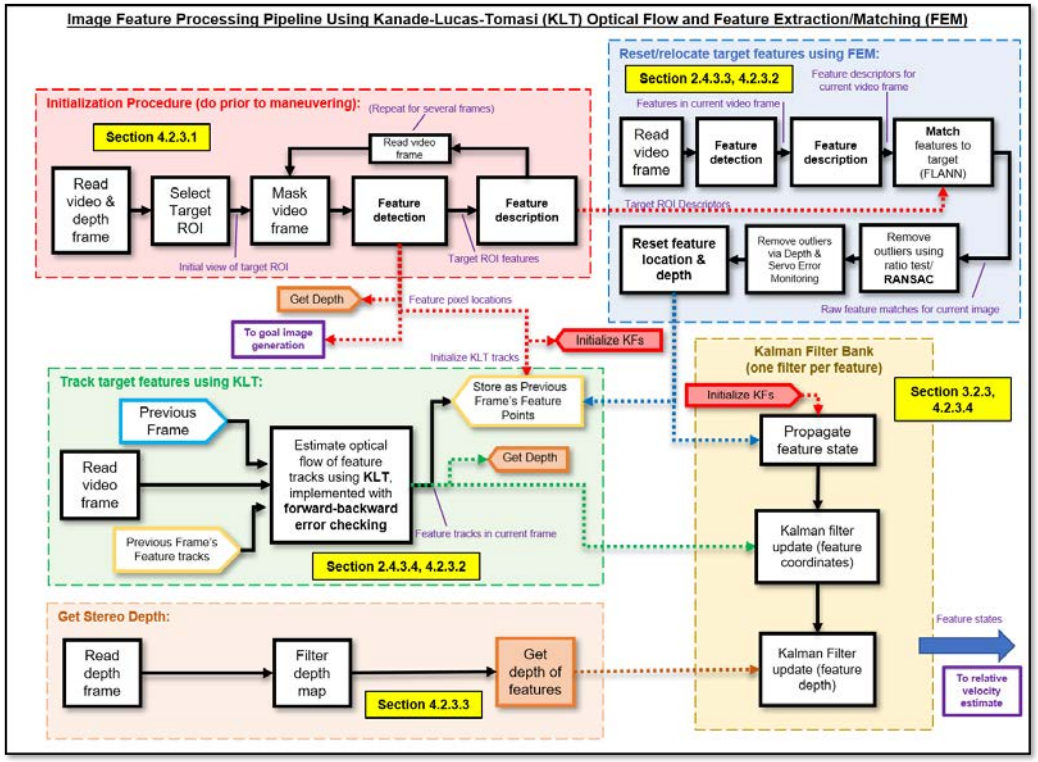

Figure 4.4: Image feature processing pipeline

previous chapter, the Kalman filter bank was discussed with the navigation subsystem, but is included here to show how these subroutines are associated. These subroutines are discussed in the following subsections.

4.2.3.1 Maneuver Initialization.

Prior to maneuvering, a region of interest (ROI) on the target that will be the objective of the final approach maneuver is manually selected. Considerations for this selection process are discussed in Section 3.2.1. This subsection addresses how features are identified within the target ROI after it has been selected. Here it is assumed that an image frame from the chaser's camera sensor captured from the initial relative pose is available to a human operator prior to the start of maneuvering. The operator specifies a quadrangle-shaped ROI on the target by manually selecting four ROI corners in this initial image.

In order to identify a rich set of target features, multiple image frames are used to identify features present within the selected ROI. These image frames are captured one after another and portray the same view of the target (assuming there is no rapid relative motion between chaser and target). Each frame will have differences imperceptible to the human eye as a result of measurement noise associated with the camera sensor. Accordingly, using multiple frames gives the feature detection algorithm several opportunities to identify new features under slightly different noise conditions.

In order to only locate features within the ROI, an image mask is used to restrict the feature detection algorithm to only check the portion of the image containing the ROI. For the initial pass of this process, the mask is a binary image, with white section corresponding to the ROI. For each subsequent frame of this process, a small area around each feature identified in previous frames is also masked. This is done to ensure that feature identifications are not duplicated, and features are spread out over the entire ROI. Figure 4.5 shows an example sequence of image masks used for this procedure.

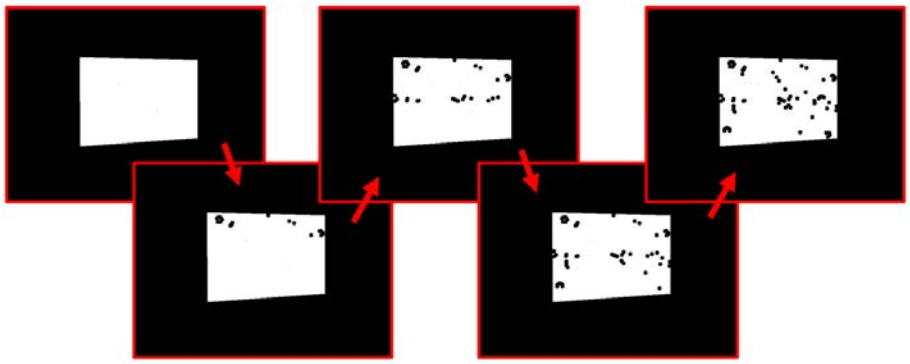

Figure 4.5: Example sequence of image masks used to identify features in the target ROI

Adjustable parameters for this initialization procedure include: the number of frames this process is repeated for, the max number of features that can be detected in each frame, the radius of the area to mask off for each feature in subsequent frames, and all

of the parameters associated with the chosen feature detector/descriptor. There are several trade-offs to consider for these parameters. If too many frames are processed, relative motion could make locations of features detected early in the process inaccurate. If the feature mask radius is set to be too large, very few features will be identified, and feature descriptors will be poor. If too many features are allowed to be identified, increased computational requirements to track these features may overburden the flight processor. Parameter selections for the results of this work are outlined in Section 4.5.

After features are identified in the target ROI, descriptors for these features are generated, the depth to each feature is recorded using the aligned depth map, features are warped to a generate goal image, as described in Section 3.3, and the feature locations/depth are used to initialize the bank of Kalman filters.

4.2.3.2 Filtering Stereo Depth Maps.

Raw depth maps are prone to contain significant measurement noise as a result of the stereo matching process, which directly affects the quality of depth measurements for feature points on the target. Fortunately, there are several methods available to filter depth maps, thus providing more consistent depth measurements to features over the course of a maneuver.

In their white paper for Intel® RealSense™ cameras, Grunnet-Jepsen and Tong discuss a recommended processing pipeline for filtering stereo depth maps [138]. As this research effort utilizes a RealSense™ camera, the pipeline is implemented as described by Grunnet-Jepsen and Tong with filtering parameters tuned for this application. However, the methods of this pipeline should apply equally well for any stereo camera setup. The pipeline is shown in Figure 4.6 and is summarized below.

The first step in the pipeline is to apply a decimation filter, which reduces the resolution of the depth map and therefore its complexity. This can be done by simply taking every nth pixel of the depth map, or Grunnet-Jepsen and Tong describe an intelligent

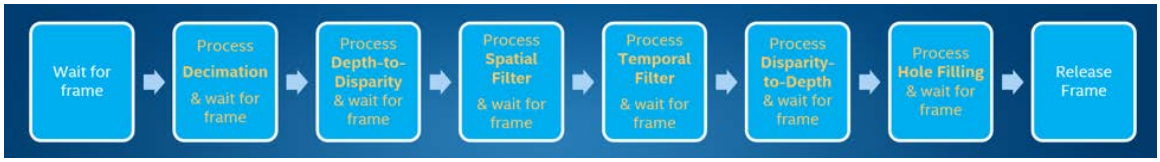

Figure 4.6: Depth map filtering pipeline [138]

method to perform sub-sampling for decimation. The effect of this filter is to significantly accelerate subsequent processing, and to additionally provide some basic smoothing and hole-filling of the depth map. How much decimation to apply is dependent both on how much processing power is available and on how much depth resolution is required.

Before spatial and temporal filters are applied to the depth map, Grunnet-Jepsen and Tong recommend transforming the map from the depth domain to the disparity domain. In the depth domain, each pixel encodes the depth to the corresponding point in space, whereas in the disparity domain, each pixel encodes the pixel disparity between corresponding points in the left and right image of the camera.

Next, spatial filtering of the depth map is applied using an edge-preserving filter based on the method proposed by Gastal and Oliveira [139]. This simplified implementation essentially applies multiple passes of a 1D exponential moving average filter vertically and horizontally over each pixel of the depth map, except for pixels where the difference in disparity from the previous pixel is greater than some threshold (i.e. edges containing instantaneous jumps in depth). The tuning parameters for this filter include the number of passes to conduct, the decay parameter for the moving average filter, and a distance threshold used to determine what is an edge.

A temporal filter is then used to additionally smooth the depth map. This filter is applied in much the same way as the spatial filter, except instead of being applied across different pixels in the same map, it is applied to the same pixel across depth maps captured

in sequence. As with the spatial filter, it utilizes an exponential moving average filter with a single decay parameter, and a distance threshold that determines how edges are preserved.

Finally, after performing a transformation back to the depth domain, a hole-filling filter can be optionally applied to estimate depth for pixels without depth information. There are a number of ways to do this, though all methods rely on using information from surrounding pixels to estimate depth for pixels without depth information. The danger of doing this is that a very incorrect depth could be applied to the pixel, especially if it is located near an edge. For this reason, depending on the nature of the target surface, it may be best to not apply a hole-filling filter, and rather just reject/not update features for pixel locations with no depth information.

4.2.3.3 Feature Point Tracking using KLT and FEM.

Tracking of feature points is achieved using two separate methods: KLT (see Section 2.4.3.4) and FEM (see Section 2.4.3.3). Information about the state of tracked target feature points from these two methods is fused via the Kalman filter bank. The methods are both utilized in order to mitigate the relative disadvantages associated with implementing each method individually. Since the KLT method is computationally faster than FEM, it is implemented for every image frame and is therefore the primary feature tracking mechanism. FEM is implemented periodically (since calling it every image frame could be too computationally expensive), and it acts to both refine locations of tracked features and re-initialize lost features.

The KLT method implemented for this pipeline is a modified version of the original method introduced by Kanade, Lucas, and Tomasi. A sparse iterative pyramidal implementation of the basic version of KLT is used, as proposed in [140]. This version of KLT implements a set of image pyramids, and searches the coarsest level for feature matches first. As a result, it is more computationally efficient and can track features moving farther between image frames.

Additionally, forward-backward error checking is implemented as a secondary mechanism to detect tracking failures, thus making the method more robust. Proposed in [141], this check works by calculating the optical flow of features from the previous frame to the current frame, then again calculating the optical flow from the current frame backwards to the previous frame. If the distance between the feature location in the previous frame and the KLT backward estimate of the feature location is greater than a threshold, the track for that feature is deemed to have failed. While this method requires two calls of KLT per image frame, it significantly reduces the number of erroneous feature tracks.

KLT is not used exclusively for two reasons. First, in the event a feature track is lost, it has no way to reinitialize that track. In this way, the number of features tracked by KLT is monotonically decreasing over the course of the maneuver, unless new feature tracks are added by a separate method. Identifying new features mid-maneuver is not straight-forward for this use-case, as the homography warping used to determine the goal locations of each feature is only defined for the initial pose. Augmenting this pipeline in order to identify new features is worthy of future research, as discussed in Section 4.4.1.

Secondly, using KLT exclusively, poor feature tracks may become insidious outliers over the course of a maneuver. Because KLT only searches a local region around a feature, the track of a feature identified for example on an edge may slide along that edge as the scene changes, thus insidiously deviating from its true location. Here FEM often does a better job identifying the true location of a feature as it uses more of the image to match features.

FEM is implemented periodically in an effort to refine feature locations and reinitialize lost tracks. FEM is not implemented exclusively because it is computationally expensive and it typically does not identify every tracked feature in every image frame (leading to more jagged feature trajectories). Two layers of outlier rejection are used to robustly

remove incorrect feature matches. The first layer implements Lowe's ratio test and RANSAC, as discussed in Section 2.4.3.3. The second layer implements servo error monitoring and is discussed in the next section. Features matches generated by the FEM process that pass all outlier rejection steps are used to reset associated KLT feature tracks and position states of corresponding Kalman filters.

4.2.3.4 Kalman Filter Bank and Feature Health Vector.

The use of a bank of Kalman filters and a health vector to develop state estimates of target feature points was introduced in Section 3.2.3. This section provides more detail on the functions of this bank and vector. The primary functions of the bank of filters and health vector are to:

1. Fuse feature location measurements from the KLT tracker and FEM methods.
2. Reject feature measurement outliers.
3. Smooth feature pixel trajectories.
4. Estimate feature pixel velocities.
5. Keep track of which features are 'healthy.'

The first function pertains to updating the estimated location of a feature given a measurement from either the KLT tracker or FEM method. If a KLT feature track exists for a given feature, then each timestep a Kalman update is performed using the tracked position of the feature as a measurement. Established models for the measurement noise associated with KLT are not available, so in lieu of additional experimentation to quantify these noise models, the covariance matrices for these measurements are considered tuning parameters.

It may seem tempting to perform a Kalman update when FEM identifies a feature in the current image frame in the same manner as KLT, but doing so would create the

possibility that the feature location is updated to an erroneous location in between the current estimate of the position and the FEM measurement of the position. Instead of performing a Kalman update, if an FEM measurement is determined not to be an outlier via the methods discussed previously, the estimated location (position state) in the Kalman filter is simply reset with this measurement. Velocity states and covariance remain unchanged, as they will be quickly corrected during the next Kalman update using the KLT track measurement. If a valid measurement of depth is not available for the feature, the feature is not reset.

Mahalanobis distance monitoring is implemented as a final layer of outlier rejection for each feature measurement, as described in Section 3.2.3. Since this rejection method is only implemented in the update step, it is only applicable to the KLT-based measurements.

Being a state estimator, each Kalman filter also works to smooth the pixel location measurements and estimate the pixel velocities (recall these velocities are needed to estimate 6DOF relative velocity between chaser and target). The pixel dynamics model is simply a double integrator, so the amount of smoothing that is applied is determined by the filter process and measurement covariance tuning. As the filter acts a causal low-pass filter, the amount of smoothing that is applied is traded off with the induced delay in the state estimate. In practice, given the comparatively slow dynamics associated with spacecraft rendezvous and proximity operations (RPO), a fair amount of smoothing can be applied since the associated delay is not consequential.

Finally, a health vector is utilized to keep track of which features have been identified in the current image frame. That is, features that actively have a KLT track associated with them or have been reinitialized via FEM are considered 'healthy' in the sense that they can be used for navigation and control functions. Thus, a vector of the health values (where a value of 1 denotes a healthy feature and 0 a feature that is not presently being tracked) associated with each feature is updated each time step and used to determine which features

are used for relative velocity and IBVS calculations for that time step. Kalman filters for features without active KLT tracks are still propagated. Without measurements, however, state estimates for these features degrade quickly, making them unusable for navigation and control.

4.2.3.5 Feature Outlier Rejection via Servo Error Monitoring.

This section presents a method to identify outliers using the servo error (Equation (3.9)) calculated for all tracked features. The method works by comparing servo error for individual features with the mean servo error for all tracked features. Doing this effectively compares where a target feature is expected to be at any given time to where it actually is—if the location of a feature deviates too far from its expected location, it is considered an outlier and is rejected.

In simulation and experimentation with CASpR, it was observed that outliers in feature state estimates can have a disastrous effect on the quality of a maneuver. While a single outlier is not always catastrophic since navigation estimates and control commands are computed using a least-squares solution, significant or multiple outliers can significantly degrade this solution. Unfortunately using Lowe's ratio test and RANSAC alone does not remove all outliers associated with FEM.

Feature measurement outliers can broadly be divided into two categories: insidious outliers that occur due to KLT tracks wandering from a feature's actual location, and features outliers occurring due to being incorrectly identified in the wrong location during the FEM process. FEM is meant to help mitigate insidious outliers caused by KLT, and Lowe's ratio test coupled with RANSAC is intended to reduce the number of FEM matching errors. However, even with these methods, outlier measurements can sometimes be incorporated in feature state estimates.

Figure 4.7 shows servo error plots from two separate maneuvers conducted with CASpR, with feature outliers circled in red. For the servo error plots on the left, the outliers

are the result of a FEM matching error, as is apparent by the fact that the error associated with the feature outliers exhibits a sudden change from one timestep to the next. The servo error plots on the right depict insidious outliers resulting from KLT tracks erroneously sliding along an edge on the target surface. The servo error for these features does not discretely jump, but does grow to be significant over the course of the maneuver.

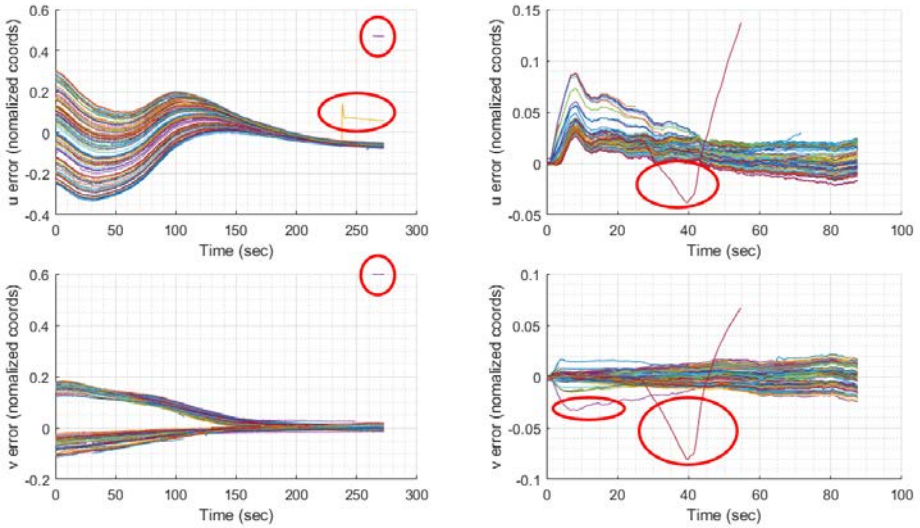

Figure 4.7: Examples of the effect of outliers on servo error. Servo error plots for two different runs are shown, with outliers circled. The run on the left contains outliers resulting from FEM matching error. The run on the right displays insidious outliers resulting from KLT tracking error.

In both of the cases shown in Figure 4.7, the feature outliers are easily identifiable as they do not follow the trend of the rest of the servo error. Outliers like these can be detected online by performing a check on the servo error for each feature prior to using that feature for navigation and control calculations. If the servo error for an individual feature is determined to be too far from the mean servo error, then that feature is considered an outlier. The threshold for rejection is defined as a multiple of the standard deviation of servo error for all features for the current time step plus the mean servo error. This threshold is defined separately for the horizontal or vertical direction. If a feature is determined to be an outlier,

the health vector is updated to reflect this feature as unhealthy, meaning it is not used for navigation and control, and its KLT track is deleted.

In practice, two separate instances of servo error monitoring are implemented in the navigation and control loop. The first instance occurs during the FEM subroutine on the image processing pipeline, as shown in Figure 4.4. This instance of monitoring is effective for rejecting FEM-associated outlier measurements before they are used to reset feature locations. The second instance of monitoring is implemented during construction of the interaction matrix for the current timestep (see Section 3.2.3, Equation (3.4)). Here the servo error for each tracked feature is compared to the dynamic threshold calculated using the servo error for all features from the previous timestep. This instance of servo error monitoring effectively rejects insidious outliers, as the servo error for every tracked feature is verified every timestep.

The threshold for determining outliers using servo error is a tuneable parameter—if an assumption is made that servo error is distributed normally, then a convenient way to determine an appropriate threshold is to use the empirical rule of standard deviations, which states that about 68% of samples should fall within 1 standard deviation of the mean, 95% within 2 standard deviations, and so on. For this work, the servo error monitoring threshold used during the FEM subroutine is set to 2 standard deviations plus the mean servo error, and the threshold used for servo error monitoring during velocity command calculation is set to 3 standard deviations plus the mean. These thresholds are shown graphically on the servo error plots presented in Section 4.3.

As will be shown in Section 4.3, servo error monitoring has been observed to be very effective at preventing both insidious and FEM outliers from significantly affecting the outcome of the maneuver.

4.2.4 Pose-Hold Mode.

A critical function of GNC algorithms for autonomous spacecraft RPO pertains to the ability to recognize and prevent potentially catastrophic situations. Without a doubt the most catastrophic situation that can occur during RPO is collision of spacecraft, which will likely lead to destruction of one or both spacecraft, and possibly generate space debris that will pose a significant risk to RSOs in nearby orbits. The ability to recognize impending failure of a final approach maneuver is so important that it is stated in Chapter I as a distinct task of Research Question 3. This section addresses that task, with a focus on how the GNC method examined in this chapter is augmented to automatically revert to a failsafe mode if conditions degrade to a point that a successful maneuver cannot be guaranteed. If criteria are met to activate this mode, the chaser will cancel its current course of maneuver and immediately act to maintain its current relative pose to the target. This 'pose-hold' mode can also be used intentionally to maintain a precise relative pose e.g. for station keeping or formation flying maneuvers.

When used as a failsafe, there are a number of criteria worth considering that can be used to activate this mode. For example, if the relative velocity along the chaser's optical axis exceeds a preset threshold, pose-hold mode could be triggered. Or a method could be implemented whereupon the mode is initiated if the target ROI gets too close to an edge of the image plane. One criteria that should always be implemented involves initiating the mode if the number of healthy target features falls below a minimum threshold. This would be indicative of impending failure of the final approach maneuver, since both estimation of relative velocity and generation of control commands rely on least-squares computations using tracked target features.

The ability of IBVS to operate directly using image features makes implementation of this mode straight-forward. In fact, the only difference between operation of this mode and the normal method is in the initialization procedure of the sensing subsystem. Figure 4.8

shows this procedure; it is a drop-in replacement for the initialization procedure in Figure 4.4.

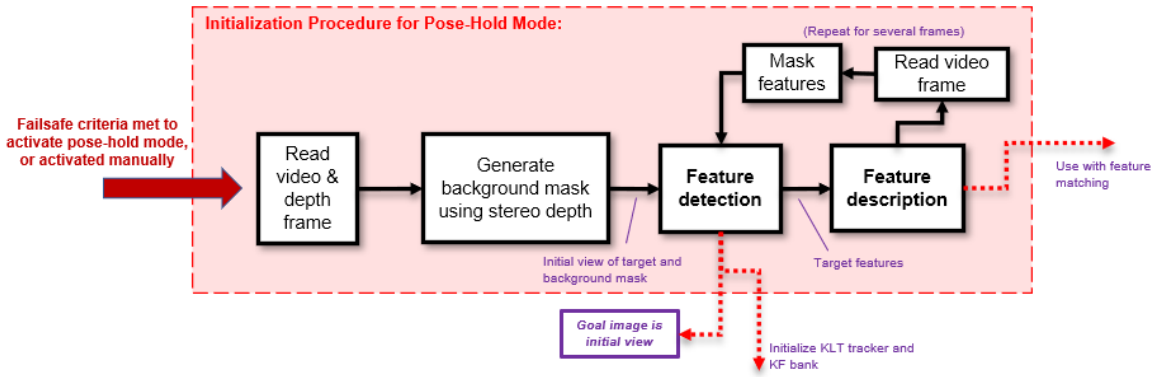

Figure 4.8: Pose-hold initialization procedure (replaces initialization procedure in Figure 4.4)

Upon entering this mode, a new maneuver is commanded to maintain the current relative pose. Instead of a remote operator selecting the ROI on the target as previously described, the entire portion of the target that is visible in the image frame is used as the ROI for maneuvering. This maximizes the opportunity for target features to be identified. Doing this is not as simple as using the entire current image however; as if the Earth, Moon, or stars can be seen in the background of the image, features will be identified on these bodies. Instead, the background behind the target is subtracted using an aligned stereo depth map. This is accomplished by generating a feature detection mask using pixels in the camera image with corresponding depths beyond a distance threshold (for example beyond the maximum distance that can be estimated by the stereo sensor) to 0. The background mask and initial image are passed to a feature detector and a set of target features are identified as before. Beyond this procedure, the framework operates as described previously, with the exception of how the goal image is generated.

As this mode assumes the current pose is the desired pose, no warping is necessary to generate goal positions for each target feature. Rather, the goal positions for each feature are taken as their initial position when the pose-hold mode was activated.

Finally, it is worth noting that because no relative motion should occur with this mode activated, it is possible to identify new target features during the course of operation (as opposed to waiting for the number of features to fall below the failsafe threshold and restarting a new pose-hold maneuver).

4.3 Results

As with the previous chapter, results for this chapter are presented via a set of scenarios, with the intent to demonstrate the efficacy of the method under a variety of conditions representative of possible final approach maneuvers. The simulation results of Chapter III indicated that performance of the method was affected by how planar the ROI was. Accordingly, the first three scenarios presented, 4P1-4P3, consider a planar target, and the last three scenarios, 4NP1-4NP3, consider a target with significant nonplanar protrusions.

4.3.1 Maneuvering with Respect to a Planar (P) ROI.

Scenarios 4P1-4P3 consider a planar target ROI (shown in Figure 4.9). The target surface is not based on any particular spacecraft or object. This surface was chosen because it lends itself to identification of numerous features. Since features can be readily identified on this surface, the number of features used for maneuvering can be easily controlled by limiting the maximum number of features that can be identified during the initialization procedure of the sensing subsystem. This facilitates examination of the affect that the number of tracked features has on the quality of the final approach maneuver.

4.3.1.1 Scenario 4P1: Simple Maneuver with Static Planar Target.

The first scenario considers a simple maneuver: a nearly on-axis approach requiring the chaser to move toward the target ROI. The intent of this scenario is to examine

performance of the method for a comparatively simple maneuver and generate a baseline of performance for comparison with subsequent scenarios. The chaser is initially located at $[-0.13, -1.70, -0.10]^T$ meters with respect to the LVLH frame (LVLH origin at target ROI centroid), with its camera boresight pointing at the target ROI centroid (rotated $-6.46°$ about the LVLH Z-axis). The target ROI and goal image corners are selected such that large translation in the LVLH positive Y direction and slight translations in the positive Z and X directions are required to achieve the desired goal pose. These goal corners implicitly define a goal pose that should place the chaser camera about 0.5 meters from the target ROI centroid, rotated such that the camera boresight is orthogonal to the ROI surface (i.e. $0°$ rotation about the LVLH Z-axis). Figure 4.9 displays the initial and final views from the chaser camera. The red cross in the center of the image denotes the camera's boresight, and the other four crosses denote the goal locations of the ROI corners.

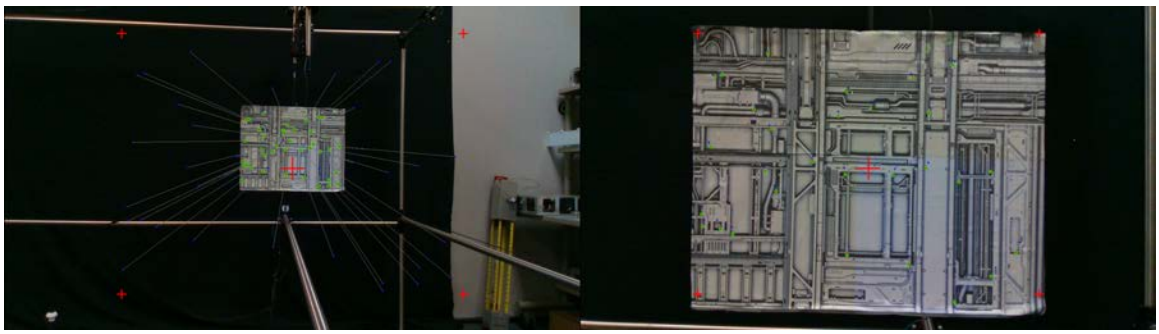

Figure 4.9: Scenario 4P1: Initial and final view from chaser's camera

Figure 4.10 displays the chaser's trajectory in the LVLH frame, with the target ROI centroid (red triangle), and ROI's surface plane normal vector (red arrow) overlaid. The chaseer smoothly captures the goal pose, ending the maneuver at an LVLH position of $[0.00, -0.49, 0.01]^T$ meters with $0.21°$ rotation about the LVLH Z-axis.

Figure 4.11 shows the servo error for all tracked target features (in normalized image coordinates), as well as the average servo error for each time step during the maneuver.

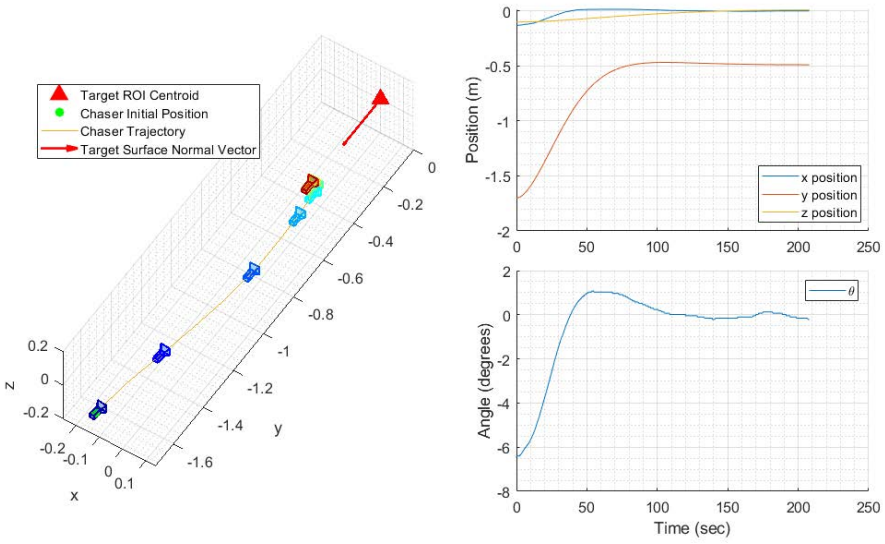

Figure 4.10: Scenario 4P1: Chaser trajectory (LVLH frame)

The servo error collapses smoothly, though error does not reach zero for every feature. The standard deviation of servo error at the end of the maneuver is 0.0054 normalized pixels in the horizontal (u) direction and 0.0061 normalized pixels in the vertical (v) direction. The failure to reach zero servo error is expected and assessed to be due to (1) incorrect generation of feature goal locations via the warp method, and (2) measurement error associated with tracking the locations of each feature.

Figure 4.12 displays the number of healthy features tracked via KLT for each timestep of the maneuver. The number of features initially identified is manually capped at 40 features in order to examine the method under sub-optimal conditions. Of note, the number of features is not monotonically decreasing, due to the use of FEM (implemented every 20 video frames) to re-identify features after they have been lost. In total, 108 features re-initializations occur over the course of the maneuver. The plot also shows the raw number of FEM matches, the number of FEM updates (those matches that make it though all outlier rejection methods), and the number of FEM rejections/feature tracks deleted as a result of servo error monitoring.

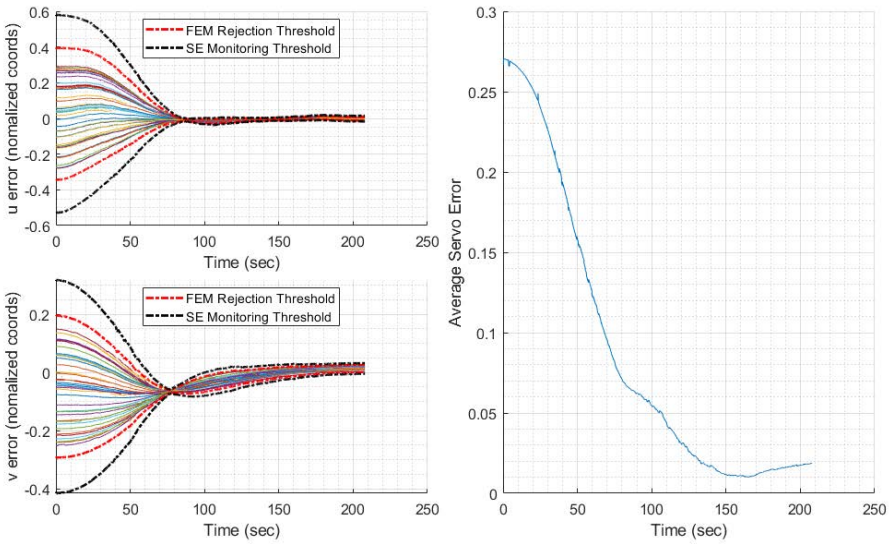

Figure 4.11: Scenario 4P1: Servo error

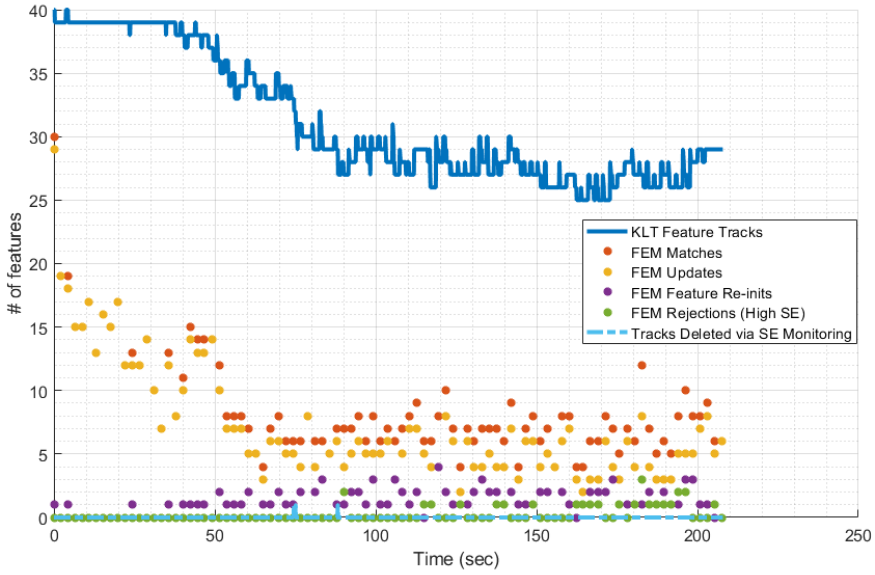

Figure 4.12: Scenario 4P1: Number of features tracked throughout maneuver

Recall that this work assumes that the only relative navigation sensor available is a stereo camera sensor, meaning this method relies exclusively on information derived from this sensor (target features) to generate navigation estimates and control commands.

Specifically, the estimate of relative velocity between chaser and target is derived from optical flow of target features and is used by both the navigation and control subsystems. Figure 4.13 shows the relative velocity estimate for each maneuvering degree of freedom (rotated into the LVLH frame to facilitate comparison), overlaid with the derived actual velocity (see Section 4.2.1.1 for description of this value). Though the relative velocity estimate is noisy (as it was in simulation), it tracks the derived actual well enough to affect a successful maneuver.

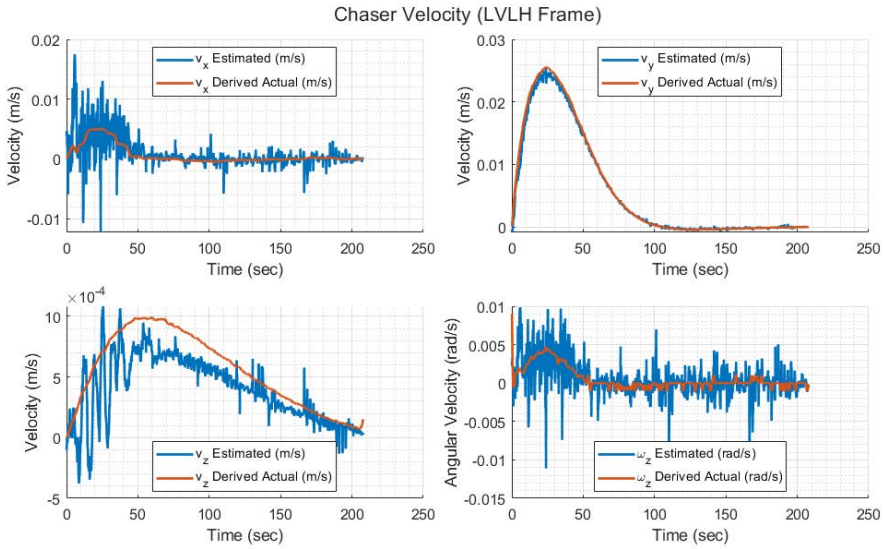

Figure 4.13: Scenario 4P1: Chaser velocity with respect to LVLH frame (estimated, commanded, derived actual)

Overall, this scenario's results indicate proper functionality of each component of the architecture. The computer vision methods and sensing subsystem are sufficient in this case to produce a high quality relative velocity estimate and in turn a satisfactory chaser trajectory. The results are largely comparable to those produced in simulation, and further demonstrate the potential of the method for this type of final approach maneuver.

4.3.1.2 Scenario 4P2: Complex Maneuver with Static Planar Target.

Scenario 4P2 examines a more complex maneuver with the same static planar target ROI as 4P1, this time with chaser beginning the maneuver with significantly more rotation relative to the target ROI. Figure 4.14 shows the initial and final views from the chaser's camera. The chaser is initially positioned at $[-1.23, -1.20, -0.10]^T$ meters with respect to the target/LVLH frame, pointing at the target ROI centroid ($-46.46°$ rotation about the LVLH Z-axis). For this scenario, the number of target features is capped at 20 in order to demonstrate the ability of the method to operate using very few target feature tracks.

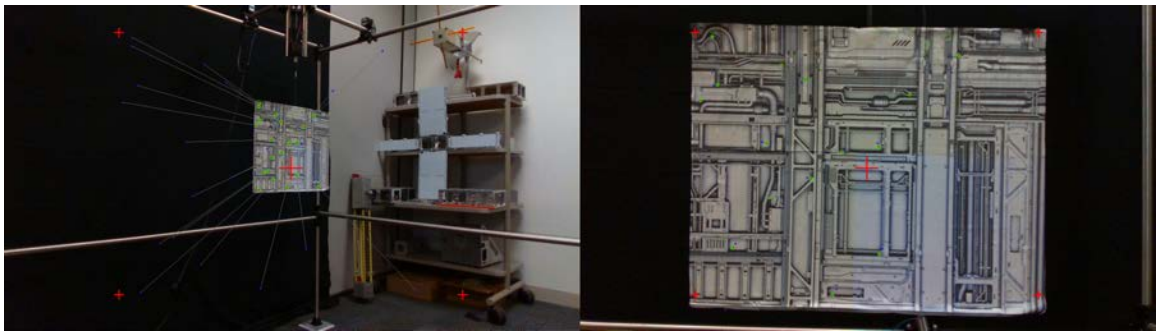

Figure 4.14: Scenario 4P2: Initial and final view from chaser's camera

The trajectory of the chaser, shown in Figure 4.15, indicates that the chaser smoothly converges on the desired goal pose while maneuvering in all 4DOF. The position of the chaser upon termination of the scenario matches nearly exactly with that of Scenario 4P1: $[0.00, 0.49, 0.01]^T$ meters in the LVLH frame, and $0.28°$ rotation about the LVLH Z-axis.

Servo error plots are shown in Figure 4.16. The initial increase in average servo error occurs as the camera initially rotates faster than it can translate to compensate. Standard deviation of servo error at termination of the maneuver is comparable to that of Scenario 4P1: 0.0067 normalized pixels in the horizontal (u) direction and 0.0062 normalized pixels in the vertical (v) direction.

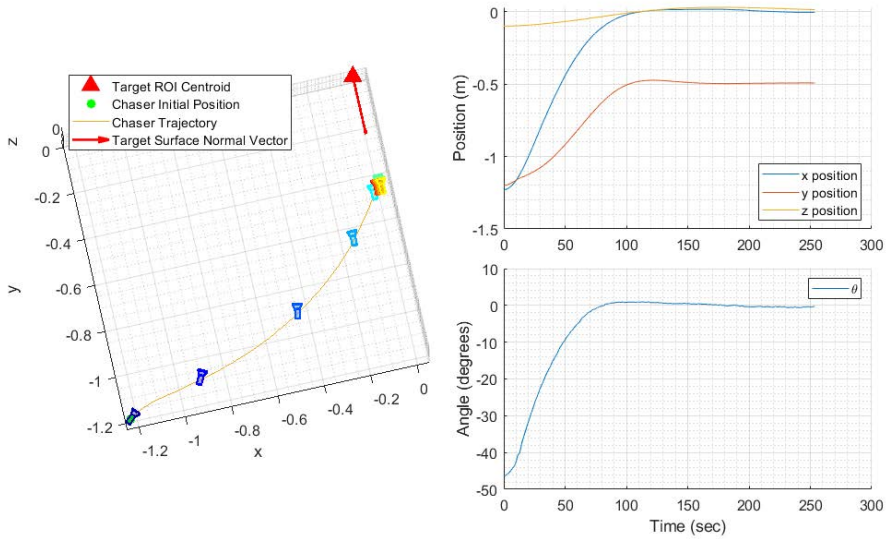

Figure 4.15: Scenario 4P2: Chaser trajectory (LVLH frame)

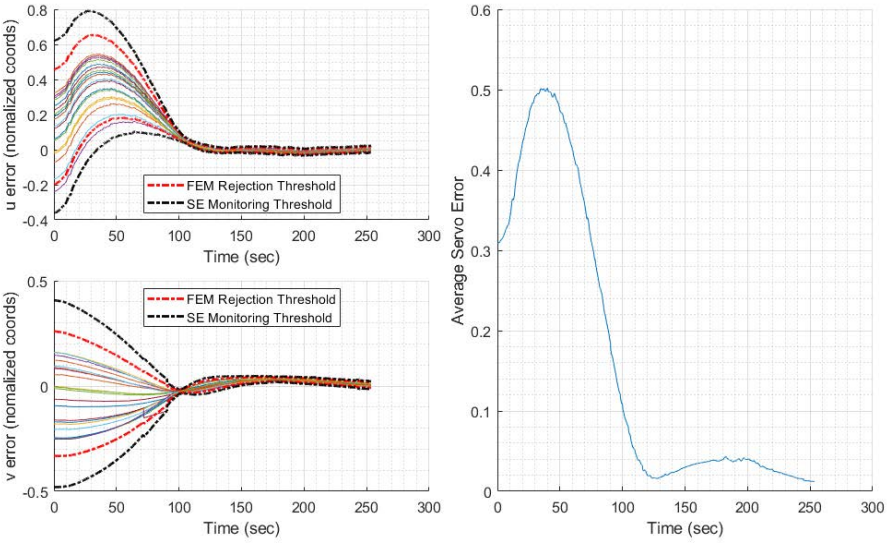

Figure 4.16: Scenario 4P2: Servo error

The quality of the relative velocity estimate is also similar to that of Scenario 4P1, and is shown in Figure 4.17. The variance of the velocity estimates decrease over the course of the maneuver as the camera nears the target, enabling it to better estimate target feature pixel velocities.

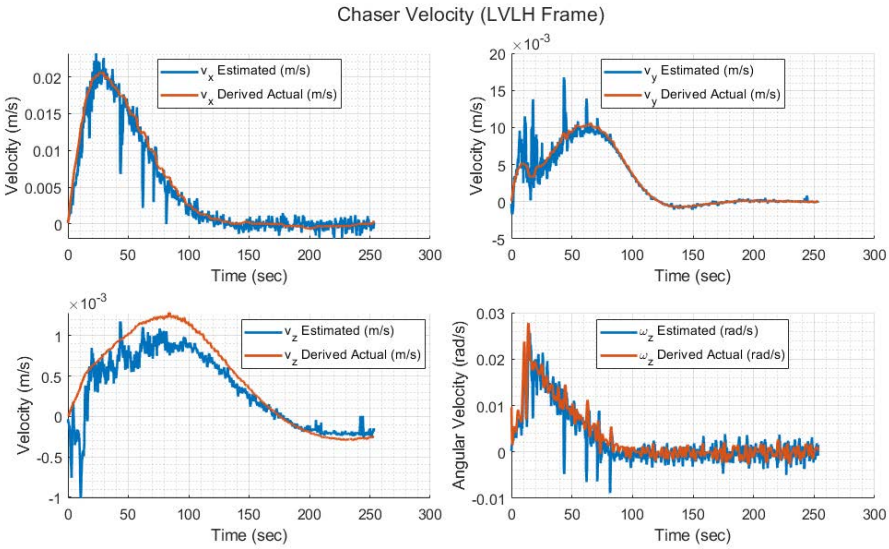

Figure 4.17: Scenario 4P2: Chaser velocity with respect to LVLH frame (estimated, commanded, derived actual)

Figure 4.18 shows example plots of the Kalman filter states for a tracked feature over the course of the maneuver and are representative of filter states for all the tracked features. The small jumps in position most notable in the horizontal (u) position are indicative FEM position resets.

This scenario corroborates the results of Scenario 4P1 and further shows that the method is capable of affecting a successful maneuver in all 4DOF available with CASpR. The success of the maneuver starting from a highly skewed initial pose indicates that the warp method of goal image generation is able to satisfactorily generate goal image coordinates for target features of a planar ROI under a variety of conditions.

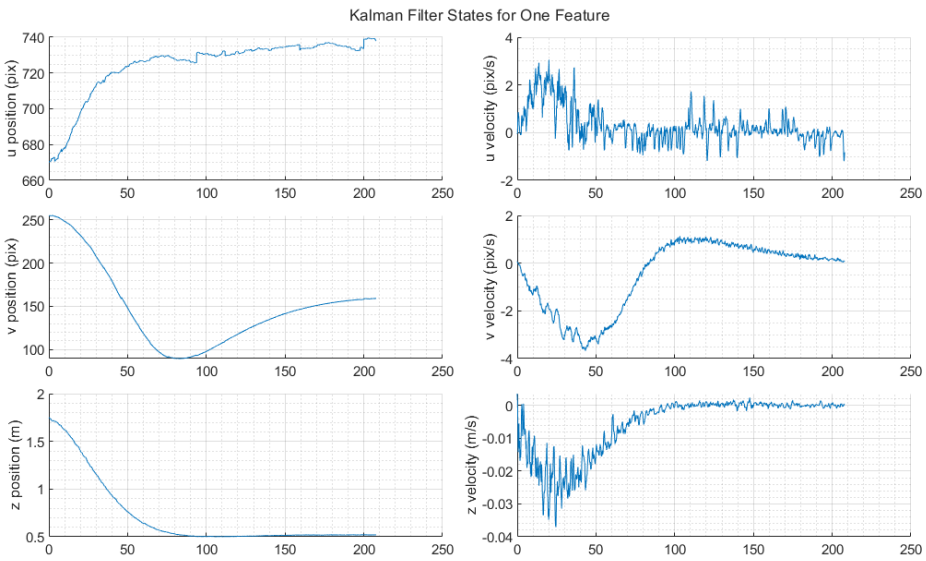

Figure 4.18: Scenario 4P2: Kalman filter states for a single tracked feature

4.3.1.3 Scenario 4P3: Complex Maneuver with Rotating Planar Target.

This scenario demonstrates an angled approach with a rotating target. The chaser is initially positioned at $[-0.57, -0.10, -0.10]^T$ meters with respect to the target/LVLH frame, pointing at the target ROI centroid (30.98° rotation about the LVLH Z-axis).The target is given an angular velocity of $0.52°/sec$ about the LVLH Z-axis at the start of the chaser's maneuver. The chaser must compensate for the target's motion as it maneuvers to achieve the same goal pose as the two previous scenarios. The target motion begins as the chaser initiates its maneuver, meaning that the Kalman filters have not yet converged on the moving target features at the start of the maneuver. As with Scenario 4P2, only 20 target features are allowed to be initially identified for this maneuver.

The trajectory of the chaser and target in the LVLH frame is shown in Figure 4.19. Colored arrows emanating from the ROI centroid indicate the surface normal vector of the ROI surface throughout the maneuver. Camera symbols (representative of the chaser's pose) with matching colors as ROI surface normal vectors correspond to the same time step. The scenario ends with the chaser in a relative pose very similar to that of the previous two

scenarios: $[0.53, 0.01, -0.01]^T$ meters with respect to the target body frame, and $-0.16°$ rotation about the target body frame/LVLH Z-axis (camera frame Y axis).

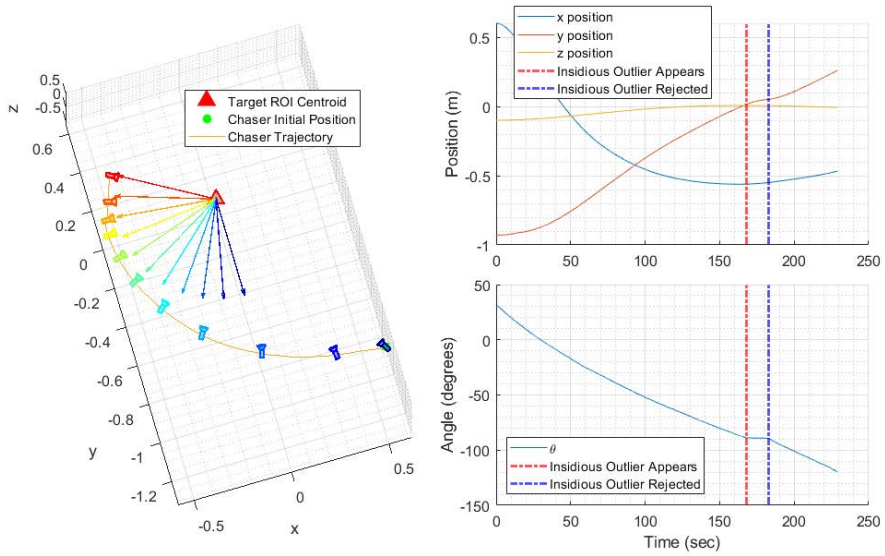

Figure 4.19: Scenario 4P3: Chaser trajectory (LVLH frame)

As with Scenarios 4P1 and 4P2, the servo error for all tracked features (shown in Figure 4.20) exhibits a smooth collapse towards zero during the maneuver. The standard deviation of error in the horizontal (u) direction (0.0066 normalized pixels) and in the vertical (v) direction (0.0064 normalized pixels) is nearly identical to that of previous scenarios, indicating that the feature tracks do not in general drift more for a moving target than for a static target (with the exception of the outlier discussed below).

The relative velocity of the chaser with respect to the target, resolved in the camera frame, is shown in Figure 4.21. Like previous scenarios, the estimate largely tracks the true relative velocity very well. However, at about 168 seconds into the maneuver, one of the target features located on the edge of the target ROI becomes an insidious outlier as its feature track catches on to an erroneous feature located behind the target. The feature

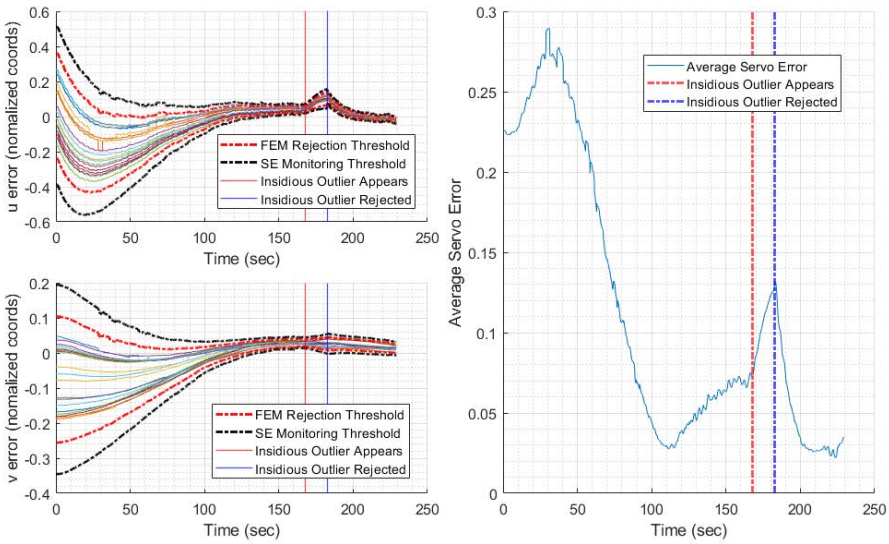

Figure 4.20: Scenario 4P3: Servo error

slowly drifts horizontally away from its true location, until it is rejected via servo error monitoring at 182 seconds into the maneuver.

The times this outlier appears and is rejected are indicated on all the plots in this subsection. As can be seen, the outlier degrades the relative velocity estimate, especially in the camera frame X direction and about the camera frame Y axis. This degraded velocity estimate has an effect on the chaser trajectory: the camera essentially stops rotating while the outlier is present. It is important to note that this outlier is able to noticeably affect the maneuver because there are only 16 features being tracked at the time it appears. If there were more healthily features being tracked while this outlier was present, then it would have less of an effect on the maneuver due to the least-squares nature of the velocity estimate and command calculations (however it is worth noting many more features would be required to overcome a single outlier, one or two would not be enough).

Figure 4.22 shows the number of features tracked over the entire maneuver, and clearly depicts that the insidious outlier is rejected at 182 seconds. The figure also shows that in the

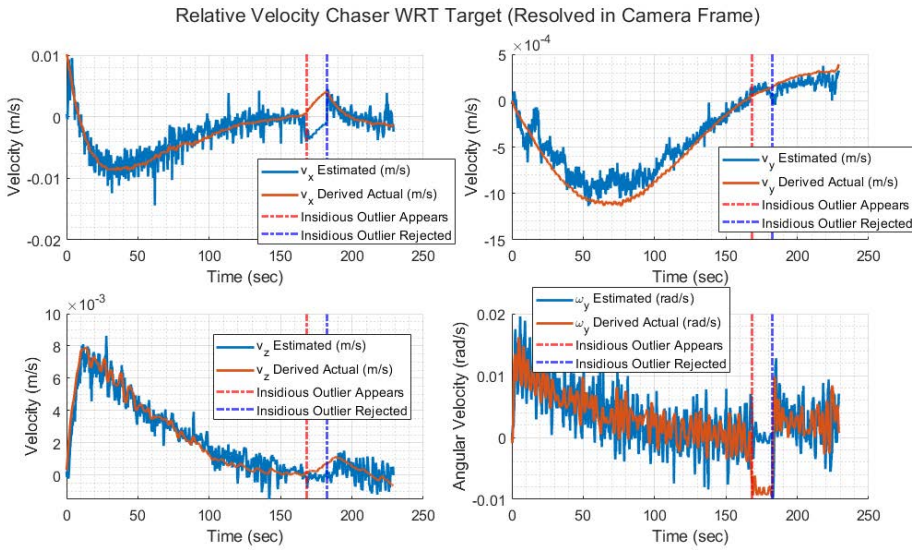

Figure 4.21: Scenario 4P3: Relative velocity of chaser with respect to target, resolved in the camera frame (estimated, derived actual)

later half of the maneuver, many features match outliers resulting from the FEM subroutine are rejected via servo error monitoring.

Overall, the scenario demonstrates the ability of the method to successfully control a maneuver with a moving planar target using less than 20 target features. Additionally, it shows the degrading effect of insidious outliers when they constitute a significant portion of the total number of features tracked. Importantly, it confirms the ability of servo error monitoring to identify and reject such insidious outliers.

4.3.2 Maneuvering with Respect to a Nonplanar (NP) ROI.

The results presented above for a planar target ROI are promising. However in practice the target ROI may contain significant nonplanar features, such as antennas, thrusters, fixtures, etc. The simulation results of Chapter III indicate that using this method with a nonplanar ROI made the chaser less likely to achieve the desired goal pose implicitly defined by a goal image generated using the warp method. Specifically, when the target ROI contains significant nonplanar features, the chaser fails to achieve the goal attitude of

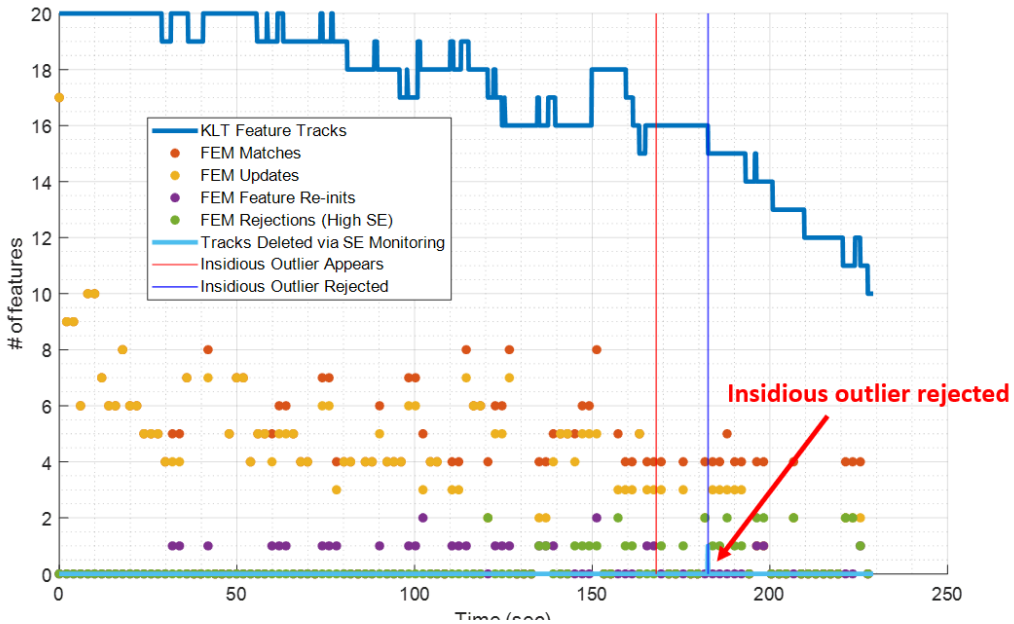

Figure 4.22: Scenario 4P3: Number of features tracked throughout maneuver with annotation showing where insidious feature outlier is deleted

being orthogonal to the predominant surface of the ROI. This was determined to be a result of the way goal positions for the features were generated using a homography warping, which relies on an assumption that target features lie on a plane.

Figure 4.23 displays the nonplanar target used for Scenarios 4NP1, 4NP2, and 4NP3: a mock-up resembling the Canadarm grapple fixture commonly used for docking with the International Space Station. For the actual case of docking with this fixture, the fixture is considered a known target—therefore established navigation and control methods are able to be implemented due to the well known nature and location of the fixture with respect to the space station. This work pertains to unknown targets however, so it is assumed that nothing is known about the target prior to initiating the final approach maneuver. This design was chosen for this work because it is could be representative of possible target ROIs containing nonplanar features, and it is considered a challenging target ROI due to its significant protrusions.

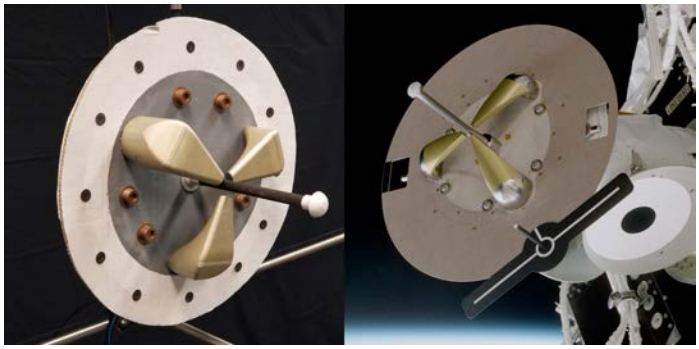

Figure 4.23: Canadarm grapple fixture used as target for nonplanar scenarios. Mockup used for this research effort shown on left, actual image of grapple fixture shown on right [142].

4.3.2.1 Scenario 4NP1: Simple Maneuver with Nonplanar Target.

Scenario 4NP1 is similar to Scenario 4P1, with the exception that a nonplanar target ROI is now the subject of the maneuver. Figure 4.24 displays the initial and final chaser camera images. The chaser's initial position in the LVLH frame is $[-0.10, 1.70, -0.10]^T$ meters with its camera boresight rotated $-4.95°$ about the LVLH Z-axis, pointing at the target ROI centroid. The intent of the maneuver is to approach the target along the axis orthogonal to the predominant surface plane of the target ROI. The goal corners are selected such that the desired pose equates to a relative position approximately 0.6 meters away from the predominant target surface.

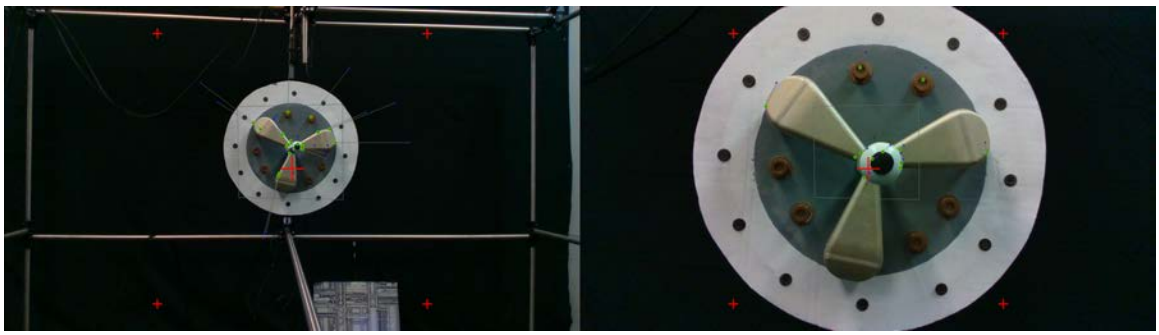

Figure 4.24: Scenario 4NP1: Initial and final view from chaser's camera

Figure 4.25 shows the chaser's trajectory in the LVLH frame. The final position of the chaser is $[-0.02, 0.66, -0.1]^T$ meters with respect to the LVLH/target body frame and 2.48° degrees rotation about the LVLH frame Z-axis. Compared to Scenario 4P1, the chaser ends the maneuver with 2.27° more rotation relative to the axis orthogonal to the target, and the chaser's final position is within about 0.06 meters of of the desired relative pose implicitly defined by the generated goal image.

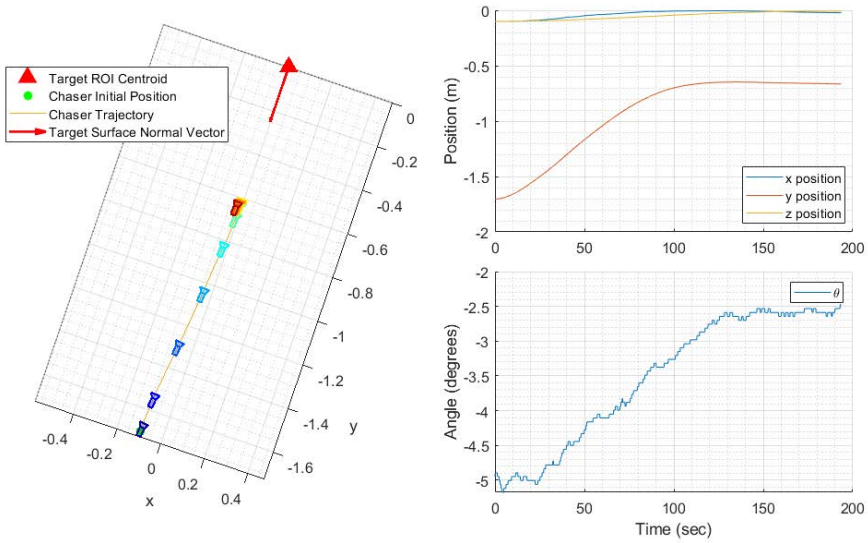

Figure 4.25: Scenario 4NP1: Chaser trajectory (LVLH frame)

The servo error for all tracked features is shown in Figure 4.26. While the error does collapse over the course of the maneuver, it is clear that the variation in error as the chaser achieves its goal pose is larger as compared to maneuvers with planar targets. The standard deviations in the horizontal (u) and vertical (v) directions at the end of the maneuver are 0.0095 and 0.0146 normalized pixels, respectively. These deviations are roughly 1.4 times larger in the horizontal direction and 2.3 times larger in the vertical direction as compared to Scenario 4P1. The increased deviation is expected, as the warp method for goal image generation fails to correctly predict feature goal locations for nonplanar surfaces. However,

in spite of this large steady state servo error, the chaser is similarly able to achieve the desired relative pose as compared to Scenario 4P1.

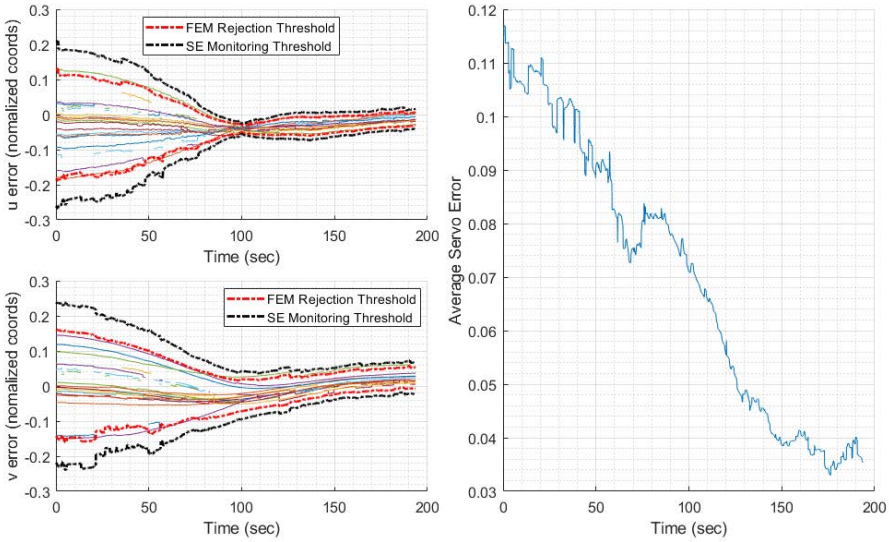

Figure 4.26: Scenario 4NP1: Servo error

Figure 4.27 shows the chaser's estimated and derived actual with respect to the LVLH frame. These plots strongly resemble those of Scenario 4P1, once again indicating that the method is able to sufficiently estimate relative velocity for this type of approach.

Given how similar the results of this scenario are compared to Scenario 4P1, it is apparent that a nearly straight-on approach maneuver with a nonplanar target can be conducted with similar efficacy as with a planar target, depending on how precise a final relative pose is required. Unfortunately a nearly straight-on approach may not be guaranteed, so Scenario 4NP2 examines a highly skewed approach with respect to the same target.

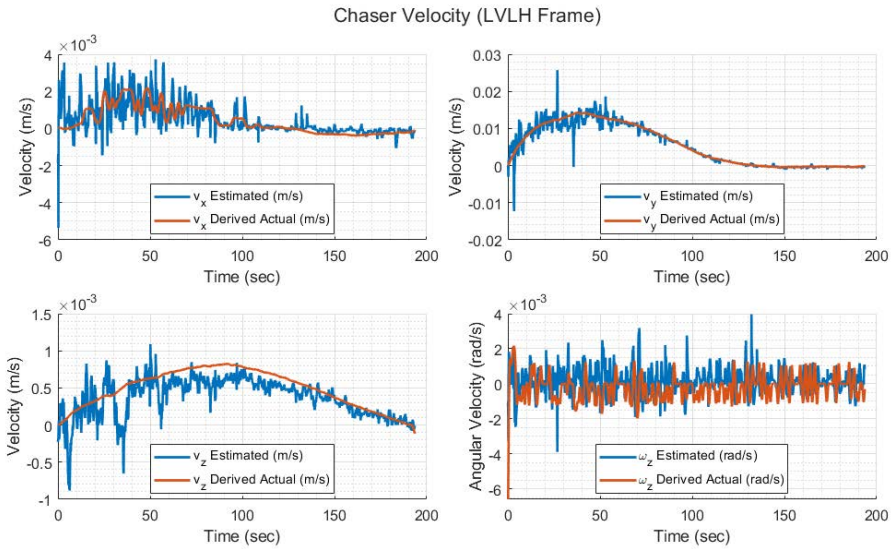

Figure 4.27: Scenario 4NP1: Chaser velocity with respect to LVLH frame (estimated, commanded, derived actual)

4.3.2.2 Scenario 4NP2: Complex Maneuver with Nonplanar Target.

Scenario 4NP2 resembles Scenario 4P2, with the exception that the target ROI is now nonplanar. The chaser is initially positioned at $[-1.20, 1.20, -0.10]^T$ meters with respect to the target/LVLH frame, pointing at the target ROI centroid (about -47° rotation about the LVLH Z-axis). Section 3.4.3.3 showed that conducting an approach maneuver with a nonplanar ROI can be made more effective by splitting the approach into a series of smaller incremental maneuvers. Accordingly, two trajectories are shown for this scenario—one where the approach is conducted as a single maneuver, and one where the approach is conducted as two incremental maneuvers. Unfortunately, the small size of CASpR is not conducive to examining incremental maneuvering with more than two maneuvers. However, the results of this section demonstrate that the positive effect of incremental maneuver is apparent with as little as two maneuvers.

Figure 4.28 shows the initial and final camera images from this scenario, including final images for both the single maneuver run and for the two incremental maneuver

run. The chaser clearly does not achieve the desired goal pose of being orthogonal to the predominant surface of the target ROI upon termination of a single maneuver (bottom left of Figure 4.28). Two incremental maneuvers yield a relative pose closer to the desired pose (final image shown in bottom right of Figure 4.28), but still visibly not orthogonal to the predominant surface of the target ROI.

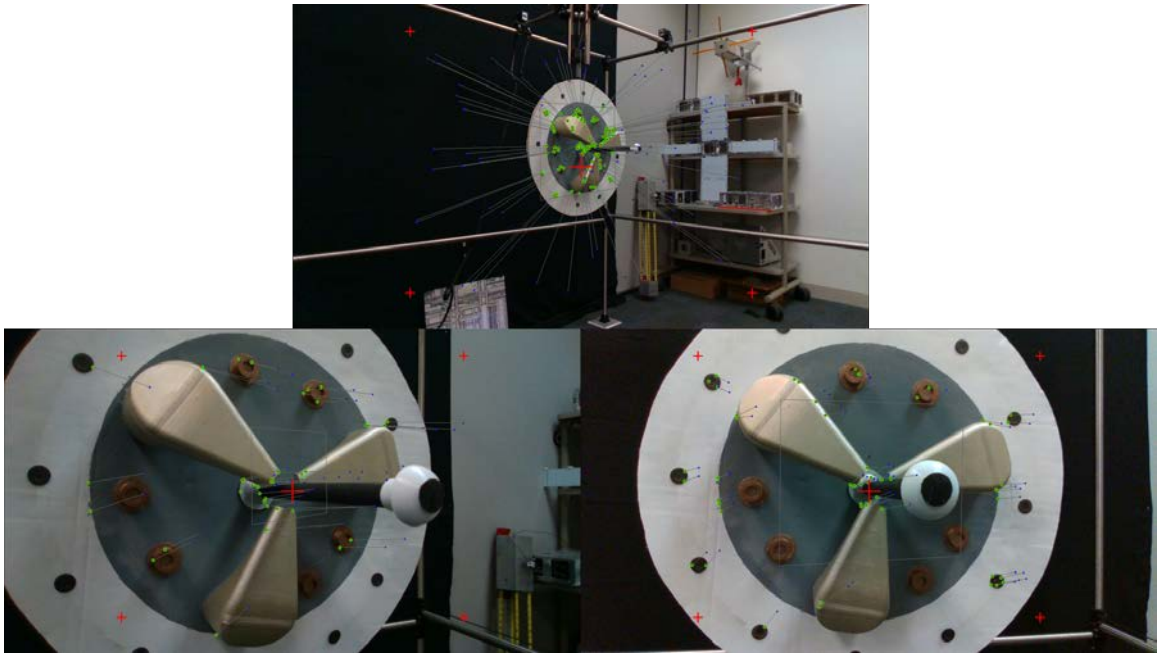

Figure 4.28: Scenario 4NP2: Initial chaser camera view (top), final view for single maneuver (bottom left) and final view for two incremental maneuvers (bottom right)

The chaser's trajectories for the single maneuver and two incremental maneuvers are shown in Figure 4.29, along with a camera symbol indicating the approximate desired relative pose. The two incremental maneuvers yield a better final relative position: $[0.56, -0.07, 0.00]^T$ meters versus $[0.49, -0.20, 0.00]^T$ meters with respect to the target/LVLH frame, and a significantly better relative rotation: $-10.69°$ versus $-27.39°$ rotation about the LVLH Z-axis. However, even for the incremental maneuvers, there is

over 10° more relative rotation at the end of the maneuver compared to the analogous scenario with a planar target.

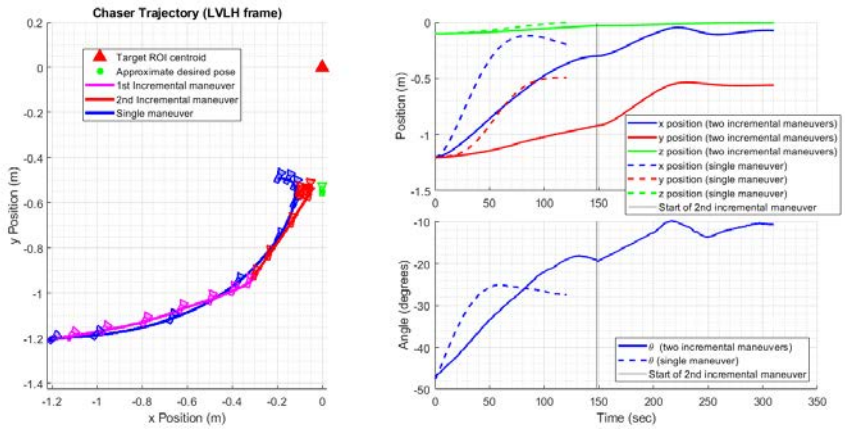

Figure 4.29: Scenario 4NP2: Chaser trajectory (LVLH frame)

The standard deviations of the servo error at the end of the maneuver are also significantly better for incremental maneuver as compared to a signal maneuver: 0.0144/0.0159 normalized pixels in the horizontal/vertical directions for two maneuvers versus 0.0486/0.0241 normalized pixels for a single maneuver. However, these deviations are still worse than that of all previous scenarios, indicating that the warp method performs more poorly for nonplanar targets viewed from an angle.

Chaser velocity plots for the single maneuver are shown in Figure 4.30 (plots for two incremental maneuvers are similar and are not shown for conciseness). Importantly, these plots show that the method still does well estimating relative velocity even when the target is nonplanar.

These results corroborate the simulation results presented in Section 3.4.3.3, namely that while conducting a single large maneuver with a nonplanar target may be unsatisfactory, conducting a series of small incremental maneuvers can improve performance (get the chaser closer to the desired relative pose).

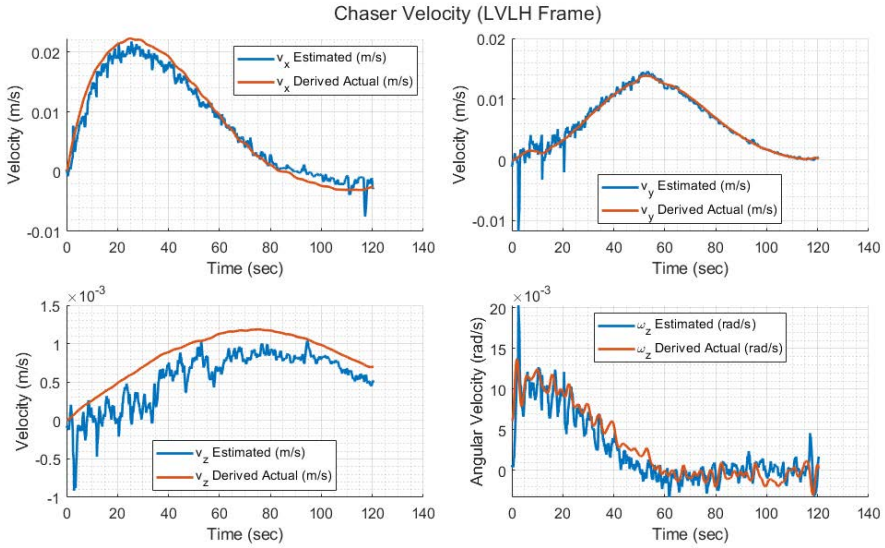

Figure 4.30: Scenario 4NP2: Chaser velocity with respect to LVLH frame (estimated, commanded, derived actual)

4.3.2.3 Scenario 4NP3: Pose-Hold with Moving Nonplanar Target.

The final scenario demonstrates pose-hold mode (introduced in Section 4.2.4) with a rotating nonplanar target. For this scenario, the chaser is initially positioned at $[0.60, 0.00, 0.00]^T$ meters with respect to the target body frame (which is initially aligned with the LVLH frame). At the start of the scenario, the target is given an angular rotational rate of $1.0°/sec$ about its body frame Z-axis, and maneuvering proceeds until the target has rotated 135°. As the chaser is in pose-hold mode, it attempts to maintain its initial relative pose with respect to the target for the duration of the maneuver.

Figure 4.31 shows the trajectory of the chaser and target. Chaser camera and target normal vector symbols with the same color correspond to the same time step. The final relative position of the chaser with respect to the target body frame is $[0.66, 0.02, 0.01]^T$ meters. The rotation of the chaser with respect to the target body frame Z-axis at the end of the maneuver is 0.95°.

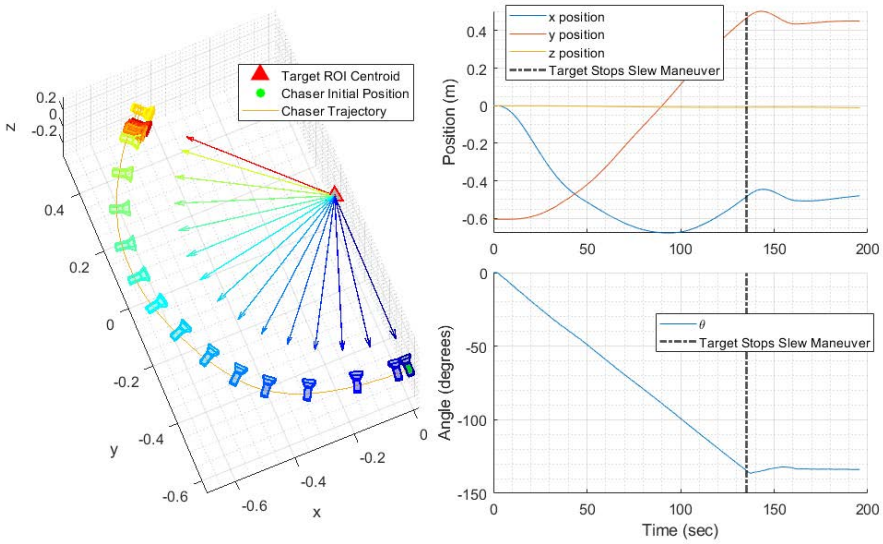

Figure 4.31: Scenario 4NP3: Chaser trajectory (LVLH frame)

The servo error for all tracked features is depicted in Figure 4.32. Since pose-hold mode is implemented, the servo error for all features starts at 0, as the chaser is assumed to be in its goal pose. There is initially an decrease in the horizontal (u) direction servo error, which is expected as the target initiates its horizontal rotation at the beginning of the maneuver and the chaser cannot instantaneously compensate for this.

The estimated and derived actual relative velocity of the chaser with respect to the target, resolved in the chaser/camera frame, is displayed in Figure 4.33. As with previous scenarios, the estimate tracks the derived actual velocity well enough to affect a successful maneuver.

This scenario demonstrates that pose-hold mode can be used to maintain relative pose, even with respect to maneuvering, nonplanar targets. This represents an important capability for close-in servicing, inspection, or formation flying RPO missions.

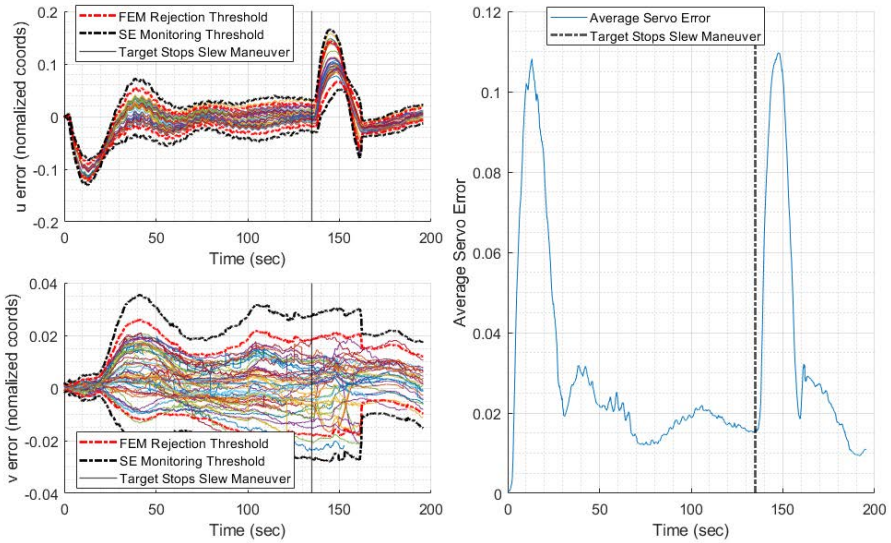

Figure 4.32: Scenario 4NP3: Servo error

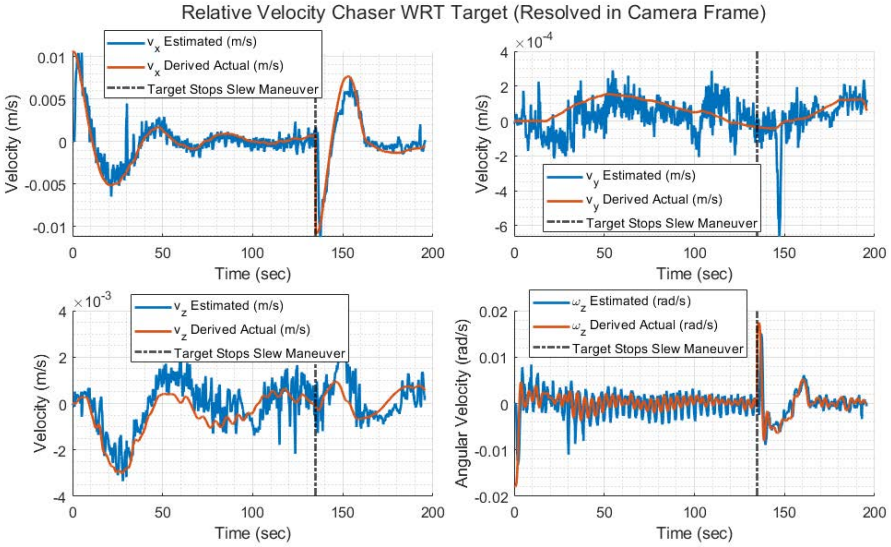

Figure 4.33: Scenario 4NP3: Chaser velocity with respect to LVLH frame (estimated, commanded, derived actual)

4.3.3 Results Summary.

The numerical results for all the scenarios presented in this chapter are included in Table 4.1 for ease of comparison. Color-coded cells provide a visual indication comparing the results of each scenario, with green tones indicating better performance and red tones indicating poor performance.

Table 4.1: Summary of Results for Chapter IV Scenarios

	Scenario						
	4P1	4P2	4P3	4NP1	4NP2 Single Maneuver	4NP2 Two Maneuvers	4NP3
Target type	Planar	Planar	Planar	Nonplanar	Nonplanar	Nonplanar	Nonplanar
Target angular velocity (deg/s)[1]	0.00	0.00	0.53	0.00	0.00	0.00	1.00
Initial relative position (m)[2]	[1.70, -0.13, -0.10]	[1.20, -1.23, -0.10]	[-0.57, -0.10, 1.00]	[1.70, -0.10, -0.10]	[1.20, -1.20, -0.10]	[1.20, -1.20, -0.10]	[0.60, 0.00, 0.00]
Final relative position (m)[2]	[0.49, 0.00, 0.01]	[0.49, 0.00, 0.01]	[0.53, 0.01, -0.01]	[0.66, -0.02, -0.01]	[0.49, -0.20, 0.00]	[0.56, -0.07, 0.00]	[0.66, 0.02, 0.01]
Intial relative rotation (deg)[3]	-6.46	-46.46	30.98	-4.95	-47.48	-46.97	0.00
Final relative rotation (deg)[3]	-0.21	-0.28	0.16	-2.48	-27.39	-10.69	0.95
Number of features detected[4]	40	20	20	20	90	90	88
σ of hort. servo error (normalized pixels)[5]	0.0054	0.0067	0.0066	0.0095	0.0486	0.0144	0.0059
σ of vertical servo error (normalized pixels)[5]	0.0061	0.0062	0.0064	0.0146	0.0241	0.0159	0.0055

Notes:
1 - Angular velocity about LVLH/target body frame Z-axis.
2 - Position of chaser with respect to target body frame, resolved in the target frame.
3 - Relative rotation refers to relative anglular position about the LVLH/target body frame Z-axis.
4 - Number of features detected during maneuver initialization procedure.
5 - Standard deviation of servo error at termination of maneuvering.

4.4 Recommendations for Future Work

This section provides recommendations for future work that could improve the navigation and control method proposed in Chapters III and IV. Recommendations are split into three categories: improvements regarding the feature tracking pipeline, improvements specific to nonplanar targets, and improvements for practical implementation during on-orbit RPO.

4.4.1 Improving the Feature Tracking Pipeline.

Since features identified on the target via computer vision methods represent the sole measurement available for relative navigation, correctly tracking target features is paramount to success this of this architecture. In both simulation and experimentation, tracking more features was observed to improve performance and reliability. Additionally, the accuracy of the feature state estimates is important, especially when only a few features are tracked during the maneuver. Thus, continuing research to improve the feature tracking pipeline is worthwhile. To this end, two possible improvements are proposed in this section: a method to improve FEM by updating feature descriptors, and a method to add new features during maneuvering.

In the implementation of the feature tracking pipeline introduced in this chapter, a descriptor vector for each target feature is computed during the initialization subroutine. During each subsequent instance of FEM, descriptor vectors for features identified in the current image frame are compared against the initial set of target feature descriptors in order to determine feature matches. As the maneuver progresses and the current image scene becomes more and more different than the initial image scene, the likelihood of matching descriptors from the current image to the initial image becomes more unlikely (this can be seen in Figures 4.12 and 4.22). Therefore, it is proposed that each time a target feature is matched in the current image scene, the descriptor vector for that feature is updated to the vector computed for the current scene. Doing so should allow target features to be matched further into the maneuver, however it is unknown how this could affect the ability of the chaser to achieve the desired relative pose.

As introduced in this chapter, there is no mechanism in the feature tracking pipeline to add new target features during the maneuver. Features are only identified during the initialization as the homography used to generate the goal feature locations is only valid for the initial relative pose. It should be possible estimate a homography from the current

pose to the goal pose during maneuvering (see Section 2.4.2.2), which could then be used to generate goal positions for new target features identified in the current image scene using the warp method described in Section 3.3.2. However, homography estimates computed online can be susceptible to large variation, so experimentation would be required to ascertain the effectiveness of this augmentation.

4.4.2 Improving Utility with Nonplanar Targets.

The results of Section 3.4.3.2 and 4.3.2.2 indicate that the chaser can fail to fully achieve the desired relative pose during for angled approaches with highly nonplanar targets, therefore future work should consider methods to improve performance for these types of maneuvers. Two possible improvements include: exclusively utilizing features that approximately lie on a plane associated with the target ROI, and developing a better method to generate goal locations of feature points.

The failure to reach the desired goal pose for nonplanar targets is a result of the warp method's inability to correctly predict the goal image locations for nonplanar features on the target surface. It is possible that this degeneracy could be circumvented by only utilizing target features which lie on a plane associated with the selected ROI on the target. To implement this however, a predominant surface plane would have to exist in the target ROI, and a method would need to be developed to ignore detected features that are not on that surface plane.

Alternatively, future work should explore developing a better method to generate goal locations for target features on nonplanar targets. Here depth information for each feature point would surely be useful to transform the initial understanding of target feature points to desired locations.

4.4.3 Improvements for Practical Implementation.

Future work should also focus on improvements that will increase utility during actual spacecraft RPO. Two possible focus areas for this work include: augmenting the method

and/or sensor hardware to enable its use over a wider relative range, and examining how the method can be optimized to minimize control effort during maneuvering.

The fixed-focal length cameras assumed in this work limit the ability of the method to control maneuvers requiring a large change in relative range between the chaser and the target. This is primarily due to the difficultly associated with tracking feature points on the target as the relative size of the target in the image plane changes significantly over the course of a maneuver. Several research efforts have examined using zooming cameras with IBVS in order to increase usable range, including [127, 129, 143]. Alternatively, the method could be implemented using multiple sets of stereo cameras, each with a different focal length intended to be used at different ranges as the chaser maneuvers over a large distance relative to the target.

Minimizing fuel use is paramount during RPO—additional research should examine modifying the method in order to optimize the chaser's relative trajectory for fuel use. Research on this topic for traditional visual servoing methods exists, see for example [144, 145].

4.5 System Parameters

This section provides a summary of all the tunable parameters associated with this method, including the values used for the results presented herein, and relevant notes for each parameter. The intent of this section is to provide designers a starting point for future implementations of this method. Tables 4.2, 4.3, and 4.4 detail the tunable parameters associated with the sensing subsystem, navigation subsystem and control subsystem, respectively.

Table 4.2: Sensing Subsystem Tunable Parameters

Tuning Parameter	Value/Setting for this Work	Description/Notes
Stereo Camera Settings		
RGB camera resolution	1280x720 pixels	Higher resolution increases computational load but improves feature tracking
Depth map resolution	1280x720 pixels	Higher resolution increases computational load but may improve depth estimation for small nonplanar target components
Other settings (focal length, shutter speed, ISO, etc.)	Intel RealSense D345 defaults, autoexposure off for RGB and depth	Unique to equipment/application, not addressed in this work
Initialization Procedure		
Number of frames to process	10	Number of sequential frames to taken from initial pose to use for feature detection
Feature mask radius	10 pixels	Radius around dtected features to mask off in subsequent detection images (pixels)
Max features to ID each frame	Variable depending on scenario	Used to limit the number of features identified in each detection image
KLT Feature Tracking		
Window size	10x10 pixels	Area around feature to search for new location of feature (pixels)
Max level	2	Number of pyramidal levels to use
Termination criteria	10 iterations, epsilon 0.3	Number of iterations and threshold to complete search for matches. Tradeoff speed and accuracy
Forward/backward error	1 pixel	Threshold to dertimine if LKT match is valid
Feature Extraction		
Feature detector	BRISK	Method to detect features in current image (e.g. SIFT, SURF, ORB, etc.)
Feature descriptor	BRISK	Method to generate descriptor vectors for detected features in current image (e.g. SIFT, SURF, ORB, etc.)
Feature Matching		
Matching algorithm	FLANN	Method to ID matches between current image and initial image (e.g. FLANN)
Outlier rejection method	RANSAC	Model-based method used to detect matching errors (e.g. RANSAC, LMEDS)
Model	Homography	Model used by RANSAC to determine which matches are outliers (e.g. homography or essential matrix)
Max number of iterations	3000	Maximum number of iterations to run RANSAC
Max reprojection threshold	3.0	Maximum reprojection error beyond which a point is considered an outlier
Confidence	0.99999	Required confidence of matches (between 0-1)
Servo error rejection threshold	2σ +/- mean servo error	Threshold of servo error variance beyond which a match is considered an outlier
Depth Filtering		
Decimation filter magnitude	2x	More decimation reduces scene complexity and improves computational efficiency
Spatial filter magnitude	2	Number of spatial filter iterations
Spatial filter smoothing factor	0.25	Alpha factor for exponential moving average filter
Spatial filter edge size threshold	20	Threshold used to determine what is an edge
Temporal filter smoothing factor	0.4	Alpha factor for exponential moving average filter
Temporal filter edge size threshold	20	Threshold used to determine what is an edge

Table 4.3: Navigation Subsystem Tunable Parameters

Tuning Parameter		Description/Notes
Kalman Filter Bank		
Process covariance (pixel dynamics)	Position states: 0.001 Velocity states: 1	Set high to prioritze feature location measurements
Process covariance (depth dynamics)	Position state: 0.001 Velocity state: 1	Set high to prioritze depth measurements
Measurement covariance for KLT Tracker	0.2	High values smooth feature trajectory, but introduce lag
Measurement covariance for depth sensor	0.5	High values smooth depth estimate, but introduce lag
Initial covariance	Position states: 1 Velocity states: 0.1	Affects rate of convergence of filter, not observed to have a major effect on performance
Mahalonobis distance threshold	1E6	Threshold for outlier detection
Relative Velocity Estimate		
LPF to smooth estimate	Not utilized for this work	Could be implemented to smooth relative velocity estimate, at cost of delayed estimate
Velocity estimate pseudo-inverse sigular value tolerance	None	Could potentially be used to improve velocity estimate, not observed to be a factor for this work

Table 4.4: Control Subsystem Tunable Parameters

Tuning Parameter		Description/Notes
Control Mode		
Minimum number of healthy features	8	Number of healthy features below which pose-hold failsafe mode is initiated.
IBVS Controller		
Interaction matrix estimation	See Eqn. (2.50)	See Section 2.8
IBVS velocity command gain	50*diag([0.15,0.25,0.3,.15])	nDOF by nDOF diagonal matrix, each diagonal element corresponds to DOF
Estimated depth in goal pose	0.6 m	Coarse estimate OK, only affects rate of convergence
Velocity command pseudo-inverse sigular value tolerance	None	Could potentially be used to improve velocity command, not observed to be a factor for this work
Servo error monitoring threshold	3σ +/- mean servo error	Threshold of servo error variance beyond which a feature is considered an outlier
PID Controller		
Proportional gain	1000*eye(4)	PID control gains heuristically tuned to provide suitable decay of commanded - estimated relative velocity error. Derivative filter is a single-pole inifinite impulse response low-pass filter.
Iterative gain	0.01*eye(4)	
Derivative gain	50*eye(4)	
Derivative filter parameter	0.5	

4.6 Conclusion

This chapter examined the architecture proposed in Chapter III using HITL experimentation. AFIT's CASpR was used to simulate relative motion of chaser and target bodies with respect to the LVLH frame. A stereo camera sensor attached to the chaser was used exclusively for relative navigation and control. Several extensions to the method were proposed and discussed to enable hardware implementation and improve performance.

The efficacy of the method was demonstrated using actual camera feedback and computer vision algorithms to track target features. It was shown that performance of the method with respect to planar target ROIs is similar to that seen in simulation. Notably, pose-hold mode, introduced in Section 4.2.4, demonstrated the method's ability to maintain a relative pose, even with respect to maneuvering nonplanar targets.

Finally, Section 4.4 discussed several topics pertaining to the method, with the intent to facilitate future implementation and make recommendations for improvement. Future work should focus on improving the method's effectiveness with highly nonplanar target ROIs and continuing to improve the feature tracking pipeline. The results of this chapter and the previous chapter motivate the next section of this work: development of a novel visual servoing method known as Surface Normal Visual Servoing (SNVS).

V. GNC Method 2: Surface Normal Visual Servoing (SNVS) for Final Approach Maneuvers with Unknown Targets

5.1 Overview

This chapter proposes a novel method of visual servoing, hereafter referenced as Surface Normal Visual Servoing (SNVS). Two variations of the method are developed and analyzed. These variations, termed SNVS Variant 1 and SNVS Variant 2, share many guidance, navigation, and control (GNC) functions, with the exception of how translational and attitude control commands are generated. Both variations are presented as each has been found to have utility for different use cases.

Like the augmented Image-Based Visual Servoing (IBVS) method presented in Chapters III and IV of this work, SNVS is intended to enable rapid initialization of final approach maneuvers with unknown, noncooperative targets. No information about the target is required prior to maneuvering other than a single image of the target captured from the chaser's initial pose, which is used by an operator to define the intent of the maneuver. As with the previous IBVS method, the only relative navigation sensor used during maneuvering is a stereo camera sensor.

SNVS was developed to overcome the noted drawbacks of the IBVS method previously presented. Compared to that method, SNVS:

1. Is able to precisely capture a desired relative pose when maneuvering with respect to targets containing significant nonplanar features.
2. Does not require natural feature point tracking.
3. Allows the desired relative pose to be explicitly defined in a straight-forward manner.
4. May be implemented with depth information only, depending on the nature of the target and intent of a maneuver. Doing this permits use of alternative depth sensors

such as flash LIDAR, which would enable utilization of the method under a wider range of environmental conditions (such as low light conditions). See Section 5.6.3 for a discussion of how SNVS could be implemented without a camera sensor.

In addition to these advantages, SNVS has several notable benefits over other potential methods for maneuvering with respect to unknown targets:

1. Does not require a geometric model of the target to be developed *a priori* or online, nor are other properties of the target (such as mass or moments of inertia) required to be estimated.

2. Only requires a single relative navigation sensor, a stereo camera.

3. Can be easily implemented in real-time using low-cost hardware.

The defining characteristic of SNVS is its use of a unique navigational vector associated with a region of interest (ROI) on the target, termed the Predominant Surface Normal (PSN) vector. The PSN vector for the ROI is computed using a novel algorithm capable of being implemented in real time. This vector, along with a tracked Point of Interest (POI) on the target, are used in a feedback loop to generate control commands for the chaser. Note that the SNVS methods presented here do not command relative rotation about the chaser camera's optical axis (boresight). See Section 5.2.5 for a discussion regarding how this degree of freedom can be controlled using a separate controller.

Implicit in the use of the PSN vector is the assumption that the target ROI contains a predominant surface plane. That is, the ROI cannot be continuously curving. This caveat does not significantly limit utility of the method for spacecraft rendezvous and proximity operations (RPO) however, as many spacecraft surfaces are either predominantly planar, or locally planar ROIs can be selected for maneuvering (see Section 5.6.2 for a discussion of the method's effectiveness with curved targets).

The rest of this section provides a brief description of each SNVS variant. Section 5.2 presents GNC functions common to both variants. Sections 5.3 and 5.4 discuss elements unique to each variant. Section 5.5 presents simulation results for both variants. Finally, Section 5.6 discusses several topics relevant to this method, and Section 5.7 provides concluding remarks.

5.1.1 Description of SNVS Variant 1.

Figure 5.1 shows a block diagram overview of SNVS Variant 1. The sensing and image processing functions for this method are composed of stereo image processing, target POI tracking, tracking of the predominant surface plane associated with the target ROI, and estimating the PSN vector of the target ROI. Kalman filters are used to refine the state estimates of the PSN vector and the POI. As measurements for these filters come from a camera sensor, *state estimates are resolved in the camera frame*. Separate controllers are implemented to regulate relative translational and relative rotational motion.

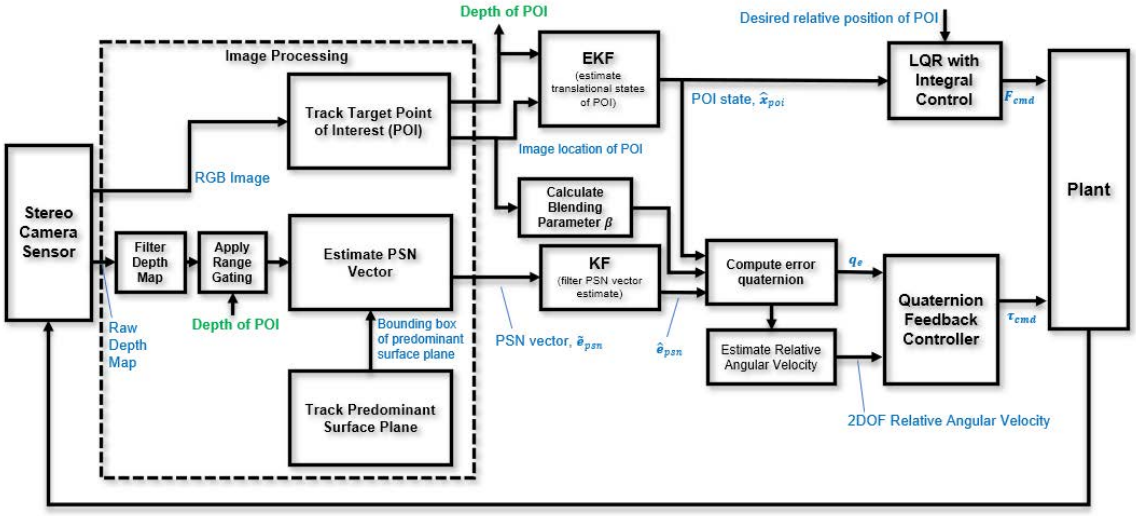

Figure 5.1: SNVS Variant 1 block diagram

For Variant 1, the translational controller utilizes full-state feedback to drive the POI to a desired state. The attitude controller drives the camera boresight vector to be parallel with the PSN vector, while simultaneously ensuring that the POI stays in the camera's field of view. Notably, because the POI state is resolved in the camera frame, the desired relative attitude must first be achieved in order to reach a desired relative position with respect to the target frame of reference.

5.1.2 Description of SNVS Variant 2.

Figure 5.2 shows a block diagram overview of SNVS Variant 2. The sensing, image processing, and navigation filter functions of this variant are identical to those of Variant 1. This variant also implements separate translational and attitude controllers.

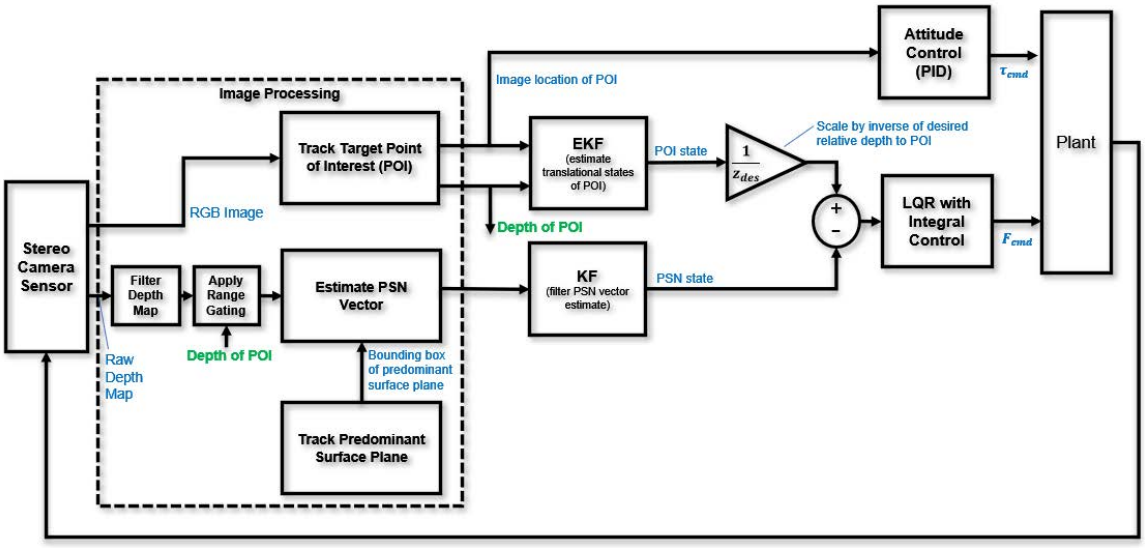

Figure 5.2: SNVS Variant 2 block diagram

For Variant 2, attitude control regulates the POI to a desired position in the camera image. Translational control regulates the POI state (resolved in the camera frame) scaled by the inverse of the desired depth to the target, to the PSN vector state estimate. For this

variant, the desired attitude need not be achieved prior to the chaser achieving a desired position with respect to target frame of reference.

5.2 Common Elements of Both SNVS Methods

5.2.1 Prior to Maneuvering.

As with the IBVS method presented previously, prior to employing this method for RPO with unknown targets, it is first necessary for an operator to define the desired end state (i.e. the desired relative pose) of the final approach maneuver. To do this, it is assumed that the operator has access to resolved imagery of the target from the chaser's initial pose, such that a portion of the target which will be the subject of the approach maneuver can be manually selected. For the control method as presented, it is assumed that the desired end state of the approach maneuver places the chaser at a desired relative distance from a POI on the target, with that POI centered in the chaser camera's Field of View (FoV), and the camera's optical axis orthogonal to the predominant surface plane associated with an ROI on the target.

In order to initiate a maneuver, the operator must:

1. Select a POI on the target.
2. Define the desired relative distance to the target POI.
3. Select an area on the target containing a predominant surface plane, such that the final relative attitude of the chaser with respect to the target will place the camera optical axis orthogonal to this plane.

5.2.2 Estimating the Predominant Surface Normal (PSN) Vector of an ROI.

SNVS relies on the ability to track a navigational vector derived from information available about the target ROI. This navigational vector is the PSN vector—a normal vector orthogonal to the predominant surface plane associated with the selected ROI on the target.

The PSN vector must be estimated in real-time over the course of the maneuver in order to generate position and attitude control commands.

The use of surface normal vectors is common in the field of computer vision and computer graphics. Numerous research efforts have examined methods to compute normal vectors associated with objects in a visual scene using visual and/or depth information about the scene (see e.g. [146, 147]). While there are a few basic similarities to the methods presented herein, these efforts generally pertain to the use of normal vectors for image segmentation and object identification, not navigation and control.

This section presents two procedures to estimate the PSN vector of an ROI. The first ('naïve procedure'), introduces how normal vectors can be computed using depth information, and the second ('improved procedure'), modifies the basic procedure to increase accuracy and computational efficiency.

5.2.2.1 Naïve Procedure.

A basic method to estimate the PSN vector associated with the target ROI involves first generating a map of surface normal vectors for every pixel within the ROI, in a similar manner to [146]. Then, histograms for each vector component are generated using every pixel's computed surface normal vector in the ROI. These histograms are used to find the most commonly occurring surface normal vector in the ROI—the PSN vector.

Recall the pinhole camera model, which creates a perspective projection of a 3D point $\boldsymbol{P} = [X, Y, Z]^T$ on a 2D image plane,

$$\boldsymbol{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & \gamma & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{5.1}$$

where $[u, v]^T$ are the pixel coordinates associated with $\boldsymbol{P}$, f_u, f_v are the horizontal and vertical focal length (expressed in units of pixels), (c_u, c_v) is the principal point of the image

frame, γ is the skew coefficient representing the amount of shear distortion for each pixel of the sensor (typically 0 or nearly 0), and $\boldsymbol{K}$ is the camera calibration/intrinsic matrix. The vector $[x, y, 1]^T$ represents the point expressed as a homogeneous vector using normalized image plane coordinates.

For this procedure, to compute the surface normal vector corresponding to a pixel $\boldsymbol{p}$ in an image, the surface gradient vectors associated with this pixel in the vertical and horizontal directions are computed, then the cross product of these two vectors is calculated to determine the surface normal vector associated with $\boldsymbol{p}$. To do this, the 3D points associated with $\boldsymbol{p}$ and its neighbors are first calculated,

$$\boldsymbol{P} = Z(u, v)\boldsymbol{K}^{-1}\boldsymbol{p}, \tag{5.2}$$

$$\boldsymbol{P}_{\Delta u} = Z(u - \Delta u, v)\boldsymbol{K}^{-1}\left(\boldsymbol{p} - \begin{bmatrix} \Delta u \\ 0 \\ 0 \end{bmatrix}\right), \tag{5.3}$$

$$\boldsymbol{P}_{\Delta v} = Z(u, v - \Delta v)\boldsymbol{K}^{-1}\left(\boldsymbol{p} - \begin{bmatrix} 0 \\ \Delta v \\ 0 \end{bmatrix}\right), \tag{5.4}$$

where $Z(u, v)$ is the depth associated with pixel $[u, v]^T$, and Δu and Δv are horizontal and vertical pixel variations, respectively. For a noiseless depth map, Δu and Δv can be set to 1 pixel, though when using real (noisy) depth maps, setting these values to 10-20 pixels yields better results. The gradient vector in each direction is then:

$$\Delta \boldsymbol{U} = \boldsymbol{P}_{\Delta u} - \boldsymbol{P}, \tag{5.5}$$

$$\Delta \boldsymbol{V} = \boldsymbol{P}_{\Delta v} - \boldsymbol{P}. \tag{5.6}$$

The surface normal vector associated with $\boldsymbol{p}$ is

$$\boldsymbol{n_p} = \frac{\Delta \boldsymbol{U} \times \Delta \boldsymbol{V}}{||\Delta \boldsymbol{U} \times \Delta \boldsymbol{V}||}. \tag{5.7}$$

Note that the sign of the Z component of $\boldsymbol{n_p}$ indicates whether the normal vector is pointing into the surface of the ROI or out of it. For the control law as formulated in this work, positive values for this Z component are required, thus $\boldsymbol{n_p}$ is negated if its Z component is initially calculated as negative.

In practice, occluded portions and gradients around sharp edges can be ignored if the difference in depth between $\boldsymbol{p}$ and its neighbors is large [146]. Depth maps generated from stereo are typically noisy, so smoothing the depth map is useful, and can be implemented as described in Section 4.2.3.2 (see also [148, 149]). Figure 5.3 shows an example depicting surface normal calculation for an entire scene generated using a simulated image and noiseless depth map. Figure 5.3(c) encodes the surface normal vectors associated with each pixel on the target spacecraft as a specific RGB value.

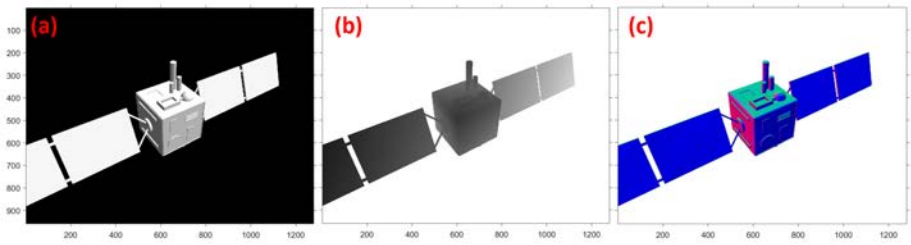

Figure 5.3: (a) Camera sensor gray-scale image (b) depth map (c) surface normal map

Once a surface normal map for the target ROI is generated, the PSN vector of this ROI is taken as the most common normal vector occurring in this set. Histograms are used to compute the most likely vector; a histogram is generated for each vector component using the set of surface normal vectors within the ROI. The peak of each histogram is

then taken as the corresponding component of the PSN vector. Finally, the PSN vector is re-normalized to ensure unit magnitude.

Figure 5.4 displays an example of this procedure generated using actual stereo imagery. An ROI containing nonplanar features is selected as shown in Figure 5.4 (a). The depth map aligned with this scene is shown in Figure 5.4 (b), and Figure 5.4 (c) shows the surface normal map associated with the ROI. Figure 5.4 (d) displays the PSN vector component histograms generated using the ROI surface normal map. The bin values of the peaks of each histogram are taken as the corresponding vector components of the PSN vector.

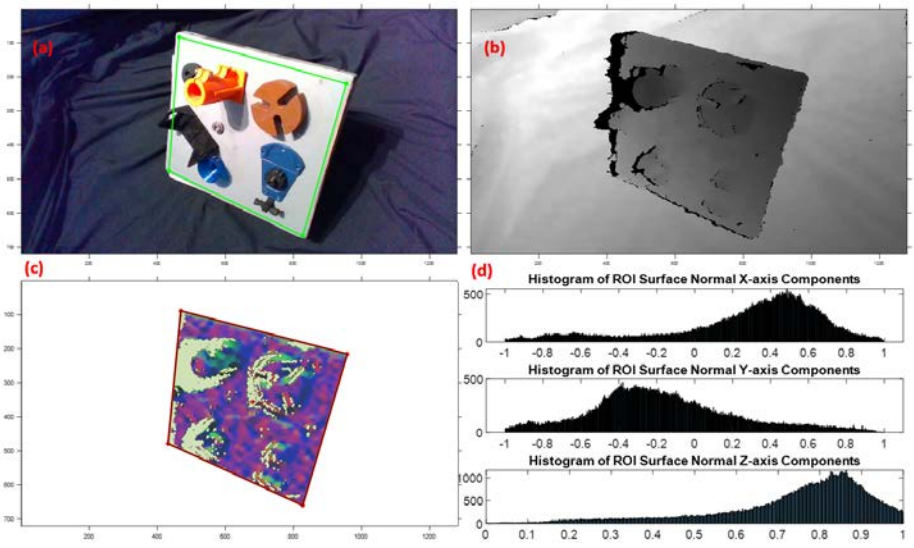

Figure 5.4: (a) Color image of target ROI (depicted by green quadrangle), (b) depth map computed by stereo camera sensor, (c) surface normal map (d) histograms of ROI normal vector components

In order to visualize the calculated PSN vector and assess its quality, a 3D point cloud can be generated from the depth map using Equation (5.2). The point cloud for this example with the estimated PSN vector overlaid are shown Figure 5.5. As can be seen, the method correctly estimates the PSN vector associated with the ROI.

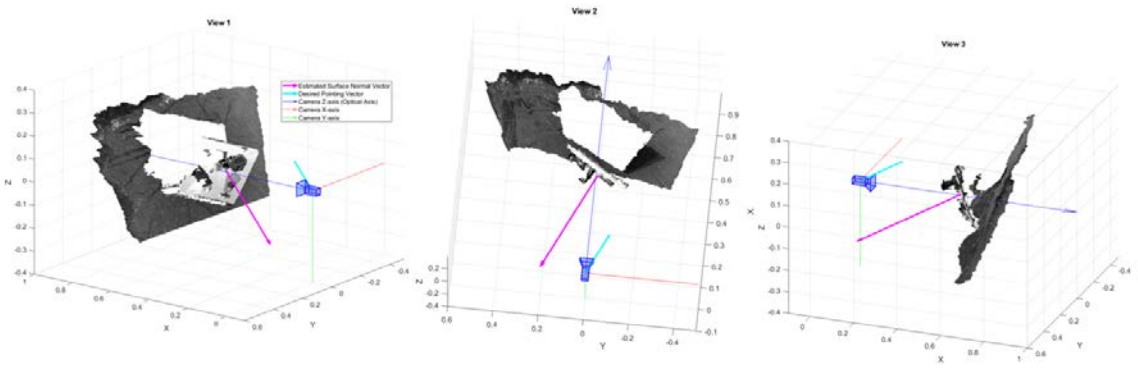

Figure 5.5: Three views depicting the estimated predominant surface normal vector of ROI and desired pointing direction of camera, overlaid on stereo generated point cloud of scene

5.2.2.2 Improved Procedure.

A major limitation of the procedure described in the previous section is that its computational complexity depends on the size of the target ROI in the image plane. That is, because the procedure calculates surface normal vectors for every pixel in the ROI, the size of ROI determines how many computations must occur to calculate the PSN vector. Furthermore, because the procedure only uses depth data from nearby pixels to determine the surface gradients, calculated gradient vectors are highly influenced by noise in the depth map.

Rather than cycling over every pixel in the ROI, the quality and speed of PSN vector calculation can be improved by calculating the surface normal vector for a fixed number of randomly sampled points within the ROI. Additionally, instead using local neighbors in the horizontal and vertical directions to calculate surface gradient vectors as with the naïve procedure, the improved procedure randomly samples two additional points, which when combined with the center point form two vectors whose cross product represents the normal vector for the center point.

To implement the improved procedure, a fixed number of sets of three points are uniformly sampled from within the ROI. To illustrate this, Figure 5.7 depicts 10 sets of

sampled points and their corresponding vector pairs for the same example presented in the previous section. 3D positions of the points of each set ($\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3$) are calculated using Equation (5.2), and gradient vectors are calculated,

$$\Delta \boldsymbol{v}_A = \boldsymbol{P}_2 - \boldsymbol{P}_1, \tag{5.8}$$

$$\Delta \boldsymbol{v}_B = \boldsymbol{P}_3 - \boldsymbol{P}_1. \tag{5.9}$$

The surface normal vector associated with $\boldsymbol{P}_1$ is then calculated as before (negating normal vectors whose Z component is negative),

$$\boldsymbol{n}_{\boldsymbol{p}_1} = \frac{\Delta \boldsymbol{v}_A \times \Delta \boldsymbol{v}_B}{||\Delta \boldsymbol{v}_A \times \Delta \boldsymbol{v}_B||}. \tag{5.10}$$

There is a chance that sampled points within a set may equal another point in the set, causing the computed normal vector to be undefined. This is easily handled by ignoring normal vectors with undefined values.

The PSN vector is calculated using histograms as described previously. Figure 5.7 shows the histograms generated using the improved procedure for the same example ROI presented in the previous section. As can be seen, these histograms have much more well-defined peaks compared to those generated with the naïve procedure. Sharper peaks like these facilitate more consistent re-identification of the correct PSN vector over multiple image frames and are indicative of this procedure's superior ability to identify accurate PSN vectors.

For the same example using the naïve procedure, normal vectors for all 197,930 pixels in the ROI were calculated. Using the improved procedure, only 10,000 sets of points were sampled, meaning for this example the improved procedure required almost twenty times

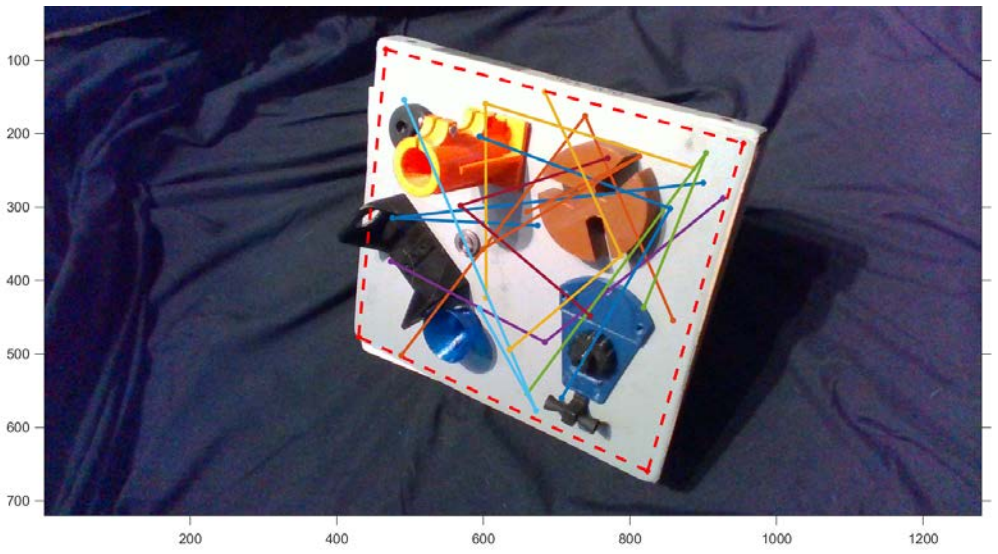

Figure 5.6: Example showing ten sets of randomly sampled points and their corresponding vector pairs used to calculate normal vectors for points with the ROI

less computation. If the code for this procedure is vectorized, it can easily be implemented in real time, as demonstrated in Chapter VI of this work.

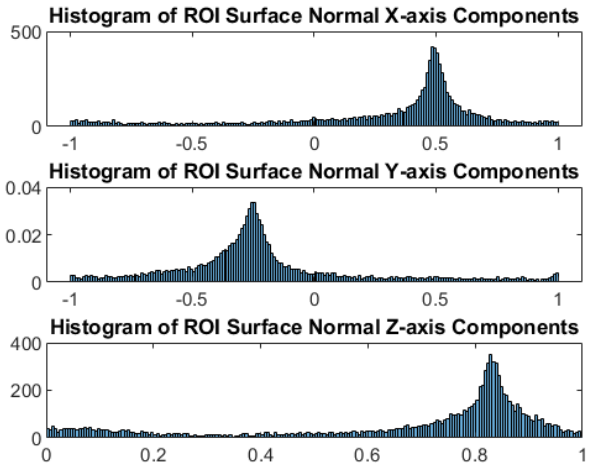

Figure 5.7: Histograms of the ROI normal vector components produced using the improved procedure

5.2.3 Target Tracking with Computer Vision.

The computer vision functions necessary to facilitate implementation of SNVS include tracking the target POI, and tracking the ROI associated with the predominant surface plane. As discussed in Section 2.4.3.2, there are many options for tracking feature points and ROIs in an image. This section presents tracking methods considered for this work, including a novel method for tracking the predominant surface plane associated with an ROI.

5.2.3.1 Tracking the Target POI.

Prior to maneuvering, a POI is selected on the target surface that will be used to regulate the relative trajectory of the chaser. This point must be tracked for the duration of the maneuver. One option here is to use Kanade-Lucas-Tomasi (KLT) with the manually selected feature point, or some augmented/similar method such as was presented in Section 4.2.3.

Alternatively, a single object tracker algorithm, such as those discussed in Section 2.4.3.2 can be used if a small region around a distinct POI on the target is selected. Using this method, the POI is assumed to be the located at the geometric centroid of the bounding box generated by the single object tracker. An experimental test campaign examining point tracking for several representative target objects showed that a better alternative to KLT is to track the target POI using a such an algorithm.

5.2.3.2 Tracking the Target ROI's Predominant Surface Plane.

In addition to tracking a POI on the target, the predominant surface plane associated with the target ROI must be tracked in order to enable estimation of the PSN vector of the ROI. As with the POI, one option here is to use a separate instance of a dedicated ROI tracking algorithm, with the operator judiciously selecting the ROI such that it contains a predominant plane that will satisfactorily define the desired relative pose.

Alternatively, this section proposes a novel method of tracking the predominant surface plane that is faster and potentially more robust than using a dedicated tracking algorithm. The basic concept of this method is to determine which sampled points lie on the predominant surface plane, then generate a bounding box that encompasses all of these points. This bounding box is used during the next timestep as the portion of the image from which to sample new points for PSN vector computation.

Sampled points are deemed to lie on the predominant surface plane if all of their vector components are within some threshold (a tunable parameter) of the corresponding PSN vector component. Figure 5.8 illustrates this concept.

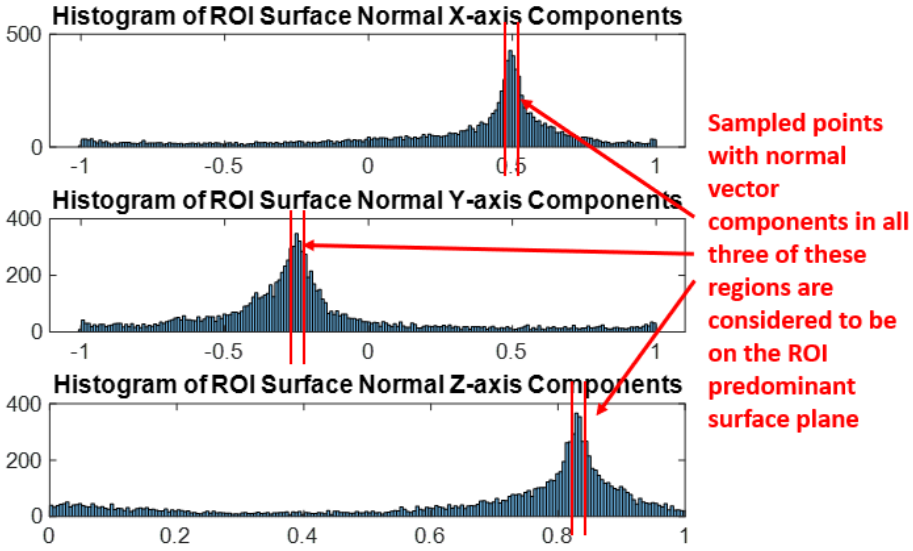

Figure 5.8: Example histograms of sampled points surface normal vectors, depicting how predominant surface plane points are determined

To create the bounding box surrounding the predominant surface plane points, the maximum and minimum pixel positions of all plane points in the horizontal and vertical directions are determined, and a buffer of several pixels is applied to these values (added to the max positions and subtracted from the minimum positions). This buffer is necessary to keep the bounding box from shrinking each iteration. The buffer also allows the

bounding box to grow from the initial region selected by the operator—the box will grow to encompass the entire predominant surface plane, even if only part of the plane was initially selected. The value of the buffer is a tunable parameter, with larger values causing the box to grow more quickly. Figure 5.9 depicts the predominant surface plane bounding box (magenta dashed rectangle) generated for the example previously discussed. Note that the bounding box sides are oriented horizontally and vertically—this is in general not a problem as long as the desired surface plane remains predominant within the bounding box.

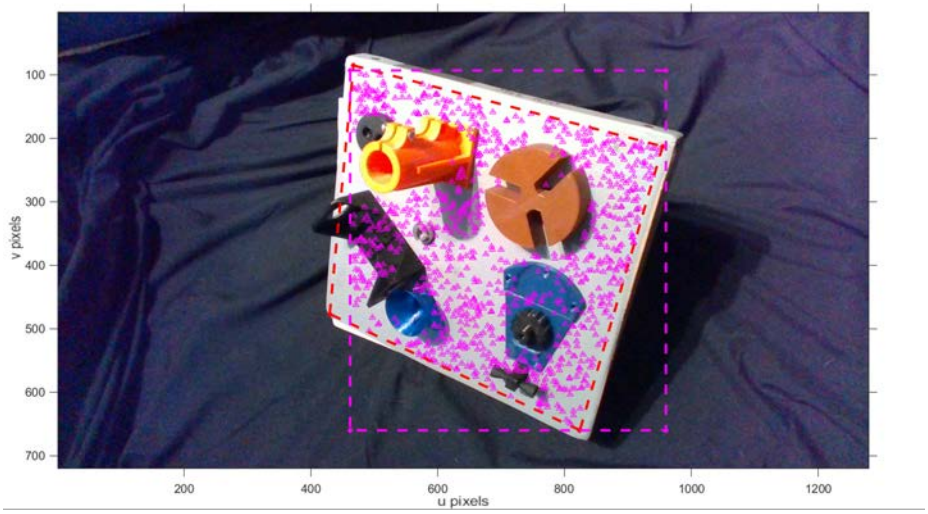

Figure 5.9: Example predominant surface plane points and bounding box (shown in magenta)

In order to prevent the predominant plane bounding box from growing to encompass parallel planes not associated with the target surface (i.e. other objects in the environment behind or in front of the target), range gating is applied to depth image prior to sampling points for the PSN vector computation. This is accomplished by zeroing out depth pixels with depths greater than or less than the depth of the target POI plus or minus a predefined gate range. The gate range is a tunable parameter, and should be set large enough for the

predominant surface plane to remain visible in the depth image, but small enough to reject other objects in front of and behind the target.

5.2.4 Navigation Functions of SNVS.

Both SNVS variants rely on estimates of the target POI state and PSN vector state. In this section, an Extended Kalman Filter (EKF) is presented to filter the POI state, and a Linear Kalman Filter (LKF) is presented to filter the PSN vector state. Note that state estimates for both the POI and PSN vector are resolved in the camera frame.

5.2.4.1 Estimating Full State of the Target POI.

A constant velocity (double integrator) model is used to describe the dynamics of the POI, with the state vector,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\rho}_{poi} \\ \dot{\boldsymbol{\rho}}_{poi} \end{bmatrix}, \tag{5.11}$$

where $\boldsymbol{\rho}_{poi} = [X, Y, Z]$ is the vector from the camera to the target POI resolved in the camera frame.

The stereo camera sensor and computer vision tracking algorithm provide a measurement of the target POI composed of its pixel location, $\boldsymbol{p}_{poi} = [u, v]^T$, and depth, $\widetilde{Z}$, the EKF measurement vector for the ith image frame is:

$$\boldsymbol{z}_i = \begin{bmatrix} u_i \\ v_i \\ \widetilde{Z}_i \end{bmatrix}. \tag{5.12}$$

Assuming a pinhole camera model and calibrated sensor, the measurement model is a nonlinear function of the centroid state and the camera intrinsic parameters,

$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{K}) = \begin{bmatrix} f_u \dfrac{X}{Z} + c_u \\ f_v \dfrac{X}{Z} + c_v \\ Z \end{bmatrix}, \tag{5.13}$$

where the camera calibration matrix, $\boldsymbol{K}$, contains the camera focal length in the horizontal and vertical directions, f_u and f_v (in units of pixels), and the image principal point, $[c_u, c_v]$.

5.2.4.2 Estimating Full State of PSN Vector.

An LKF is used to estimate the full state of the PSN vector, $\boldsymbol{\rho}_{psn}$. As with the POI, a constant velocity model is used to describe the dynamics of the PSN vector, with the state vector,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\rho}_{psn} \\ \dot{\boldsymbol{\rho}}_{psn} \end{bmatrix}. \tag{5.14}$$

The computed PSN vector is used as the measurement for this filter, so the measurement matrix is simply:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{5.15}$$

As the PSN vector is a unit vector, the PSN vector estimate is normalized prior to being used for control.

5.2.5 Controlling Rotation About the Camera Optical Axis.

Because the method only utilizes a single vector (the PSN vector) and point on the target (the POI) for control, relative rotation of the chaser about the camera optical axis (boresight) is not controllable. Thus, an additional control method must be implemented if a specific relative rotation about the camera optical axis is required.

A simple solution here is to correct the camera optical axis rotation during a maneuver separate from SNVS maneuver. If this maneuver is conducted after the SNVS maneuver, a pure rotation about the camera optical axis is all that is required to achieve the full 6DOF desired relative pose. The IBVS controller proposed in Chapter III of this work provides a straightforward method to command this secondary camera rotation maneuver.

5.3 SNVS Variant 1

This section describes the methods unique to SNVS Variant 1. These are a Linear Quadratic Gaussian (LQG) controller to command translational motion, and a Quaternion Feedback controller to command relative attitude.

5.3.1 Translational Control.

Translational control of the chaser is achieved via the use of a LQG controller, combining a Linear Quadratic Regulator (LQR) with the POI state EKF introduced in Section 5.2.4.1. The LQR controller drives the POI state estimate to the desired state,

$$\boldsymbol{\rho}_{poi}^{des} = \begin{bmatrix} 0 & 0 & Z_{des} & 0 & 0 & 0 \end{bmatrix}^T, \tag{5.16}$$

where Z_{des} is the desired distance from the camera center to the target POI along the camera optical axis. The LQR is augmented with integral action in a similar manner as in [150] in order to reduce steady state error in the presence of measurement uncertainty and aid in convergence to the desired state when the target is moving. To implement integral action, the process model is augmented with an additional state, $\boldsymbol{x}_i$, which represents the integral of the output error,

$$\dot{\boldsymbol{x}}_i = \boldsymbol{y} - \boldsymbol{x}^*, \tag{5.17}$$

where $\boldsymbol{y} = \boldsymbol{C}\hat{\boldsymbol{x}}$ is the estimated position of the POI. Thus the full system model for the LQG controller is

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{x}}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}_{trans} \\ \boldsymbol{y} - \boldsymbol{x}^* \end{bmatrix}. \tag{5.18}$$

which assumes the following state space model for the system

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{0}_{3x3} & \boldsymbol{I}_{3x3} \\ \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} \boldsymbol{0}_{3x3} \\ \boldsymbol{I}_{3x3} \end{bmatrix} \boldsymbol{u}_{trans} \tag{5.19}$$

$$= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}_{trans}. \tag{5.20}$$

The translational control law using this augmented system is then

$$\boldsymbol{u}_{trans} = -\boldsymbol{K}_{lqr}\boldsymbol{x} - \boldsymbol{K}_i\boldsymbol{x}_i, \tag{5.21}$$

where $\boldsymbol{K}_{lqr}$ is the LQR gain and $\boldsymbol{K}_i$ is the integral gain.

The advantage of this LQG configuration, beyond being comparatively straightforward to implement, is that LQR weighting factor and integral gain components can be individually tuned to directly affect control convergence of each relative position state. Note that this controller drives the chaser to a position resolved in the camera frame, meaning that in the absence of attitude control, the chaser may converge to a position anywhere on the surface of a spherical region around the target POI (assuming the POI remains visible). It follows then that the chaser will only achieve the operator-defined desired position with respect to the target POI when the desired attitude with respect to the ROI has been achieved.

5.3.2 Attitude Control.

The nonlinear dynamics of rigid body angular motion and the possibility for large attitude changes during final approach maneuvers limit the usefulness of LQR for attitude

control in this application. Instead, eigenaxis quaternion feedback control is employed to command the chaser's attitude states. Although this type of control is not necessarily time or fuel optimal, it is notable for its ability to control large angle reorientation maneuvers [151]. The control law is implemented as in [152],

$$\boldsymbol{u}_{att} = -\boldsymbol{K}_q \bar{\boldsymbol{q}}_e - \boldsymbol{C}_q \boldsymbol{\omega} + \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}, \tag{5.22}$$

where $\bar{\boldsymbol{q}}_e$ is the vector portion of the attitude error quaternion, $\boldsymbol{q}_e$, between current and desired attitude, $\boldsymbol{\omega}$ is the angular velocity error vector associated with $\boldsymbol{q}_e$ (resolved in the chaser's frame), $\boldsymbol{J}$ is the chaser inertia matrix, and $\boldsymbol{K}_q$ and $\boldsymbol{C}_q$ are control gain matrices. The $\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}$ term is added to counteract the gyroscopic term of Equation (2.21), though in this application control contribution from this term is typically small. As shown by Wie, this controller is globally asymptotically stable if the product $\boldsymbol{K}_q^{-1}\boldsymbol{C}_q$ is chosen to be positive definite [152]. For this work, the control gains are selected as

$$\boldsymbol{K}_q = \alpha^2 \boldsymbol{J}, \tag{5.23}$$

$$\boldsymbol{C}_q = 2\zeta\alpha \boldsymbol{J}, \tag{5.24}$$

where ζ is the damping factor, and α is defined as

$$\alpha = \sqrt{\tau_{max} J_{min}}, \tag{5.25}$$

where τ_{max} is a tunable parameter that roughly corresponds to the maximum possible torque output of the controller and J_{min} is the chaser's minimum principal moment of inertia.

5.3.2.1 Calculating the Desired Pointing Vector.

A basic implementation of this SNVS variant was described in a previous work which utilizes the PSN vector exclusively as the desired pointing vector for the camera boresight, $\hat{\boldsymbol{e}}_{des}$ [153]. However, there is a possible degeneracy with this implementation where the

target POI leaves the camera FoV if the chaser rotates to achieve the desired pointing vector faster than it translates to compensate for the rotational motion. As the target POI must stay in the camera FoV, this section describes a method to generate $\hat{\boldsymbol{e}}_{des}$ that ensures the POI will remain in the camera FoV during maneuvering. The method described here is similar to a procedure for keeping features in the camera FoV during Partitioned Visual Servoing described by Corke in [116].

In order to keep the POI in the FoV, two vectors are intelligently combined to generate $\hat{\boldsymbol{e}}_{des}$. The first vector, $\hat{\boldsymbol{e}}_{PSN}$, is the PSN vector, and represents the ultimate desired point direction of the camera boresight upon completion of the maneuver. The second vector, $\hat{\boldsymbol{e}}_{POI}$, is the vector from the camera to the target POI, and is used to keep the POI in the FoV. A linear interpolation is used to combine these two vectors, with the parameter $\beta serving to specify how much each vector is prioritized$,

$$\hat{\boldsymbol{e}}_{des} = \frac{(1-\beta)\hat{\boldsymbol{e}}_{PSN} + \beta\hat{\boldsymbol{e}}_{POI}}{||(1-\beta)\hat{\boldsymbol{e}}_{PSN} + \beta\hat{\boldsymbol{e}}_{POI}||}. \tag{5.26}$$

The parameter β should prioritize $\hat{\boldsymbol{e}}_{PSN}$ if the POI is near the center of the FoV, and $\hat{\boldsymbol{e}}_{POI}$ if the POI is near an edge of the FoV. Thus, β is defined as,

$$\beta = \begin{cases} 0 & : \rho(u,v) > \rho_{max} \\ 1 & : \rho(u,v) < \rho_{min} \\ \left(\dfrac{\rho(u,v) - \rho_{max}}{\rho_{min} - \rho_{max}}\right)^{\chi} & : \rho_{min} < \rho(u,v) < \rho_{max} \end{cases}, \tag{5.27}$$

where $\rho(u,v) = min\{u, v, N_u - u, N_v - V\}$, N_u is the horizontal size of the image, N_v is the vertical size of the image, and ρ_{min}, ρ_{max}, χ are tunable parameters used to define conditions for prioritizing each vector. ρ_{min} and ρ_{max} define the minimum and maximum

image boundaries from which β transitions from 1 to 0, and χ is a shaping parameter that defines how fast this transition occurs. Figure 5.10 depicts these parameters graphically.

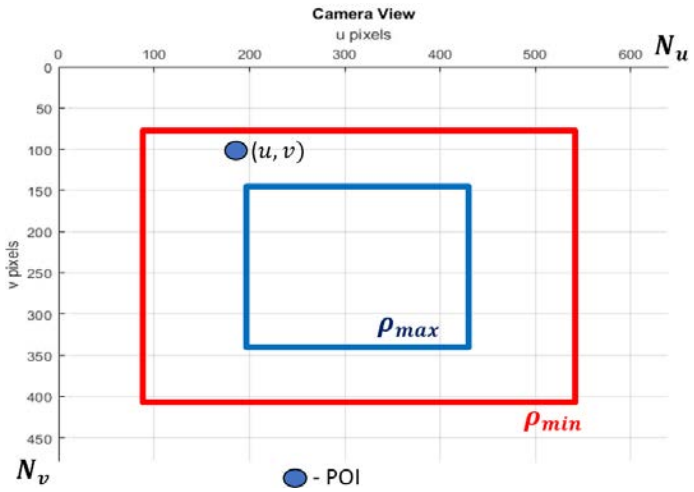

Figure 5.10: Depiction of parameters associated calculation of β

5.3.2.2 Calculating the Error Quaternion.

Once the desired pointing unit vector $\hat{\boldsymbol{e}}_d$ has been determined, this vector is used to generate the error quaternion, $\boldsymbol{q}_e$, for the quaternion feedback controller. The boresight vector of the chaser camera sensor in the chaser body frame is

$$\hat{\boldsymbol{e}}_b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{5.28}$$

$\boldsymbol{q}_e$ thus represents the error quaternion between $\hat{\boldsymbol{e}}_b$ and $\hat{\boldsymbol{e}}_d$. To derive an expression for $\boldsymbol{q}_e$, consider the following trigonometric identities:

$$\sin(\theta) = \sqrt{1 - \cos^2(\theta)}, \tag{5.29}$$

$$\sin\left(\frac{\theta}{2}\right) = \pm\sqrt{\frac{1-\cos(\theta)}{2}}, \tag{5.30}$$

$$\cos\left(\frac{\theta}{2}\right) = \pm\sqrt{\frac{1+\cos(\theta)}{2}}. \tag{5.31}$$

The vector perpendicular to $\hat{\boldsymbol{e}}_b$ and $\hat{\boldsymbol{e}}_d$ is

$$\hat{\boldsymbol{e}}_\perp = \frac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\|\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d\|} = \frac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\sin(\theta)}, \tag{5.32}$$

where $\theta = \cos^{-1}(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)$ is the angle between $\hat{\boldsymbol{e}}_b$ and $\hat{\boldsymbol{e}}_d$. $\hat{\boldsymbol{e}}_\perp$ and θ are the Euler axis and angle to rotate $\hat{\boldsymbol{e}}_b$ to $\hat{\boldsymbol{e}}_d$. Using Eq. (5.29), $\hat{\boldsymbol{e}}_\perp$ can be reformulated

$$\begin{aligned}
\hat{\boldsymbol{e}}_\perp &= \frac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\sin(\cos^{-1}(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d))} \\
&= \frac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\sqrt{1-\cos^2(\cos^{-1}(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d))}} \\
&= \frac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\sqrt{1-(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)^2}} \\
&= \frac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\sqrt{1-(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}\sqrt{1+(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}}.
\end{aligned} \tag{5.33}$$

An expression for $\boldsymbol{q}_e$ can be defined using the well-known relation between Euler axis/angles and quaternions. The vector portion of $\boldsymbol{q}_e$ is (simplified using Eq. (5.30))

$$\begin{aligned}
\bar{\boldsymbol{q}}_e &= \hat{\boldsymbol{e}}_\perp \sin\left(\frac{\theta}{2}\right) \\
&= \frac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\sqrt{1-(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}\sqrt{1+(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}} \sin\left(\frac{\cos^{-1}(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}{2}\right) \\
&= \frac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\sqrt{1-(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}\sqrt{1+(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}} \sqrt{\frac{1-(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}{2}} \\
&= \frac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\sqrt{2(1+\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}}.
\end{aligned} \tag{5.34}$$

The scalar part of $\boldsymbol{q}_e$ can be formulated using Eq. (5.31)

$$\begin{aligned} q_{e_4} &= \cos\left(\frac{\theta}{2}\right) \\ &= \cos\left(\frac{\cos^{-1}(\hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}{2}\right) \\ &= \sqrt{\frac{1 + \hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d}{2}}. \end{aligned} \tag{5.35}$$

The full expression for the error quaternion is therefore

$$\boldsymbol{q}_e = \begin{bmatrix} \dfrac{\hat{\boldsymbol{e}}_b \times \hat{\boldsymbol{e}}_d}{\sqrt{2(1 + \hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d)}} \\ \\ \sqrt{\dfrac{1 + \hat{\boldsymbol{e}}_b \cdot \hat{\boldsymbol{e}}_d}{2}} \end{bmatrix}, \tag{5.36}$$

where the vector part is used by the quaternion feedback controller to command the camera boresight vector to the desired pointing vector.

5.3.2.3 Estimating the Angular Velocity Term of Quaternion Feedback Controller.

The second and third terms of the quaternion feedback control law, Equation (5.22), include an angular velocity term, $\boldsymbol{\omega}$. This is the angular velocity associated with $\boldsymbol{q}_e$, and must be estimated online. This is accomplished using a discrete derivative of $\boldsymbol{q}_e$ and quaternion kinematics,

$$\boldsymbol{\omega} = 2 \begin{bmatrix} q_4 & q_3 & -q_2 & -q_1 \\ -q_3 & q_4 & q_1 & -q_2 \\ q_2 & -q_1 & q_4 & -q_3 \end{bmatrix} \dot{\boldsymbol{q}}_e, \tag{5.37}$$

where $\boldsymbol{q}_e = [q_1, q_2, q_3, q_4]^T$.

5.4 SNVS Variant 2

The overall architecture of SNVS Variant 2 is generally similar to that of Variant 1, with the exception that the translational and attitude controllers operate on different control errors. This section describes these controllers; all other methods of SNVS Variant 2 function as described previously.

The most significant difference of Variant 2 is that it does not require the chaser to achieve the desired relative attitude prior to achieving the desired relative position. This may be relevant for certain applications, hence it is presented here as an alternative to SNVS Variant 1.

5.4.1 Translational Control.

As with Variant 1, SNVS Variant 2 utilizes an LQG controller augmented with integral action to command translational motion. The same system matrix is utilized for this implementation, however the control error is defined differently,

$$\boldsymbol{e} = \frac{\hat{\boldsymbol{X}}_{POI}}{Z_{des}} - \hat{\boldsymbol{X}}_{PSN}, \tag{5.38}$$

where $\hat{\boldsymbol{X}}_{POI}$ and $\hat{\boldsymbol{X}}_{PSN}$ are the POI and PSN state estimates respectively, and Z_{des} is the desired relative depth (distance along the optical axis) from the camera to the POI. Recall both $\hat{\boldsymbol{X}}_{POI}$ and $\hat{\boldsymbol{X}}_{PSN}$ are resolved in the camera frame and will change over the course of the maneuver. The scaling of $\hat{\boldsymbol{X}}_{POI}$ by $1/Z_{des}$ is necessary to ensure the chaser ends the maneuver at the desired distance from the target.

5.4.2 Attitude Control.

Attitude control for SNVS Variant 2 is straightforward: the only task of the attitude controller is to drive the POI to a desired position in the camera FoV. As with SNVS Variant 1, this controller regulates pan and tilt of the camera, but not rotation about the camera optical axis. Proportional-integral-derivative (PID) compensation is sufficient for this control task, with one controller each implemented for pan and tilt of the camera.

The control error for these compensators is defined as the difference between the current pixel location of the POI and the desired location in the horizontal and vertical directions, respectively.

Unlike SNVS Variant 1, attitude control error must not be regulated to zero in order to achieve the desired relative position. Like SNVS Variant 1, operation of this controller does have an affect on the overall translational trajectory and fuel use of the the chaser, however to a far lesser extent if the attitude controller is well-tuned. In testing it was observed that oscillation around the PID setpoint can degrade the translational trajectory and increase fuel use. Thus, it is important to tune the PID gains to minimize oscillation. In addition to tuning, a deadband can be implemented to further prevent oscillation around the setpoint. A deadband of five pixels is used for the results of this chapter, whereupon PID control error is set to zero if the absolute value of the actual error is less than five pixels.

5.5 Results

5.5.1 Simulation.

This section presents results for several scenarios representative of possible RPO approach maneuvers. The simulation is similar to that used for Chapter III (see Appendix A for overview), with some modifications to implement ROI/POI tracking. The target ROI is the same nonplanar ROI used for simulation results in Chapter III (see Figure 3.24). There is no need to consider a planar ROI as it is clear that if the method can estimate the correct PSN vector for a nonplanar ROI, then it could estimate a PSN vector for a planar ROI as well. The simulation terminates when the chaser achieves a position within 0.05 meters of the goal relative position in all dimensions and a relative velocity of less than 0.005 m/s in all dimensions.

Results for both SNVS variants are overlaid to facilitate comparison. In addition, for Scenario 5NP1, results using the IBVS method presented in Chapter III are overlaid to demonstrate that method in relation to SNVS. In order to enable as direct a comparison as

possible, each methods' control gains are tuned to give an approximately 200 second time of flight for the first scenario, 5NP1, then these gains are used for subsequent scenarios. Controller tuning has a significant effect on maneuver outcomes such as chaser trajectory, time of flight, and control effort, so differences between each variant presented here should not be considered conclusive for all possible conditions. Rather, comparison of the variants and with the IBVS method is presented to motivate the concept that each framework produces different results for given initial conditions, which may be more or less advantageous depending on the application. Further comparison and tuning should be conducted in order to determine which method is best-suited for a specific use-case.

5.5.1.1 POI/ROI Tracking and Computing the PSN Vector.

The POI is the geometric center of the target ROI (top face of the target spacecraft). A perfect measurement of the POI pixel coordinates are provided to the EKF used for POI tracking at 2 Hz. The PSN vector is computed at 2 Hz as described in Section 5.2.2.2, using 10,000 randomly selected sets of points from the target ROI and 200 histogram bins. A perfect depth map is used for this computation with no measurement error applied. Rather than tracking the predominant surface plane as described in Section 5.2.3.2 (note the method introduced in Section 5.2.3.2 is demonstrated in Chapter VI), the four corners of the top face of the target spacecraft are used to construct a rectangular bounding box that is used as the region from which to sample points for PSN vector computation. The sides of this bounding box are horizontally/vertically oriented with the image plane, as if a basic single-object tracker were being used to track the top face of the target. Figure 5.11 shows an example camera view and depth map from the simulation depicting the POI and ROI bounding box.

While the method for computing the PSN vector does induce some randomness due to the way points are sampled, no other stochastic sensor measurements are utilized for these results. This is intentional to demonstrate the method's operation under ideal conditions

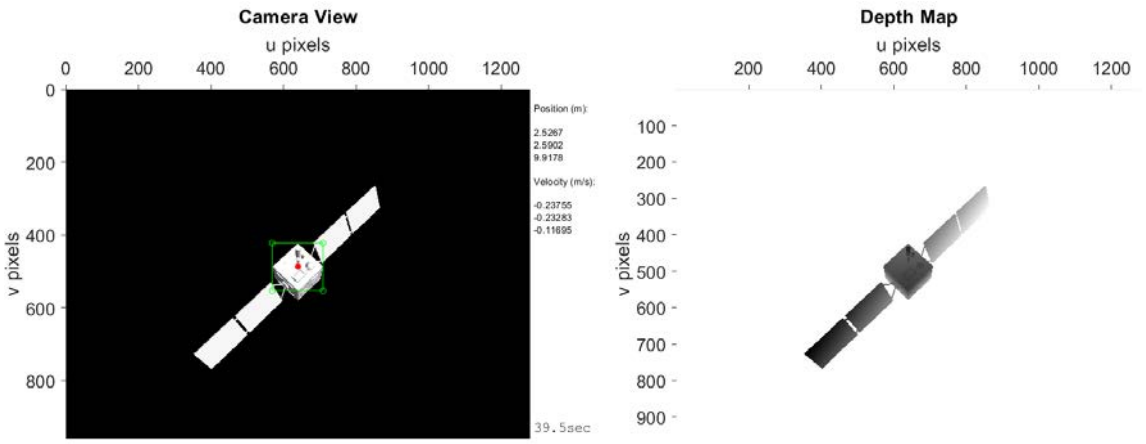

Figure 5.11: Example chaser camera image and depth map depicting target POI and ROI bounding box.

and to facilitate comparison between each variant. Chapter VI of this work demonstrates each SNVS variant's operation under more realistic sensor conditions using actual camera hardware.

5.5.1.2 Quantifying Control Effort.

To add in comparison of the two variants, the results in this chapter quantify total control effort for each scenario presented. Since translational and attitude control is decoupled, total control effort for each controller is calculated separately. For translational control, Hydrazine thrusters with a propellant specific impulse (I_{sp}) of 200 seconds are assumed. Propellant use is calculated by integrating the mass flow rate of propellant ($\dot{m}_p$) over the course of the maneuver,

$$\dot{m}_p = \frac{F_t}{I_{sp} g_0}, \tag{5.39}$$

where F_t is the total force produced by all thrusters and g_0 is standard gravity. Attitude control effort is quantified by the integral

$$\mathrm{T} = \int_0^t \sum_{i=1}^{3} \left| \frac{\tau(i)}{\boldsymbol{J}_c(i,i)} \right| dt, \tag{5.40}$$

where $\tau(i)$ is the ith component of the applied torque vector and $\boldsymbol{J}_c(i,i)$ is the ith principal moment of inertia. This integral is taken over the course of the maneuver and is useful for relative comparison of total attitude control effort required by each method.

5.5.2 5NP1: Maneuver with Respect to Static Nonplanar ROI.

The first scenario examined is similar to Scenario 3NP2: an angled approach maneuver with respect to a target ROI containing significant nonplanar features. The target remains static at the LVLH origin and the chaser is initially positioned at $[10.0, 10.0, 15.0]^T$ meters, with its camera pointing at the target POI. The chaser's goal is to maneuver to a position 2 meters from the POI and achieve an attitude orthogonal to the predominant surface plane of the ROI (this position corresponds to $[0.0, 0.0, 2.5]^T$ meters in the Local-Vertical-Local-Horizontal (LVLH) frame). Results for both SNVS variants are presented alongside results generated using Chapter III's IBVS method. To facilitate a more direct comparison between the IBVS method and SNVS, stochastic sensor measurements and control torque about the camera optical axis are not applied for the IBVS method, as is the case for the SNVS results. Figure 5.12 shows the LVLH frame trajectory of the chaser along with the initial and final camera pose for each method.

Both SNVS variants capture the desired position precisely within about 200 seconds, each taking a slightly different trajectory to arrive at the desired pose. As expected, the IBVS method fails to capture the desired relative pose due to the nonplanar nature of the target ROI and the use of the warp method to generate the goal image (incremental maneuvering was not implemented for these results). Figure 5.13 shows the relative velocity components for each method. SNVS Variant 1 generally exhibits the smoothest and lowest magnitude velocity curves, while Variant 2 exhibits sharper changes in velocity.

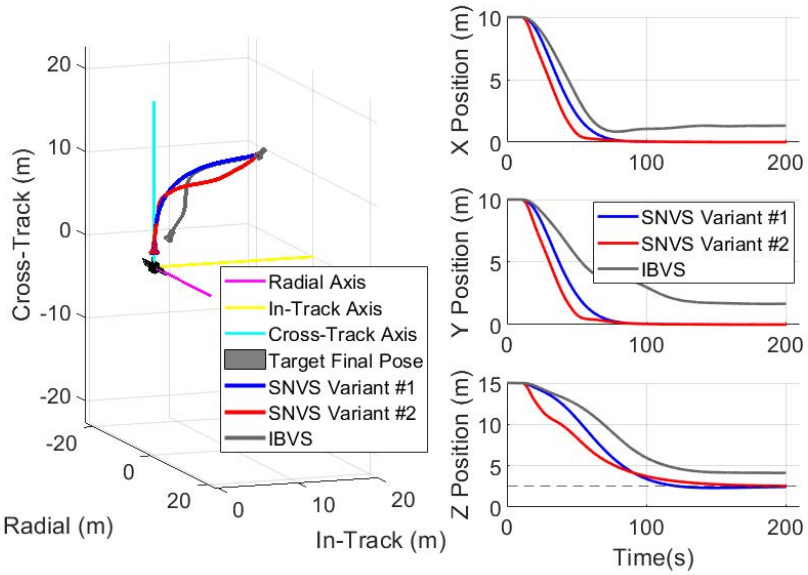

Figure 5.12: Scenario 5NP1: Chaser trajectory (LVLH frame).

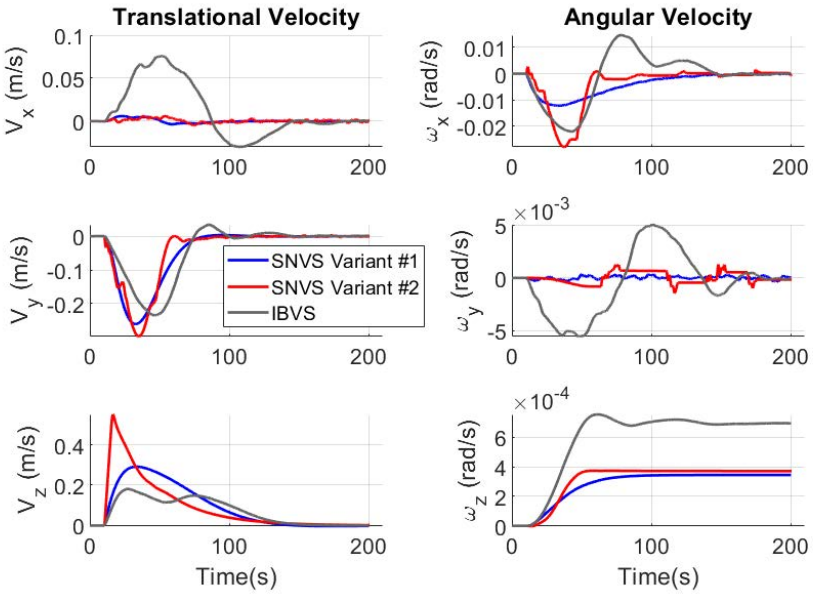

Figure 5.13: Scenario 5NP1: Translational and angular velocity of chaser with respect to target (resolved in camera frame).

Thrust applied to each positive and negative body frame direction and torque applied about each body frame axis is shown in Figure 5.14. Variant 2's thrust profiles exhibit far more chatter than Variant 1 and the IBVS method. This is a result of the fact that Variant 2's translational controller tracks the noisy PSN vector state estimate. Importantly, it was observed during estimator tuning that Variant 2's filter process covariance for the PSN vector cannot be slowed (in order to smooth the PSN vector state estimate) to the same extent as for Variant 1 without causing control instability. Thus, Variant 2's translational controller must use a noisier PSN vector state estimate, which leads to more chatter in its thrust profile. Variant 2 also exhibits sharp transitions in torque (and hence angular velocity). These rapid changes in torque are caused by attitude control error exceeding the PID deadband.

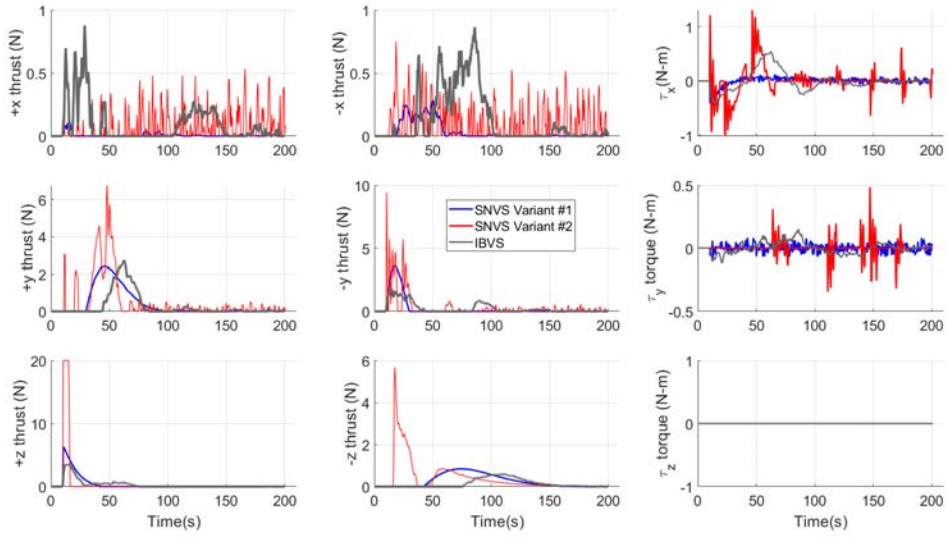

Figure 5.14: Scenario 5NP1: Translational thrust and attitude control torque.

Filter estimation error for the PSN vector is depicted in Figure 5.15. The error remains small—a component error of 5e−3 corresponds to 0.29° error from the actual PSN vector. However, perfect depth maps are utilized in the simulation, so these results are likely better

than could be expected in reality. Even so, these results demonstrate the ability of the method to correctly estimate the PSN vector over the entire maneuver.

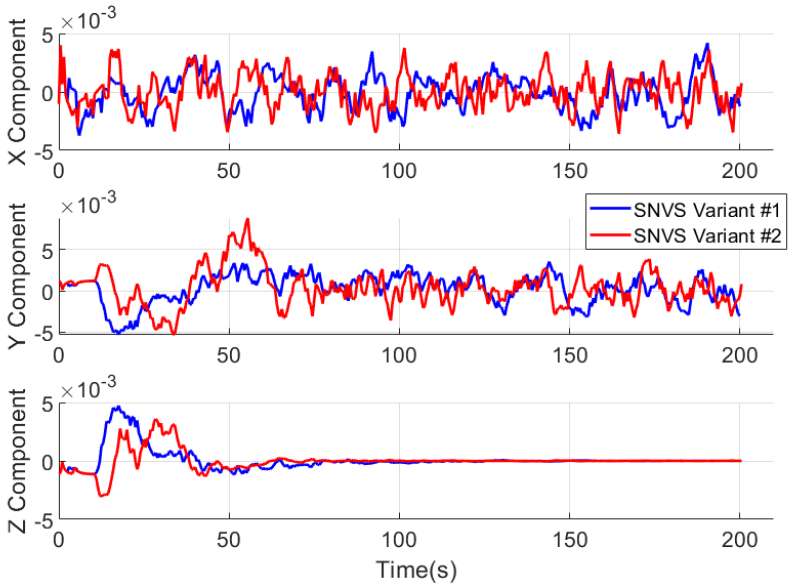

Figure 5.15: Scenario 5NP1: PSN vector estimation error (resolved in camera frame).

Finally, Table 5.1 summarizes the results for this scenario, including data for propellant use and total attitude control effort of each method. While the IBVS method uses the least propellant, it ends the maneuver more than 2 meters away from the desired relative pose. SNVS Variant 1 and 2 are comparable for maneuver time and final relative pose, but Variant 1 uses significantly less control effort.

Table 5.1: Scenario 5NP1 Summary of Results.

Method	Maneuver Time (s)	Final Relative Position (m)$^{+}$	Roll/Pitch/Yaw (deg)$^{++}$	Propellant Used (kg)	Attitude Control Effort (s^{-2})
SNVS V1	200	[0.00, 0.00, 1.95]	[-0.07, 0.02, 131.95]	0.126	0.055
SNVS V2	200.5	[0.00, 0.01, 2.05]	[-0.27, 0.06, 131.67]	0.201	0.106
IBVS	200	[1.32, 1.66, 4.13]	[28.81, 7.13, 129.65]	0.119	0.109

+ Resolved in camera frame (desired positon is [0, 0, 2] m)
++ Angles about camera axes relative to PSN vector (desired is [0, 0, N/A] deg)

5.5.3 5NP2: Maneuver with Respect to Tumbling Nonplanar ROI.

Scenario 5NP2 is identical to that of 5NP1, with the exception that the target is given an initial angular and translational velocity with respect to the LVLH frame. The target's initial angular velocity is $[0.000, -0.003, 0.005]^T$ rad/second and translational velocity is $[0.010, 0.005, 0.001]^T$ meters/second.

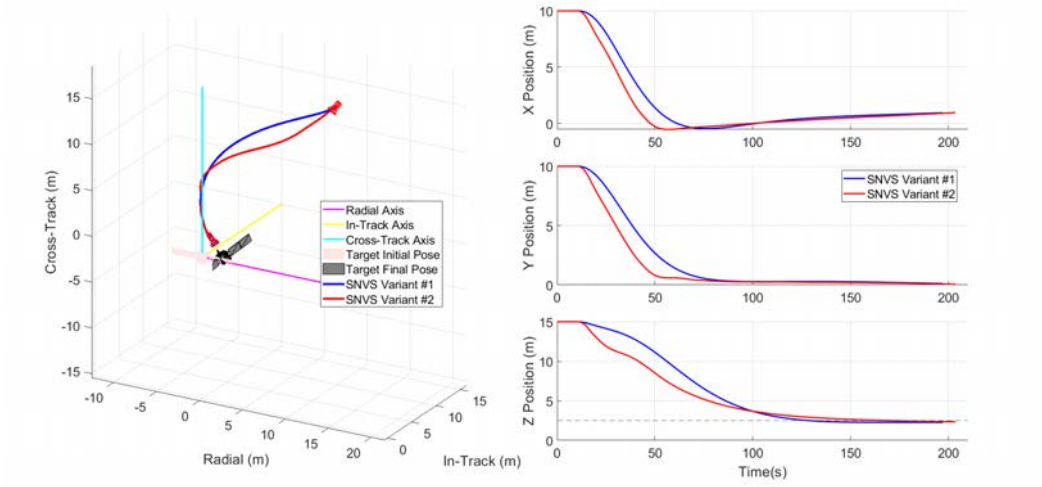

Figure 5.16: Scenario 5NP2: Chaser trajectory (LVLH frame).

Figure 5.17 depicts the relative translational and angular velocity. All components of these velocities converge on zero as the Chaser captures the desired relative pose, with the exception of angular velocity about the camera optical axis (ω_z). As stated previously, the

degrees of freedom (DOF) associated with ω_z is not controlled by the SNVS method, and would need to be controlled separately (using e.g. the IBVS method of Chapter III). Plots depicting control effort and PSN vector estimation error resemble those of Scenario 5NP1 and are not shown for conciseness.

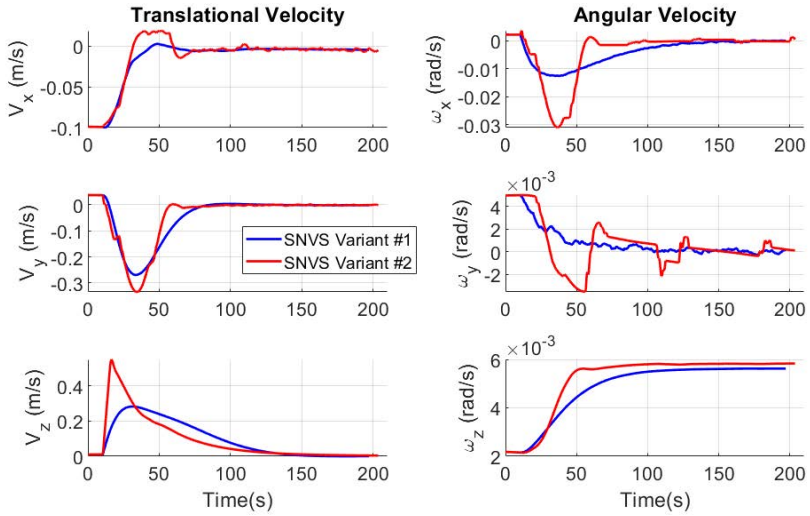

Figure 5.17: Scenario 5NP2: Translational and angular velocity of chaser with respect to target (resolved in camera frame).

Table 5.2 presents the time of flight, final pose, and total control effort for each variant. As with Scenario 5NP1, Variant 1 uses significantly less control to achieve the same final pose in roughly the same amount if time. Overall the results of this scenario strongly resemble those of 5NP1, indicating that the navigational and control functions of this method operate similarly regardless of relative target motion.

Table 5.2: Scenario 5NP2 Summary of Results.

Method	Maneuver Time (s)	Final Relative Position (m)[+]	Roll/Pitch/Yaw (deg)[++]	Propellant Used (kg)	Attitude Control Effort (s^{-2})
SNVS V1	197	[0.00, 0.00, 1.95]	[-0.15, 0.46, 77.03]	0.142	0.056
SNVS V2	203.5	[0.00, 0.01, 2.05]	[-0.11, 0.00, 71.67]	0.216	0.102

+ Resolved in camera frame (desired positon is [0, 0, 2] m)
++ Angles about camera axes relative to PSN vector (desired is [0, 0, N/A] deg)

5.5.4 5NP3: Extreme Off-Approach-Axis Maneuver with Respect to Static Nonplanar ROI.

The final scenario examined involves an approach maneuver with a static nonplanar target where the chaser's initial relative attitude is significantly different than the desired relative attitude. The intent of this scenario is to examine (1) the effectiveness of the PSN vector computation method when the target ROI is observed from a steep angle and (2) the control method's ability to control a large slew maneuver. The chaser is initially positioned at $[4.0, 4.0, 2.0]^T$ meters in the LVLH frame, with the camera pointing at the target POI. This pose makes the camera boresight axis approximately 70.5° off from the PSN vector at the start of maneuvering. Figure 5.19 depicts the chaser's initial view of the target.

Figure 5.19 depicts the chaser trajectory in the LVLH frame for both variants. Notably, both methods do not initially translate directly towards the POI. Doing so would exacerbate the steep view angle, which would likely make accurate computation of the PSN vector more difficult. Instead, for both variants the chaser follows an arcing translational trajectory, placing itself on an approach path more in line with the PSN vector earlier in the maneuver. Differential tuning of the translational and attitude controllers can be used to affect the shape of the trajectory, as will be seen in Section 5.5.5.

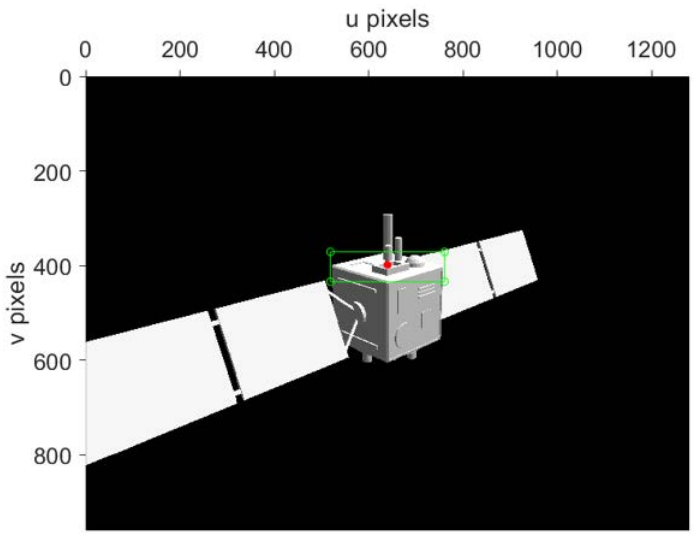

Figure 5.18: Scenario 5NP3: Initial Chaser Camera View.

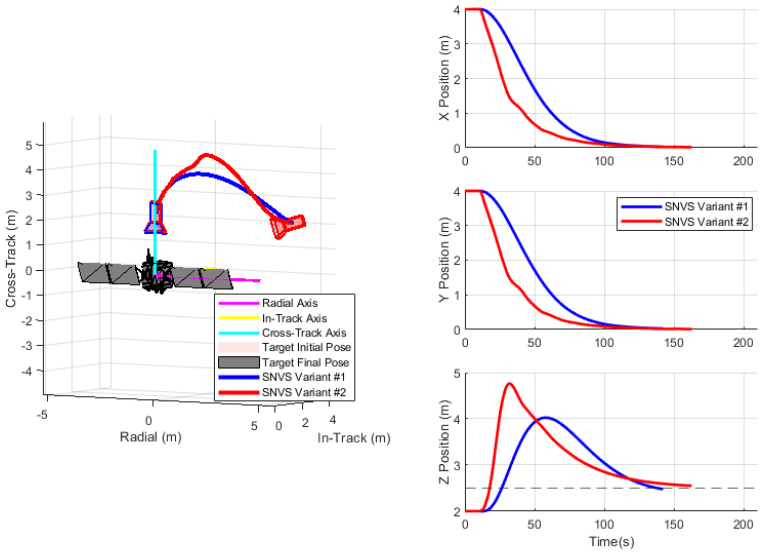

Figure 5.19: Scenario 5NP3: Chaser trajectory (LVLH frame).

The velocity of the chaser with respect to the target is shown in Figure 5.20. As with previous scenarios, the velocity profiles generated by Variant 2 are more erratic and of higher magnitude.

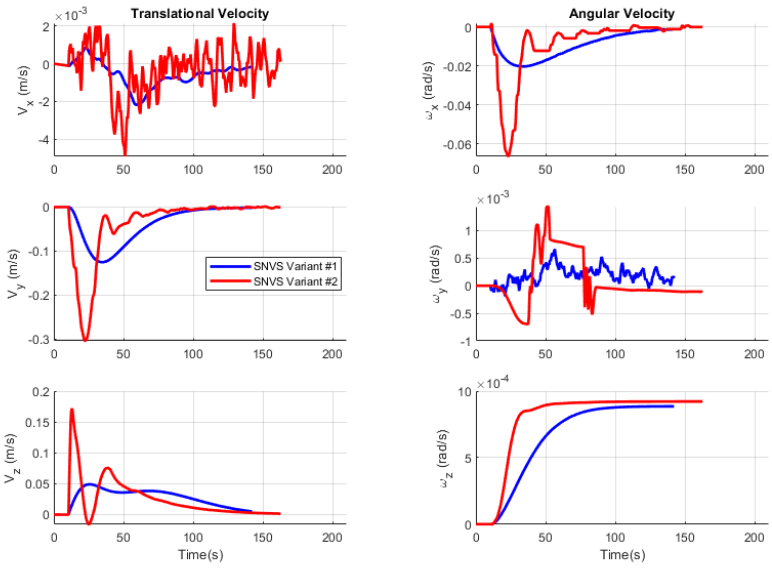

Figure 5.20: Scenario 5NP3: Translational and angular velocity of chaser with respect to target (resolved in camera frame).

Figure 5.21 shows the PSN vector component error for the estimated PSN vector relative to the true PSN vector, resolved in the camera frame. Early in the maneuver the error is worse than that of Scenario 5NP1, which is not surprising given the steeper view angle of the target ROI. The worst error of the scenario corresponds to less than 1.0° of angular relative to the actual PSN vector.

Table 5.3 summarizes the results for this scenario, which confirm the trends seen in the previous results: while the two variants reach the desired pose in similar time, Variant 2 requires more control effort for both translational and attitude maneuvering. Additionally, for this scenario, Variant 2 yields a longer maneuver time relative to Variant 1.

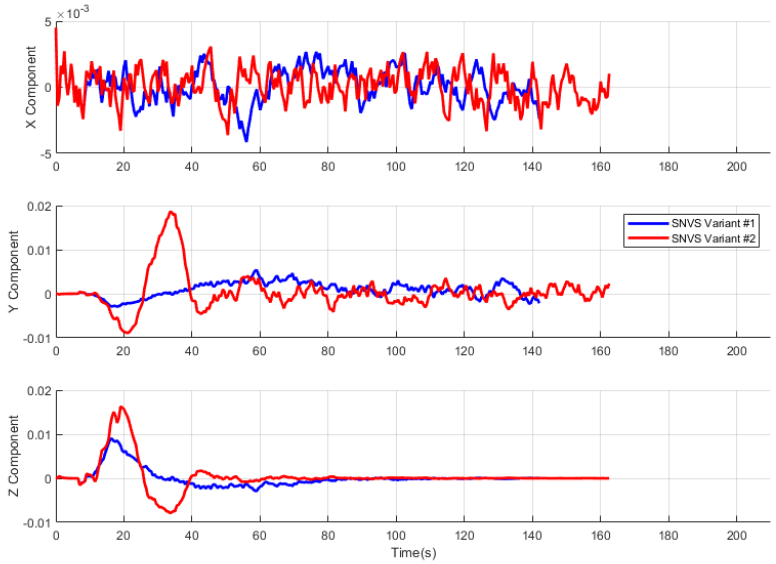

Figure 5.21: Scenario 5NP3: PSN vector estimation error (resolved in camera frame).

Table 5.3: Scenario 5NP3 Summary of Results

Method	Maneuver Time (s)	Final Relative Position (m)[+]	Roll/Pitch/Yaw (deg)[++]	Propellant Used (kg)	Attitude Control Effort (s^{-2})
SNVS V1	142.0	[0.00, 0.01, 1.98]	[1.62, 0.07, 129.77]	0.044	0.054
SNVS V2	162.5	[0.00, 0.00, 2.05]	[0.52, 0.20, 127.86]	0.143	0.19

+ Resolved in camera frame (desired positon is [0, 0, 2] m)
++ Angles about camera axes relative to PSN vector (desired is [0, 0, N/A] deg)

5.5.5 Sensitivity Analysis: Effect of Control Gains.

This section examines how changing control gains for each variant affects maneuver parameters including trajectory, time of flight, translational thruster propellant use, and total attitude control effort. Since the spacecraft and approach scenarios are notional, the results presented below are not comprehensive, and are therefore not intended to be used to determine optimal gain parameters. Rather, they are meant to demonstrate that differential control gain selection may have a non-intuitive effect on relevant maneuver parameters,

which should motivate future work to consider implementing optimization techniques on the method in order to maximize utility for particular use-cases.

5.5.5.1 Variant 1 Sensitivity Analysis Results.

The relationship between translational and attitude control for Variant 1 makes the effect of altering control gains on the chaser's trajectory not necessarily intuitive. Even though translational and rotational dynamics are assumed to be decoupled, changes in angular velocity induce a change in the translational trajectory of the chaser. This is due to the fact that the translational controller is operating on error resolved in the camera frame, so changes in attitude affect translational control error. To illustrate this, trajectories for three scenarios are shown in Figure 5.22. Each scenario represents an approach maneuver with a static target using identical initial conditions, with the exception that the attitude controller parameter τ_{max} is scaled 0.5x, 1x, and 5x, respectively. As can be seen, even though the translational control gains remain constant for all iterations, the translational trajectory of the camera changes drastically as the gains for the attitude controller are altered.

Table 5.4 shows the maneuver time, relative total translational control effort, and relative total attitude control effort for these three scenarios. Not only does increasing attitude control gains increase total attitude control effort required, but also increases maneuver time and total translational control effort.

Table 5.4: Effect of Attitude Control Gain on Chaser Trajectory, SNVS Variant 1

τ_{max} Gain Factor	Maneuver Time (s)	Integral of Translational Control Effort (relative)	Integral of Attitude Control Effort (relative)
0.5	134.5	0.805	0.659
1	139	1.000	1.000
5	197.5	1.945	4.629

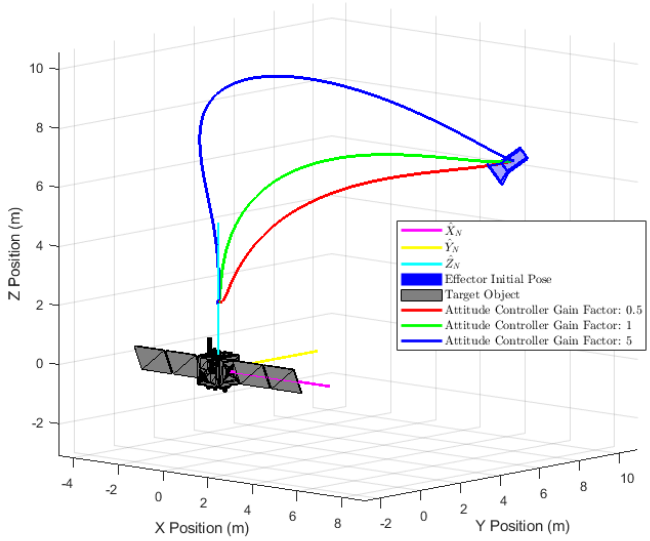

Figure 5.22: 3D trajectories of chaser for various attitude control gain factors (SNVS Variant 1).

The ability to shape the trajectory of the chaser using differential control gain tuning is advantageous. For example, it is comparatively simple to prioritize rotation to achieve a relative attitude parallel to the PSN vector over translation towards the desired relative position by changing a single control gain of the quaternion feedback controller (τ_{max}).

Figure 5.23 displays results for a larger sensitivity analysis of Variant 1. This analysis alters two gain factors: a scalar multiplier on the LQR performance weighting matrix, and a scalar multiplier for the attitude controller parameter τ_{max}. There is certainly more analysis that could be conducted by altering the control gains in different ways, but several useful observations are still apparent. Reducing the attitude control gain factor reduces both total attitude control effort and translational control effort (propellant used). The translational control gain factor has little effect on total attitude control effort. There is a relatively flat region on the surface corresponding to the time of flight, indicating that there is a

large range where control gains could be selected to optimize control effort while causing comparatively little change to the time of flight.

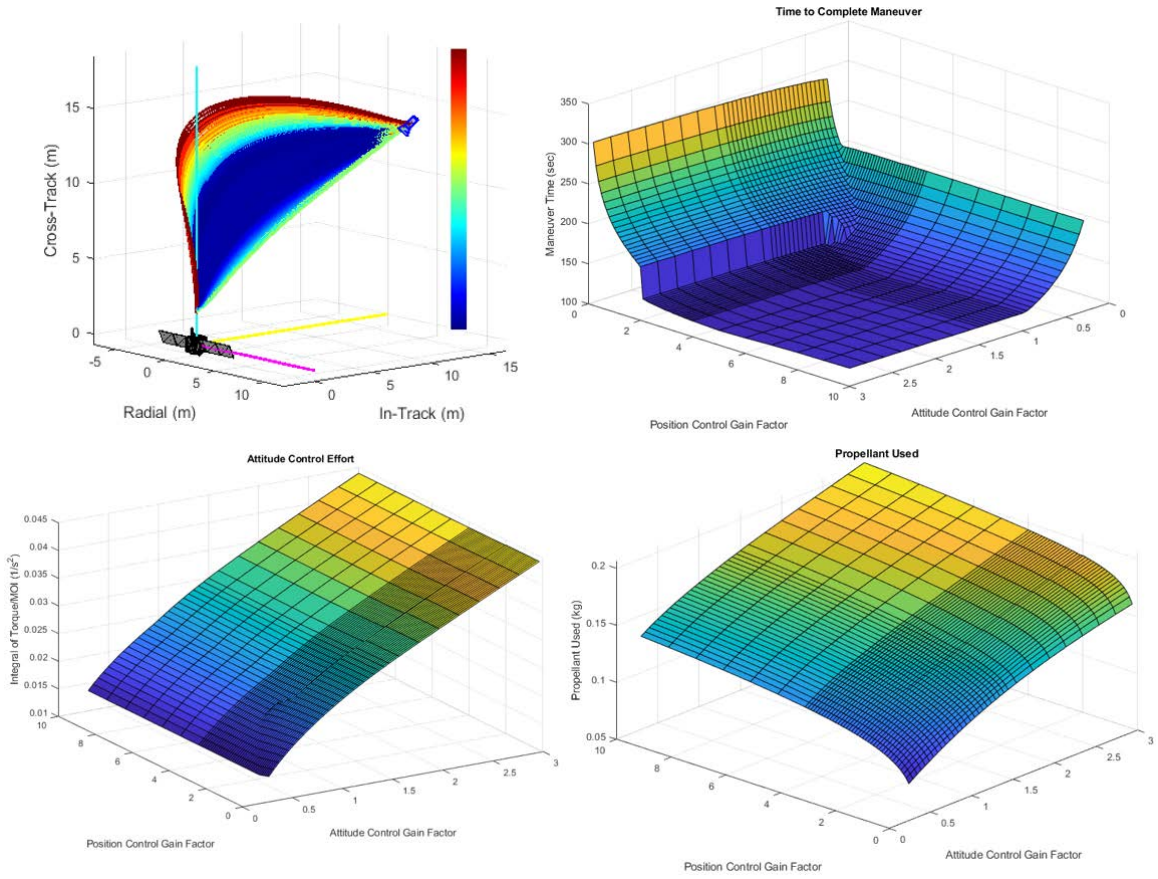

Figure 5.23: SNVS Variant 1 sensitivity analysis results. Top left: chaser trajectories colored according to time of flight (dark red is maximum, dark blue is minimum). Top right: time of flight vs. control gains. Bottom left: total attitude control effort vs. control gains. Bottom right: propellant used vs. control gains.

5.5.5.2 Variant 2 Sensitivity Analysis Results.

Unlike Variant 1, the sensitivity analysis conducted for SNVS Variant 2 only varies the LQR performance weighting matrix and not any gains associated with attitude control. Since the task of the attitude controller is to simply keep the target POI in the center of the

camera FoV, it is thought that once the attitude controller is independently tuned to achieve a desired control response, it should not need to be altered further.

The analysis for this variant examines how two scalar gain factors applied to the LQR performance weighting matrix affect maneuver parameters. One gain factor is multiplied by the LQR performance weighting matrix position states, and the other gain factor is multiplied by the velocity states. Results for the analysis are shown in Figure 5.24.

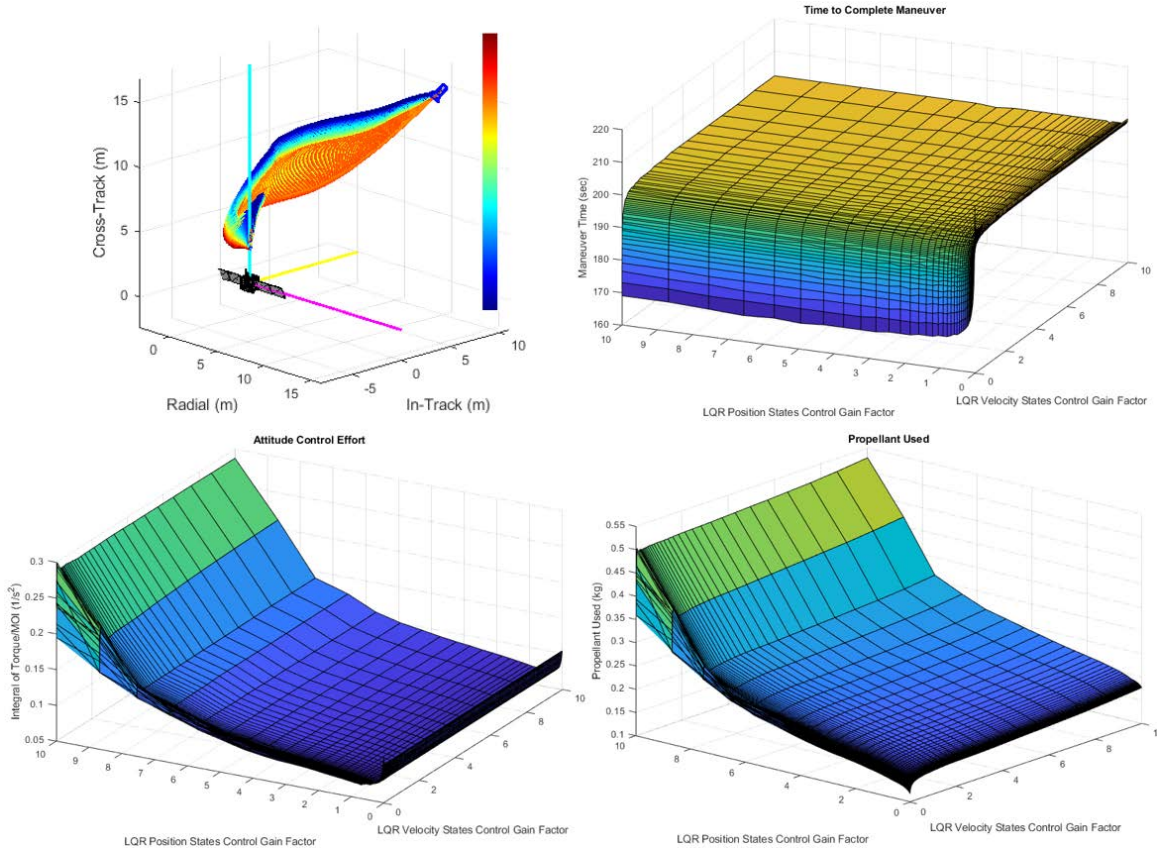

Figure 5.24: SNVS Variant 2 sensitivity analysis results. Top left: chaser trajectories colored according to time of flight (dark red is maximum, dark blue is minimum). Top right: time of flight vs. control gains. Bottom left: total attitude control effort vs. control gains. Bottom right: propellant used vs. control gains.

The results show that reducing the velocity state gain factor below 1 reduces time of flight, but increasing it by an order of magnitude does not significantly affect the time of

flight. The velocity state gain factor does not significantly affect total attitude control effort, but it does decrease propellant use as it decreases. Altering the position state gain factor does not significantly affect time of flight for the range of gains analyzed, but it does more significantly affect both translational and attitude control effort that the velocity state gain factor.

5.6 Discussion

5.6.1 Effect of Initial POI Image Location on Translational Trajectory.

An important characteristic of each SNVS variant pertains to how the translational trajectory of the chaser is affected by the camera's initial pointing direction with regards to the target POI, i.e. if the camera boresight is initially centered on the target POI or not. Figure 5.25 shows example trajectories for both variants with and without the camera initially centered on the target. Both variants' trajectories are affected if the POI is initially non-centered, however Variant 2's trajectory is far more affected than Variant 1's.

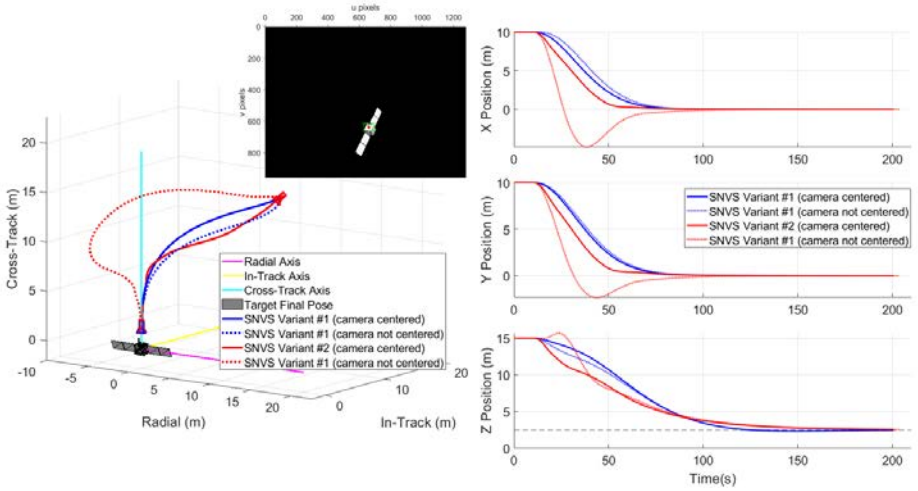

Figure 5.25: Example of chaser trajectories with and without the chaser camera initially centered on the target. Inset camera image shows the initial chaser view for maneuvers with the camera not initially centered.

Though the trajectories change with the POI initially non-centered, the maneuver times are relatively unaffected. Total control effort is significantly affected, especially for Variant 2. These results are summarized in Table 5.5.

Table 5.5: Control Effort Comparison for Maneuvers with and without Camera Initially Centered on Target

Method	Maneuver Time (s)	Propellant Used (kg)	Attitude Control Effort (s^{-1})
SNVS V1 (camera centered)	200.5	0.125	0.025
SNVS V1 (camera not centered)	200	0.148	0.038
SNVS V2 (camera centered)	200.5	0.177	0.0753
SNVS V2 (camera not centered)	203.5	0.497	0.277

Clearly, having the target POI initially non-centered at the start of translational maneuvering is not ideal, especially for SNVS Variant 2. For many applications, however, this quality does not significantly limit overall utility though: a simple solution for this issue is conduct a slew maneuver to center the camera on the POI prior to initiating translational maneuvering.

5.6.2 Utility with Curved Targets.

A fundamental characteristic of SNVS is its use of the PSN vector, which assumes that a predominant surface plane exists on the target. For a target with a fundamentally curved shape however, such as might be the case during RPO with the side of a rocket booster body or other cylindrically/spherically shaped target, there may be no predominant surface plane if the shape is considered as a whole. However, depending on the size of the target and relative range, it may be possible to select a locally planar region on the target surface such that a PSN vector exists and could be tracked. For example, given a cylindrical target shape, a thin slice along the longitudinal axis of the cylinder could be selected as the ROI for PSN vector calculation. Note that the depth-based predominant plane tracking method

described in Section 5.2.3.2 may not function as well for this case, so a robust image-based ROI tracking algorithm would likely be required to track the predominant surface plane.

It is also worth keeping in mind that the tracked predominant surface plane used for PSN vector computation need not be directly associated with the target POI, as long as both the predominant surface plane and POI can be maintained in view over the course of a maneuver. This means one locally flat portion of the target such as a solar panel, access hatch, patch antenna, etc. could be selected be selected as the predominant surface plane (assuming the PSN vector associated with this region properly defines the desired camera pointing direction), and a separate part of the target could be selected as the POI.

5.6.3 Implementation Without Camera Sensors.

If the predominant surface plane is tracked using depth data as discussed in Section 5.2.3.2, then the only relative navigation function requiring a camera image is the POI tracking algorithm. If the predominant surface plane of the ROI is well defined (for example if the ROI is the side of a box-type spacecraft), and the POI is taken as the geometric center of this surface, it is possible to track the POI using the same bounding box as the predominant surface plane. In this case, no camera image is required, and another type of depth sensor capable of providing dense depth maps, such as flash Light Detection and Ranging (LIDAR), could be used in lieu of a stereo camera. This is significant as it could improve the effectiveness of SNVS in low-light conditions, such as when spacecraft are in the Earth's shadow. Future work should examine other methods to track a target POI using only depth data, as this would further expand the utility of the method for conditions when sufficient lightning is not available.

5.7 Conclusion

In this chapter, a method of visual servoing for maneuver with respect to unknown targets was proposed and demonstrated in simulation. Two variants of the method were

examined, with all functions of each variant being the same except for how control commands are generated.

A defining characteristic of the proposed SNVS method is its use of the PSN vector associated with an ROI on the target. Procedures to estimate the PSN vector and track the predominant plane associated with the target ROI were developed and demonstrated.

Simulation results demonstrated the method for a of number representative final approach scenarios, including approach with a tumbling target and a extreme off-approach-axis maneuver requiring significant change in relative attitude of the chaser with respect to the target. Sensitivity analysis results were presented that showed how changes in control gains affected maneuver trajectory, time of flight, and control effort required.

SNVS has several advantages over other control methods that could be considered for RPO with unknown targets, including:

1. No pre-maneuver inspection of the target object is required; SNVS can be implemented with very little information about the target available at the start of relative maneuvering. SNVS does not estimate a geometric model of the target or other parameters such as moments of inertia before or during maneuvering, making it more computationally efficient and potentially more robust than SLAM-based methods.

2. The only computer vision function of SNVS that requires the RGB camera image pertains to tracking a single POI on the target object, for which there are many well-established algorithms available to perform this task. If a single object tracker is used to track the POI, SNVS does not utilize natural feature point identification or tracking methods, which are computationally expensive and prone to tracking failure as an image scene changes.

3. A generated goal image is not required as in the method proposed in Chapter III, which caused that method to perform poorly with targets containing nonplanar features. SNVS performs well with target objects that contain nonplanar features as long as a predominant surface plane exists in the selected ROI (i.e. the ROI is not continuously curvilinear).

4. Can be implemented using low-cost, low-SWAP hardware, using a single eye-in-hand relative navigation sensor (a stereo camera).

5. Can effectively compensate for unknown relative transitional and angular motion of the target with respect to the effector.

VI. Hardware-in-the-Loop Implementation of GNC Method 2

6.1 Overview

This chapter implements Surface Normal Visual Servoing (SNVS) experimentally using hardware-in-the-loop (HITL) in order to demonstrate the effectiveness of method using feedback from an actual stereo camera sensor. The intent of this chapter is to answer Research Question 3 as it applies to Research Question 2 (see Section 1.2). The structure of this chapter is similar to that of Chapter IV: considerations necessary for experimental implementation of SNVS are discussed is Section 6.2, results for several experimental scenarios are presented in Section 6.3, discussion topics are provided in Section 6.4, and concluding remarks are given in Section 6.5.

6.2 Method

The methods utilized for the results of this chapter are essentially identical to those of Chapter V. This section covers topics specific to experimental implementation of the method.

6.2.1 Experimental Setup.

As with the results of Chapter IV, the Air Force Institute of Technology (AFIT) Control and Autonomy Space proximity Robot (CASpR) is used to affect motion of the chaser and target for this chapter. The experimental setup utilized for this chapter is similar to that of Chapter IV (see Section 4.2.1), with the primary difference being that SNVS is used to control motion of chaser instead of the IBVS method presented previously. The CASpR GUI is modified to facilitate analysis of the method during operation, and is shown in Figure 6.1. A 3D point cloud of the image scene can be optionally displayed (example shown in the lower right of Figure 6.1), and histograms computed for Predominant Surface Normal (PSN) vector computation can be optionally plotted for each timestep (each one

overlaid on the plot from the previous timesteps, see lower middle of Figure 6.1 for example). The point cloud and histograms are useful to verify accuracy and consistency of the PSN vector estimate.

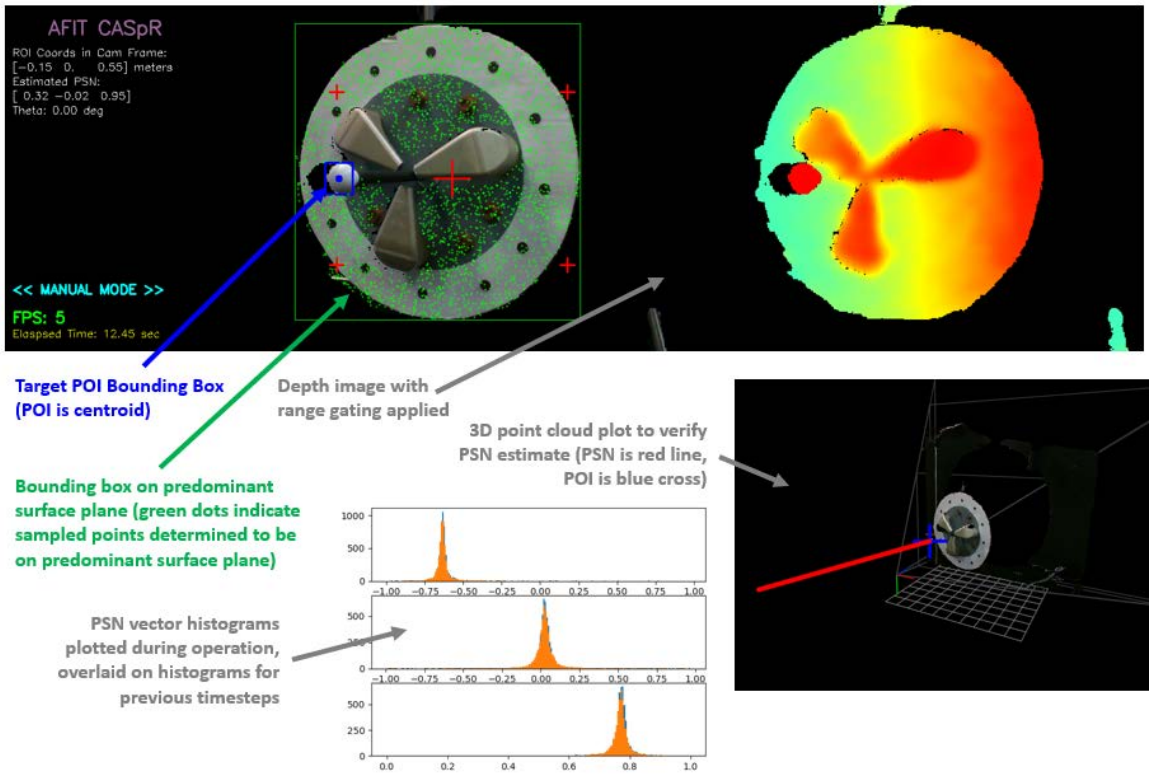

Figure 6.1: Example view of CASpR GUI utilized with SNVS method. In the upper image, camera view is shown on left, depth map on right, both with range gating applied. Image in lower right depicts example view of 3D point cloud of scene with PSN vector (red line) and POI location (blue cross) overlaid. Plot is the lower middle depicts PSN vector histograms.

The environmental lighting utilized for the results of this chapter also differs from that of Chapter IV. In Chapter IV, overhead ceiling lights were left on during operation of CASpR. For this chapter, overhead lights were turned off, and a directional lamp was used to cast light on the target. Compared to Chapter IV, overall lighting was significantly reduced and shadows were cast across the target surface. These lighting conditions are

considered to be more representative of the environmental lighting conditions present during on-orbit rendezvous and proximity operations (RPO).

The same nonplanar target examined in Chapter IV is used for the scenario results of this chapter. The POI selected for maneuvering is the tip of the boom, as shown in Figure 6.1. Notably, this POI is offset about 0.3 meters from the predominant surface plane of the the target ROI (depicted by the green bounding box and dots in Figure 6.1). This fact does not pose a problem for the method in general, as long as the range gating threshold is set large enough to not subtract the predominant surface plane from the image scene.

6.2.2 Averaging Region for POI Depth Measurement.

Each timestep the 3D position of the POI is estimated using the centroid of the target POI bounding box generated by the single object tracker. If the stereo depth calculated for the pixel associated for the centroid is invalid, then the position measurement of the POI for that timestep will be invalid. If the pixel associated with the centroid has invalid depth over several subsequent image frames, performance will be degraded as the POI Kalman filter performs no measurement updates. Therefore, a small region around the POI pixel location is used to calculate the depth associated with the POI. The size of this buffer region is adjustable (for this work a region of 5x5 pixels was observed to be sufficient). The mean of the depths to all the pixels in this region (ignoring pixels with invalid depth) is then used as the depth to the POI.

6.3 Results

This section presents results for three approach scenarios that are analogous to the three simulation scenarios presented in Chapter V. The first scenario involves an angled approach with a static nonplanar target. The second scenario is similar, except the target rotates about its vertical axis. The third scenario demonstrates a highly angled approach requiring significant relative attitude change in order to achieve the desired relative pose.

6.3.1 Scenario 6NP1: Maneuver with Respect to Static Nonplanar ROI.

The first scenario requires the chaser to conduct an angled approach with a static nonplanar target. The chaser is initially positioned $[-1.19, -1.33, -0.1]^T$ meters with respect to the target POI, angled with with the camera pointing at the target (about $-50°$ about the LVLH Z axis). The goal of the scenario is to maneuver to a relative position 0.6 meters away from the target POI with the camera optical axis pointing at the POI and orthogonal to the predominant surface plane of the target. Figure 6.2 depicts the initial and final views of the target, including both the chaser camera image and depth map (with range gating applied to remove and objects in the scene greater than 0.4 meters away beyond or in front of the target POI). Image results shown here and in the following scenarios are for SNVS Variant 1 (Variant 2 images are nearly identical and are not shown for conciseness).

Figure 6.3 depicts the trajectory of the chaser for both variants (Variant 1 on the left and Variant 2 on the right). The trajectories are similar, though as was noted in simulation, Variant 2 initially translates more directly towards the target as compared to Variant 1. Variant 1 ends the maneuver after 100 seconds at a relative position of $[0.00, -0.61, 0.00]$ meters (the desired relative position is $[0.00, -0.6, 0.00]$ meters), and Variant 2 ends the maneuver after 142 seconds at a relative position of $[0.00, -0.62, 0.00]$ meters.

Figure 6.3 also depicts the estimated PSN vector and POI location throughout the maneuver, with blueish colored vectors corresponding to vectors earlier in the maneuver, and reddish colored vectors corresponding to later in the maneuver. The error associated with these estimates is shown in Figures 6.4 and 6.5, respectively. As can be seen, both the PSN vector estimate and POI position estimate improve over the course of the maneuver.

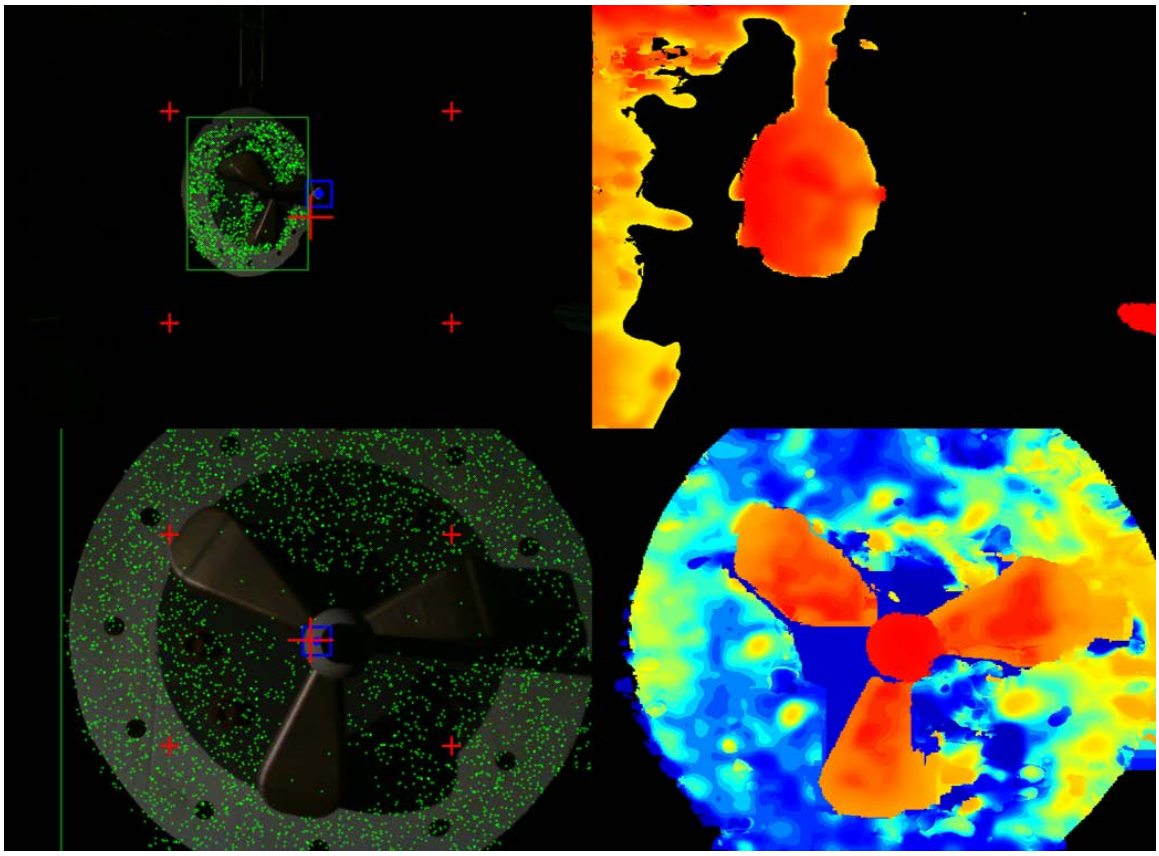

Figure 6.2: Scenario 6NP1: Initial (top) and final (bottom) view from chaser's camera and depth map (with range gating applied, images shown are for SNVS Variant 1, Variant 2 images are similar)

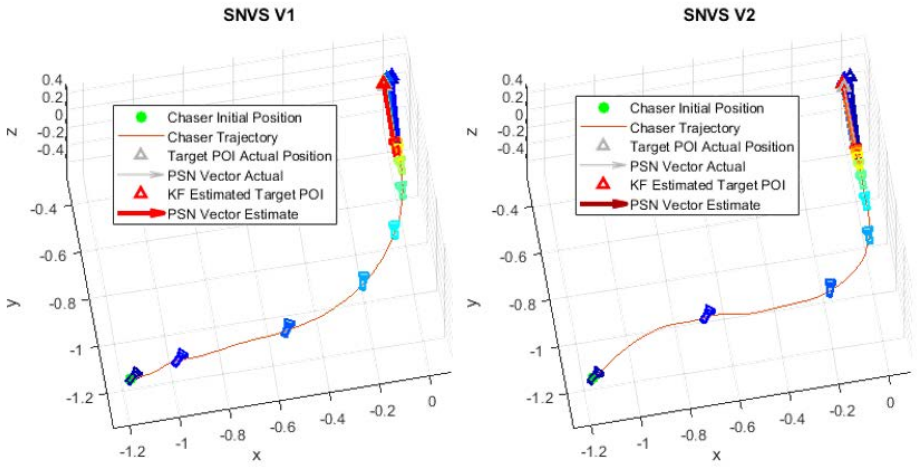

Figure 6.3: Scenario 6NP1: Chaser trajectory in LVLH frame (SNVS Variant 1 on left, Variant 2 on right).

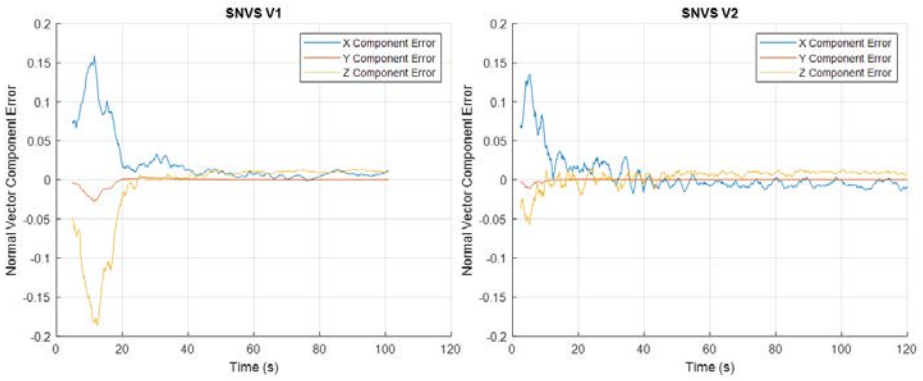

Figure 6.4: Scenario 6NP1: PSN vector component error (resolved in camera frame).

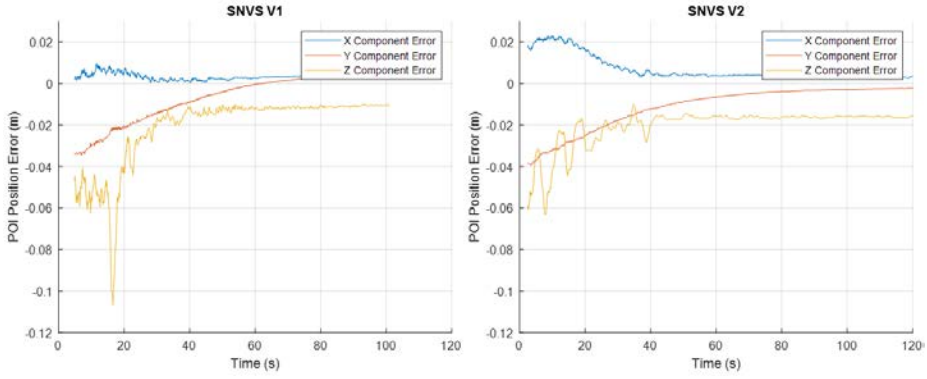

Figure 6.5: Scenario 6NP1: POI vector component error (resolved in camera frame).

6.3.2 Scenario 6NP2: Maneuver with Respect to Moving Nonplanar ROI.

This scenario involves an approach maneuver with a rotating target. The chaser is initially positioned $[0.5, -1.13, -0.10]^T$ meters with respect to the target POI. The target rotates at -1.05 degrees per second about an axis running through the predominant surface plane of the target. Since the target POI is offset from the predominant surface plane, it translates in an arc about the rotational axis during the maneuver.

Figure 6.6 shows the initial (left) and final (right) camera views for the approach maneuver, with range gating applied to remove the background and foreground beyond the target. These images show that the chaser is able to capture the desired pose, even as the lighting cast on the target object changes significantly over the course of the maneuver. Here it is worth emphasizing that lighting changes like this would certainly pose a challenge for a feature-based control method.

Figure 6.7 depicts the chaser and target trajectory, including the chaser pose and target POI/PSN vector at constant intervals over the 120 second maneuver. The actual location of the target POI and PSN vector is also shown, and the closeness of these parameters to those estimated demonstrates the accuracy of the POI and PSN vector estimation methods during practical application. The PSN vector slightly trails the the actual PSN vector in a direction opposite the target motion. This is a consequence of phase lag induced by the PSN vector Kalman vector which has gains tuned to prioritize smoothing the vector estimate over fast response to measurements.

Figure 6.8 shows the component-wise error of the PSN vector estimate relative to the actual vector over the course of the maneuver. Magnitude of component error is generally larger than observed for simulation results of Chapter V, which is as expected since the simulation utilized a perfect depth map. However, a max component error is roughly 0.06, which corresponds to about 1.3° of angular error. This is considered sufficient to affect a successful maneuver for this type of scenario.

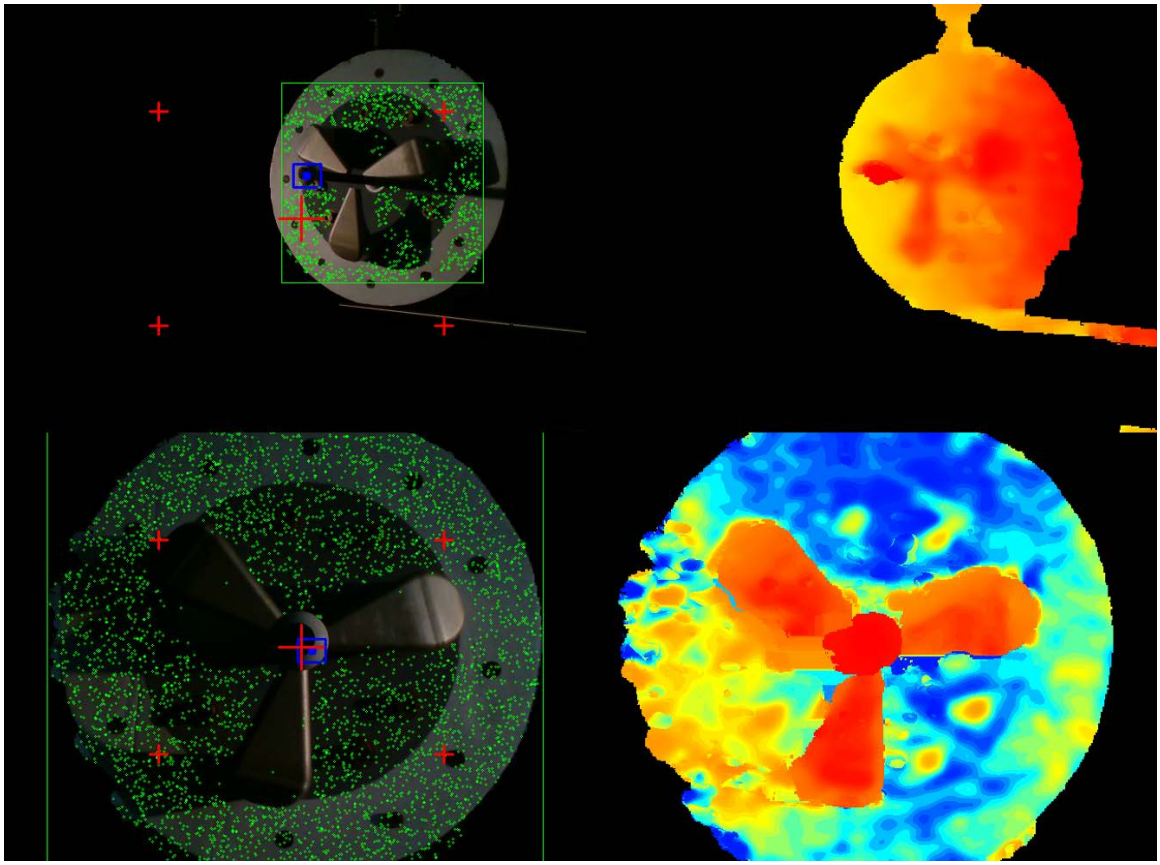

Figure 6.6: Scenario 6NP2: Initial (top) and final (bottom) view from chaser's camera and depth map (with range gating applied, images shown are for SNVS Variant 1, Variant 2 images are similar).

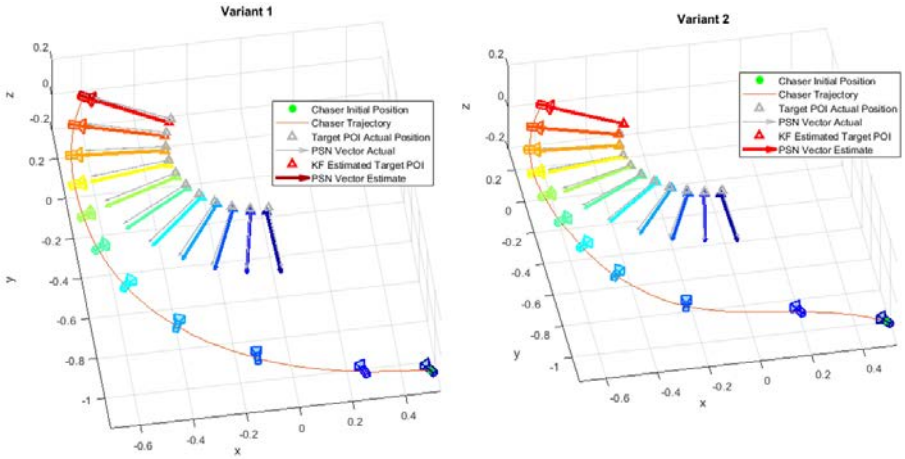

Figure 6.7: Scenario 6NP2: Chaser trajectory in LVLH frame (SNVS Variant 1 on left, Variant 2 on right).

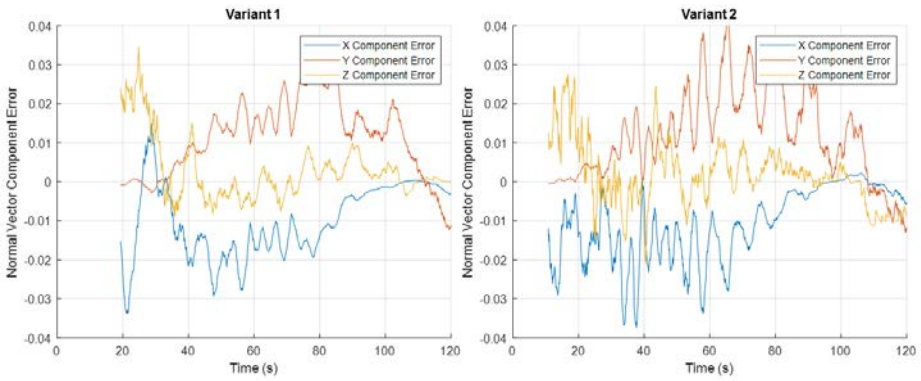

Figure 6.8: Scenario 6NP2: PSN vector component error (resolved in camera frame).

The error between the actual POI location and estimated POI location (resolved in the camera frame) is shown in Figure 6.9. The error converges to nearly sub-centimeter accuracy as the chaser reaches its desired pose, indicating the SOT successfully tracks the POI and the stereo camera provides accurate depth measurements during the entire maneuver.

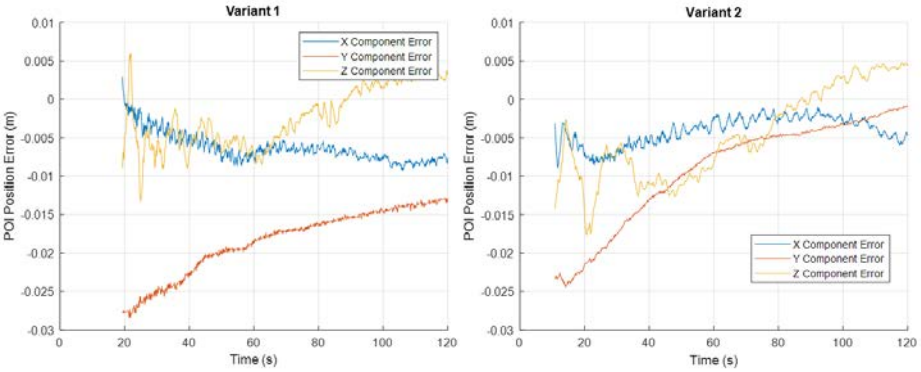

Figure 6.9: Scenario 6NP2: POI vector component error (resolved in camera frame).

6.3.3 Scenario 6NP3: Extreme Off-Approach-Axis Maneuver with Respect to Static Nonplanar ROI.

This scenario examines performance of the method for an approach maneuver requiring significant rotational motion of the chaser to achieve the desired relative attitude

with respect to the target. The target remains static throughout the maneuver. The chaser is initially positioned at $[-0.7, -0.1, -0.1]^T$ meters with respect to the target POI with the camera pointing at the POI, making the camera optical axis 82.4° off the target PSN vector about the $\hat{Y}_c$ axis at the start of maneuvering. Notably, the target object is not fully visible initially: only a portion of the target predominant surface plane can be seen in the initial camera image, shown in the upper left image of Figure 6.10. The lower images of Figure 6.10 depict the view from the chaser camera and depth map upon successfully capturing the desired relative pose.

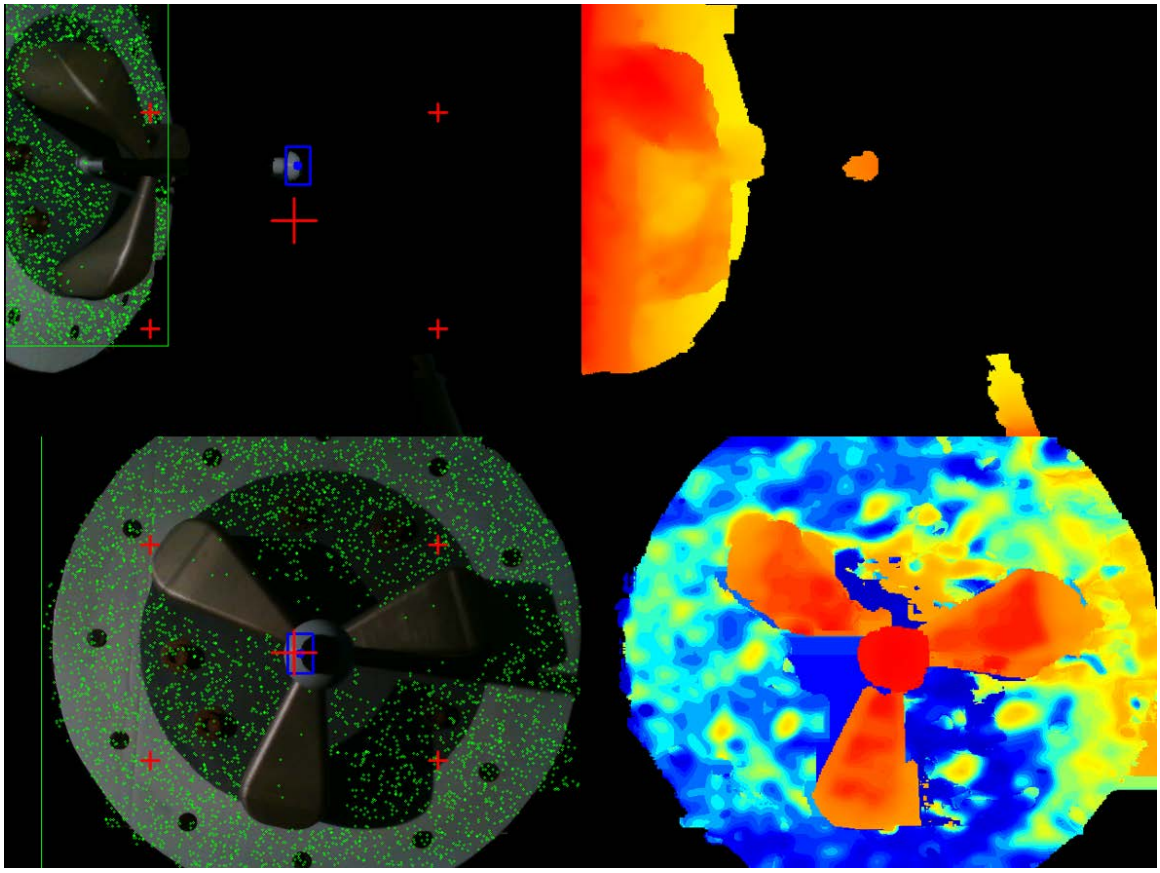

Figure 6.10: Scenario 6NP3: Initial (top) and final (bottom) view from chaser's camera and depth map (with range gating applied, images shown are for SNVS Variant 1, Variant 2 images are similar).

Figure 6.11 shows the trajectory of the chaser with respect to the static target. As with previous trajectory plots, time histories of the chaser pose, estimated target POI location, and estimated PSN vector are depicted at a constant interval. The plot shows that the estimated POI location and PSN vector for the static target remain consistent over the entire maneuver. Importantly, the chaser takes an arcing trajectory to arrive at the desired relative pose (as opposed to translating directly towards the desired position). This is intentional: the control gains were tuned to prioritize convergence to the desired relative attitude vice position. Doing so prevents the chaser from initially translating towards the target, which would cause less of the target to be visible and therefore make accurate estimation of the PSN vector more difficult.

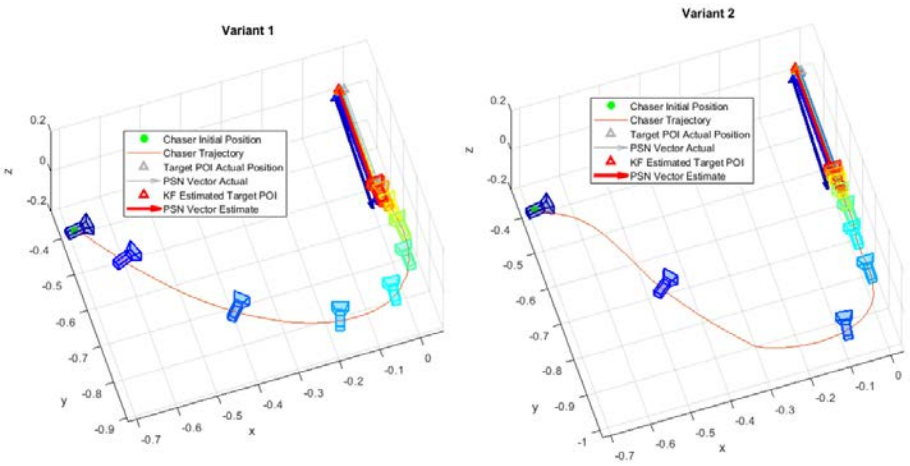

Figure 6.11: Scenario 6NP3: Chaser trajectory in LVLH frame (SNVS Variant 1 on left, Variant 2 on right).

Error for the PSN vector components is shown in Figure 6.12. The plot shows that component error for this scenario is generally better than that of Scenari 6NP2 (Figure 6.8), which is expected since the target is static for this scenario.

Figure 6.13 displays the initial and final set of histograms used to compute the PSN vector estimate. These histograms useful to assess the robustness of the PSN estimate.

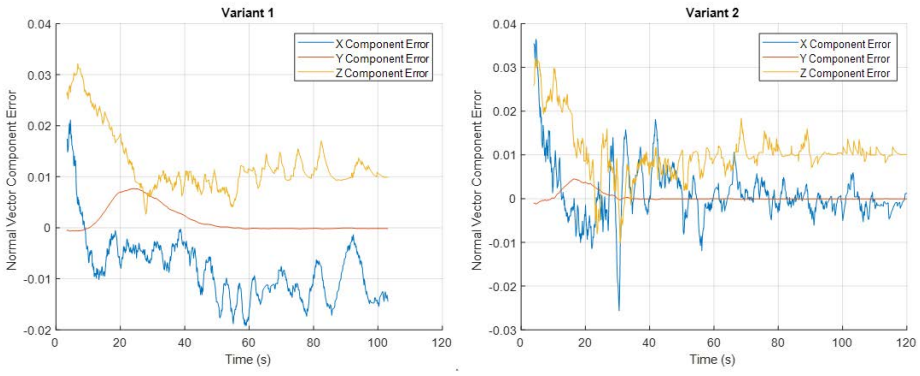

Figure 6.12: Scenario 6NP3: PSN vector component error (resolved in camera frame).

Histograms with unimodal sharp peaks such as those shown in Figure 6.13 indicate a well-defined PSN vector that will be easily re-identified in subsequent frames.

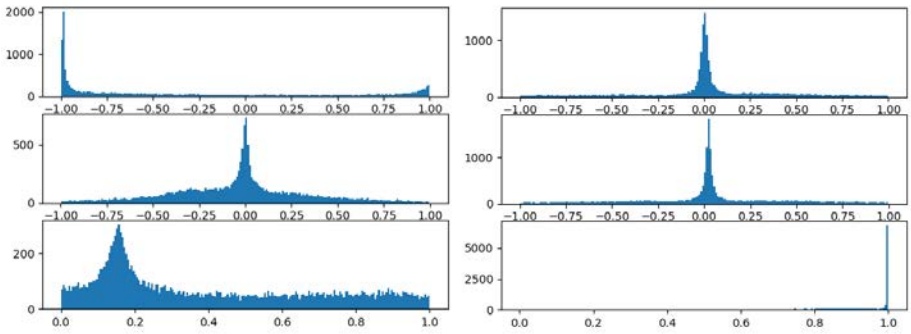

Figure 6.13: 6NP3: PSN vector component histograms (inital on left, final on right).

Finally, Figure 6.14 shows the position error of the POI estimate over the course of the maneuver, resolved in the camera frame. The error is initially worse than that of the previous scenario, likely due to a poor estimate of depth by the stereo camera and the fact the center protrusion of the target object is being viewed from the side. However, position error converges to sub-centimeter accuracy by completion of the maneuver. The error in the X direction converges to roughly 0.01 meters due to the POI being incorrectly slightly tracked by the single object tracker.

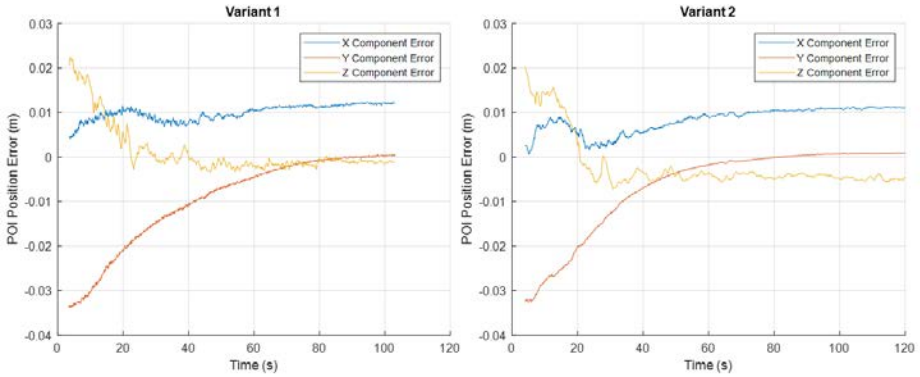

Figure 6.14: Scenario 6NP3: POI vector component error (resolved in camera frame).

6.4 Discussion

This section examines several discussion topics pertaining to SNVS based on the results of this and previous chapters.

6.4.1 Examination of ROIs with Two Equally Sized Surface Planes.

Implementation of SNVS is predicated on the ability to estimate a PSN vector associated with a predominant surface plane of a region of interest (ROI) on the target surface. If the ROI selected by an operator prior to maneuvering does not contain a single predominant surface plane, the histograms computed for the PSN vector estimate may not be unimodal, which can lead to issues determining the desired navigational vector for maneuvering. This section examines two such cases where the ROI selected by the operator contains two equally sized planar regions.

For the first case, a box-shaped target is viewed with the optical axis of the camera pointing directly at an edge, such that two sides of the box form 45° angles from the optical axis. An ROI is selected such that the ROI initially contains equal area of the left and right visible sides of the box. Figure 6.16 depicts results from the first timestep for this case. The PSN vector X-component histogram is bimodal with roughly equally sized peaks, corresponding to each side of the box within the ROI. The peak associated with the left side

of the box is slightly larger however, so the initial estimate of the PSN vector corresponds to that side of the box.

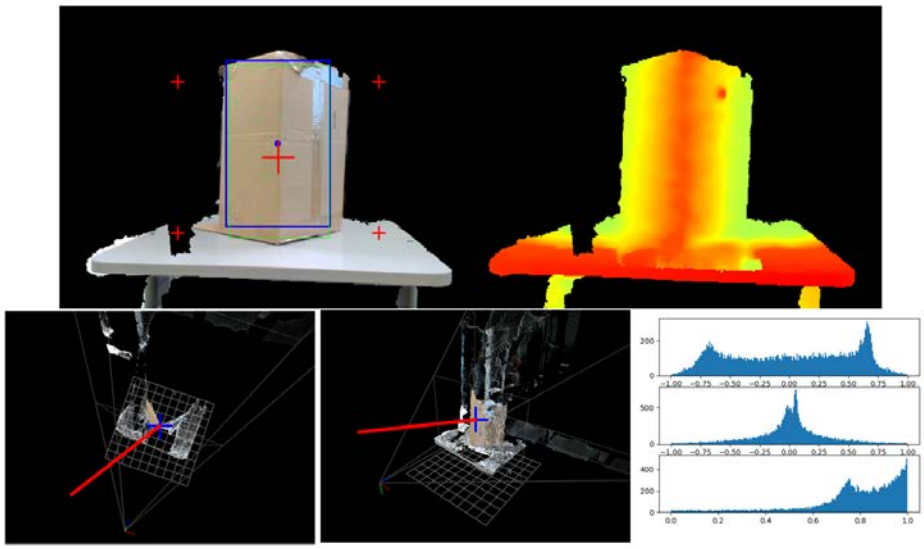

Figure 6.15: Initial timestep PSN vector results for box edge ROI. Top left: color camera image (blue bounding box tracks POI, green bounding box tracks predominant surface plane); top right: aligned depth map of scene; bottom left/middle: two views of point cloud of scene with PSN vector estimate (red vector) and POI (blue cross) overlaid; bottom right: histograms generated to compute PSN vector .

Steady-state PSN vector results for the same case are shown in Figure 6.16. Recall that the predominant surface plane tracking method uses the PSN vector histograms to build a bounding box around the group of sampled points whose normal vectors are nearly aligned with the estimated PSN vector. Since the PSN vector is initially determined to be the normal vector associated with the left side of the box, the predominant plane tracking method quickly moves the bounding box to surround only the left side of the box. In doing so, the PSN vector histograms quickly collapse to be unimodal, as now there is only one predominant surface plane for the tracked ROI.

The second case examines an ROI initially containing two equally-sized parallel planes. Figure 6.16 displays the PSN vector results for the initial timestep. Even though the

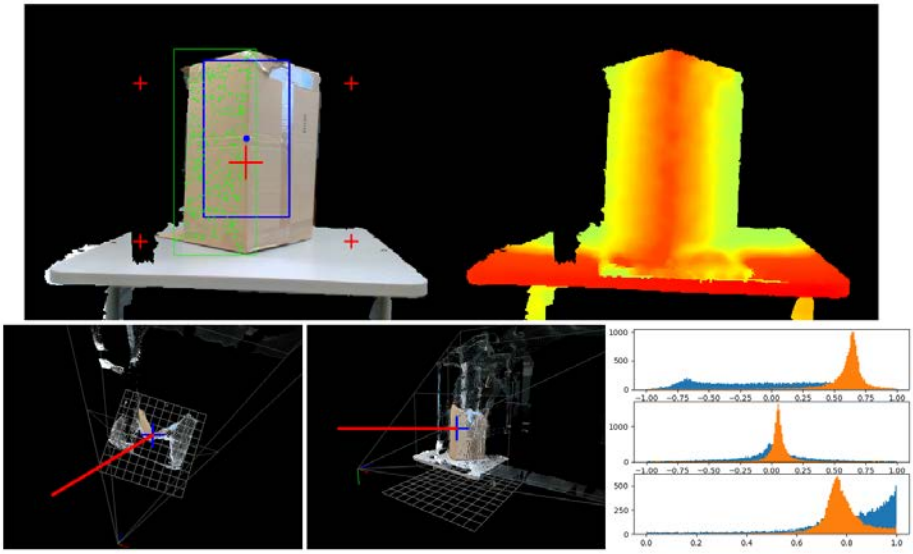

Figure 6.16: Steady state PSN vector results for box edge ROI. Top left: color camera image (blue bounding box tracks POI, green bounding box tracks predominant surface plane); top right: aligned depth map of scene; bottom left/middle: two views of point cloud of scene with PSN vector estimate (red vector) and POI (blue cross) overlaid; bottom right: histograms generated to compute PSN vector (shown in orange, overlaid on top of initial timestep histograms shown in blue).

two planes individually have the same normal vector the estimated PSN vector is initially not aligned with the correct normal vector. Once again the X-component histogram is bimodal, making consistent estimation of a single PSN unreliable.

As with the first case however, in the steady state the predominant surface plane tracking method moves the bounding box to encompass just one of the planes, and the PSN histograms collapse to be unimodal. Steady state PSN vector results are shown in Figure 6.18 and indicate a valid and stable PSN vector estimate for the right plane only.

The results of this section motivate the importance of allowing the PSN vector estimate to converge prior to initiating maneuvering. Using a combination of a 3D point cloud view of the scene and computed PSN vector histograms, operators should assess the correctness and robustness of PSN vector estimates prior to and during maneuvering.

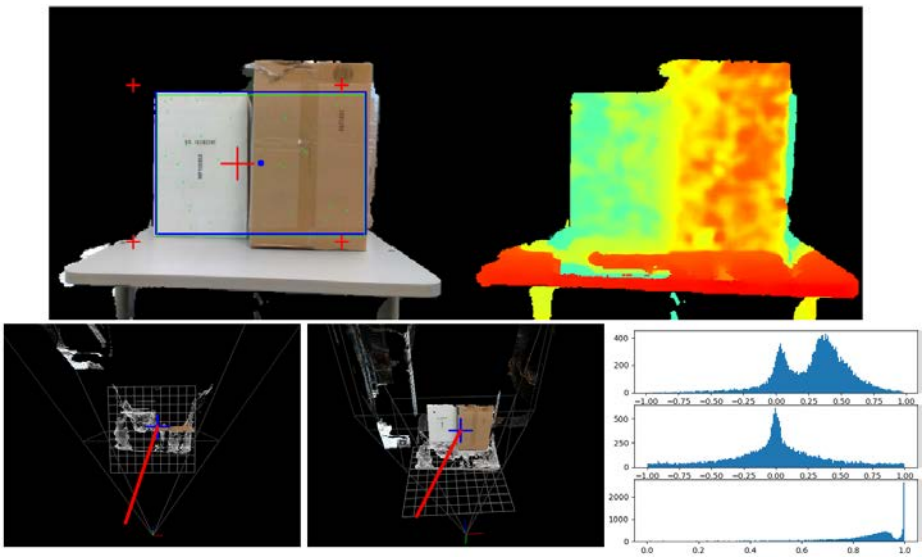

Figure 6.17: Initial timestep PSN vector results for two parallel planes ROI. Top left: color camera image (blue bounding box tracks POI, green bounding box tracks predominant surface plane); top right: aligned depth map of scene; bottom left/middle: two views of point cloud of scene with PSN vector estimate (red vector) and POI (blue cross) overlaid; bottom right: histograms generated to compute PSN vector.

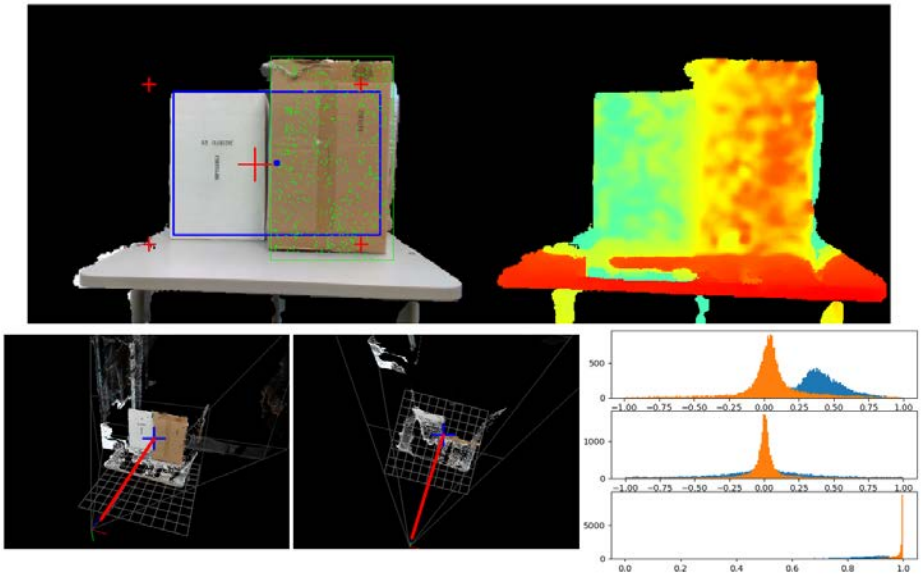

Figure 6.18: Steady state PSN vector results for two parallel planes ROI. Top left: color camera image (blue bounding box tracks POI, green bounding box tracks predominant surface plane); top right: aligned depth map of scene; bottom left/middle: two views of point cloud of scene with PSN vector estimate (red vector) and POI (blue cross) overlaid; bottom right: histograms generated to compute PSN vector (shown in orange, overlaid on top of initial timestep histograms shown in blue).

6.4.2 Comparison of SNVS Variants.

Both SNVS variants have been successfully demonstrated in simulation and via experimentation with HITL. Both were shown to be capable of achieving the desired relative pose for a variety of maneuvers with similar precision. Several differences were noted however, which may make each variant more or less preferable depending on the particular use-case.

Compared to Variant 2, SNVS Variant 1 was observed to have the following qualities:

- Producing more desirable trajectories with SNVS Variant 1 is more straightforward. Here desirable trajectories are considered to be those where the chaser initially translates towards an approach path orthogonal to the target surface (for example see Figure 6.11). This is beneficial in order to keep more of the predominant surface plane in the Field of View (FoV).

- SNVS Variant 1 was consistently shown to utilize less control effort in simulation for all test cases examined, though it is possible further analysis may demonstrate otherwise.

- If the target POI is initially non-centered in the FoV, SNVS Variant 1 yields trajectories that deviate less from the trajectory generated if the POI were centered, as discussed in Section 5.6.1.

- If the attitude control gain β is not properly tuned, maneuvers using SNVS Variant 1 may be more likely to have the POI to leave the camera FoV.

6.4.3 Comparison with GNC Method 1 (IBVS for Unknown Targets).

This section provides a general comparison of SNVS and the IBVS method presented in Chapters III and IV of this work. Compared to the IBVS method, SNVS has the following relative advantages:

- SNVS is capable of being implemented with bland target ROIs (i.e. target ROIs for which few target features could be identified via FEM), as it does not utilize target features for maneuvering.
- SNVS was observed to perform better under low light conditions (the SNVS results of Chapter VI were conducted with reduced ambient lighting as compared to the IBVS results of Chapter IV).
- SNVS is better at achieving the desired relative pose for nonplanar target ROIs, as long as a predominant surface plane exists within the ROI.
- SNVS allows explicit definition of desired relative pose, as opposed to IBVS, were the desired relative pose is implicitly defined by the goal image.
- SNVS has fewer tuning/adjustable parameters, especially with regards to computer vision functions.
- SNVS Variant 1 provides the simplest means to shape the approach trajectory such that the chaser achieves an approach direction orthogonal to the target surface as early in the maneuver as possible.
- Depending on the nature of the target, SNVS can be implemented using depth information only (see Section 5.6.3).

Compared to the IBVS method, SNVS has the following relative disadvantages:

- SNVS does not control relative rotation about the optical axis of the camera.
- SNVS requires higher quality depth information for accurate computation of the PSN vector.
- SNVS does not work well with continuously curvilinear ROI's. While IBVS performs poorly when large nonplanar protrusions exist, it will likely perform better

than SNVS for an ROI with a slight continuous curve, as might be the case if the target was the body of a booster, for example.

6.4.4 System Parameters.

This section provides a summary of all the adjustable parameters associated with both variants of SNVS, including the values used for the results presented herein, and relevant notes for each parameter. The intent of this section is to provide designers a starting point for future implementations of this method. Tables 6.1, 6.2, and 6.3 detail the parameters associated with the image processing, SNVS Variant 1, and SNVS Variant 2, respectively.

Table 6.1: Image Processing Parameters

Tuning Parameter	Value/Setting for this Work	Description/Notes
Stereo Camera Settings		
RGB camera resolution	640x480	Higher resolution increases computational load but improves POI tracking
Depth map resolution	640x480	Higher resolution increases computational load but may improve depth estimation for small nonplanar target components
Other settings (focal length, shutter speed, ISO, etc.)	Intel RealSense D345 defaults, autoexposure off for RGB and depth	Unique to equipment/application, not addressed in this work
Depth Filtering		
Range gate threshold	0.4 meters	Distance in front of or behind POI beyond which depth and RGB image pixels are ignored (set to 0)
Decimation filter magnitude	2x	More decimation reduces scene complexity and improves computational efficiency
Spatial filter magnitude	2	Number of spatial filter iterations
Spatial filter smoothing factor	0.25	Alpha factor for exponential moving average filter
Spatial filter edge size threshold	20	Threshold used to determine what is an edge
Temporal filter smoothing factor	0.4	Alpha factor for exponential moving average filter
Temporal filter edge size threshold	20	Threshold used to determine what is an edge
POI Tracking		
Tracker Type	CSRT	Single object tracker algorithm (see Section 2.4.3.2)
POI Depth Averaging Buffer Size	5x5 pixels	Size of region around POI used to determine depth to POI. Depth to pixels in region are averaged, pixels with invalid depth are ignored. (useful if pixel associated with POI has invalid depth)
PSN Vector Computation Settings		
Number of point sets to sample	2E4	Number of random point sets sampled within ROI for PSN vector computation (see section 5.2.2.2)
Number of histogram bins	200	Number of histogram bins used to compute PSN vector components (see Section 5.2.2.1)
Histogram Plane Points Threshold	+/-0.05	Threshold ranges specified by adding this value to each PSN vector component. Sampled points with all normal vector components within corresponding component range are presumed to be on predominant surface plane. Used to detemine predominant surface plane bounding box (see Section 5.2.3.2)
Predominant surface plane bounding box buffer	3 pixels	Buffer added to predominant surface plane bounding box to keep it from shrinking (see Section 5.2.3.2).

Table 6.2: SNVS Variant 1 Parameters

Tuning Parameter	Value/Setting for this Work	Description/Notes
Kalman Filter Settings		
Process covariance (POI KF)	Position states: 5E-3 Velocity states: 0.5	Set high to prioritze feature location measurements
Process covariance (PSN vector KF)	Position state: 5E-3 Velocity state: 0.5	Set high to prioritze depth measurements
Measurement covariance for POI	eye(3)	Consider setting this very low (see Section 4.2.3.4)
Measurement covariance for PSN vector	eye(3)	High values smooth feature trajectory, but introduce lag
Initial covariance	Position states: 1E4 Velocity states: 1E3	Affects rate of convergence of filter, not observed to have a major effect on performance
Translational Controller (LQR + integral action)		
Q_{lqr}	diag([1.5E6, 1.8E5, 3.6E3, 3E7, 2.4E6, 1.8E6])	LQR performance weight
R_{lqr}	eye(3)	LQR control weight
K_i	diag([20, 1, 1E-5])	Integral action gain matrix
Attitude Controll (Quaternion Feedback)		
τ_{max}	10	See eq. (5.25)
ζ	17.50	Damping ratio, see eq. (5.24)

Table 6.3: SNVS Variant 2 Parameters

Tuning Parameter	Value/Setting for this Work	Description/Notes
Kalman Filter Settings		
Process covariance (POI KF)	Position states: 5E-3 Velocity states: 0.5	Set high to prioritze feature location measurements
Process covariance (PSN vector KF)	Position state: 5E-3 Velocity state: 0.5	Set high to prioritze depth measurements
Measurement covariance for POI	eye(3)	Consider setting this very low (see Section 4.2.3.4)
Measurement covariance for PSN vector	eye(3)	High values smooth feature trajectory, but introduce lag
Initial covariance	Position states: 1E4 Velocity states: 1E3	Affects rate of convergence of filter, not observed to have a major effect on performance
Translational Controller (LQR + integral action)		
Q_{lqr}	diag([1E5, 5E4, 5E4, 4E6, 4E6, 4E6])	LQR performance weight
R_{lqr}	eye(3)	LQR control weight
K_i	diag([0.2, 0.2, 1E-5])	Integral action gain matrix
Attitude Controller (PID)		
Proportional gain	0.2	PID control gains heuristically tuned to provide suitable tracking of POI.
Iterative gain	1.5	
Derivative gain	0.001	

6.5 Conclusion

This chapter demonstrated both variants of the SNVS method proposed in Chapter V using HITL. AFIT CASpR was used to affect motion of the chaser and target, and a D435 stereo camera was used by the chaser to preform relative navigation. Experimentation showed that the method is capable of being implemented in real-time using a desktop computer. Additionally, results indicated that the method can correctly and reliably estimate the PSN vector using real video depth maps of nonplanar targets.

Three approach scenarios were presented which demonstrated complex maneuvers with with a nonplanar target object. Notably, the results show that the PSN vector estimation and target POI tracking methods work well under poor lighting conditions, thus indicating the potential of the method for on-orbit RPO. Overall, the HITL results presented in this chapter demonstrate the effectiveness of the SNVS method. Results confirm the benefits of the method discussed in Chapter V, and specifically indicate that the method is much more capable than the IBVS method presented in Chapters III and IV for maneuvering with nonplanar targets and under poor lighting conditions.

VII. Conclusion and Recommendations

This work addressed how the final approach phase of spacecraft rendezvous and proximity operations (RPO) with unknown targets can be conducted using low-cost, low size, weight, and power (SWAP) hardware. Two navigation and control methods for commanding autonomous relative maneuvers during this type of RPO were developed and demonstrated in simulation and using hardware-in-the-loop (HITL). Both methods rely on a stereo camera sensor for relative navigation and enable rapid execution of complex relative maneuvers, requiring only a single training image of a target region of interest (ROI) in order to initiate maneuvering. Unlike other control methods that could be considered for this problem, neither method requires estimation of a geometric model of the target or its moments of inertia.

While the intended use-case for each of the proposed methods is similar, it was observed that each method has advantages under different circumstances (see Section 6.4.3), thus using both of them together as part of a larger guidance, navigation, and control (GNC) architecture is recommended. A basic taxonomy for how both methods can be combined is shown in Figure 1.3.

This chapter concludes the work by summarizing the contributions generated in pursuit of answering each of research questions outlined in Chapter 1, as well as making recommendations regarding future research.

7.1 Contributions

The primary contributions of this work include two methods to autonomously control the final approach phase of RPO with unknown targets. Development of each of these methods corresponds to Research Questions 1 and 2. Research Question 3 addresses further development necessary to implement each method using HITL. This section

outlines the specific contributions associated with each research question. As well as the contributions listed below, an additional conference paper outlining a general GNC approach for spacecraft RPO was published in conjunction with this work [154].

7.1.1 Research Question 1 Contributions.

Contributions related to the first research question include:

- Development of a navigation and control framework to command autonomous RPO maneuvers with unknown targets. The method implements an Image-Based Visual Servoing (IBVS) control law, augmented with procedures necessary to utilize IBVS with unknown targets.

- A method to estimate the six degree of freedom (6DOF) relative velocity between a chaser spacecraft and its target. The method utilizes tracked feature points on the target surface and a bank of Kalman filters to estimate the velocity of each feature in the image domain. The 6DOF relative velocity is estimated via a least squares optimization using the interaction matrix relationship between the image space and Cartesian space.

- Two methods to generate goal images necessary for implementation of IBVS with unknown targets. The first method utilizes a zoomed-in image of the target to approximate the location of target feature points in a goal pose. The second method warps the current location of target feature points in order to estimate the location of the feature points when the chaser is in the desired relative pose.

- Development of a simulation for use with development and analysis of visual navigation and control methods pertaining to spacecraft RPO. The simulation models 6DOF relative motion between two spacecraft using the Nonlinear Equations of Relative Motion (NERM) and synthesizes a camera view and depth map from the perspective of the chaser spacecraft.

- A method for modeling the four main sources of error associated with feature-based image processing and stereo depth estimation.
- Demonstration of the proposed method in simulation with static and moving nonplanar targets, including analysis of the effectiveness of the goal image generation methods for planar and nonplanar target surfaces, and robustness to measurement noise associated with target feature point tracking.
- A conference paper [155] was published pertaining to this research question, as well as a journal paper accepted for publication documenting the results of this research question [156].

7.1.2 Research Question 2 Contributions.

Contributions related to the second research question include:

- Development of a novel type of visual servoing intended to control robotic maneuvers with unknown targets, termed Surface Normal Visual Servoing (SNVS). Two closely related variants were proposed, with the primary difference pertaining to how control error is defined. SNVS has several notable advantages over GNC Method 1, including increased effectiveness with nonplanar targets, better performance under low-light conditions, and lack of the requirement to track feature points on the target surface.
- A computationally efficient method to compute a novel navigational vector, termed the Predominant Surface Normal (PSN) vector, using dense depth measurements from a stereo camera. The PSN vector describes a vector orthogonal to the predominant surface plane of a target ROI and is used by SNVS to defined the desired pointing direction of the chaser's camera upon completion of maneuvering.
- A computationally efficient computer vision method to track predominant surface plane associated with a target ROI.

- Implementation of a quaternion feedback attitude control law that simultaneously regulates the camera to the desired pointing direction while keeping the target POI in the camera Field of View (FoV). Development of this controller includes formulation an expression for the error quaternion, and a method to estimate the required relative angular velocity term of the control law that does not require additional sensors.

- Demonstration of the method in 6DOF simulation with static and moving nonplanar targets, including analysis of the method for several representative approach scenarios, and sensitivity analysis of how changing control gains and initial conditions affect various aspects of maneuvering.

- A conference paper [153] was published pertaining to this research question, as well as a journal article submitted for publication documenting the results of this research question [157].

7.1.3 Research Question 3 Contributions.

Research Question 3 pertains to extending the methods developed for the previous research questions to enable implementation with HITL. Accordingly, contributions are listed as they apply to Research Question 1 and 2. In addition to the contributions outlined below, another contribution pertains to the development of the Air Force Institute of Technology (AFIT) Control and Autonomy Space proximity Robot (CASpR). CASpR was designed and constructed specifically to support this research, and it is expected to have continuing utility for follow-on research and future research regarding spacecraft RPO. CASpR realizes 4DOF relative motion of two independent spacecraft bodies using actual hardware sensors and processors.

As applied to Research Question 1, contributions include:

- A computer vision feature tracking pipeline used to identify and track image features within the target ROI.

- A method for robustly rejecting target feature measurement outliers by monitoring the IBVS servo error and rejecting features whose servo error does not follow the overall trend of servo error for all features.

- A mode of operation whereupon the chaser attempts to rigidly maintain its current 6DOF relative pose with respect to the target if certain criteria are met to activate the mode. This mode represents an important capability of the overall method as it enables safe operating conditions to be autonomously maintained if a maneuver degrades to a point that an abort of the current operation is required.

As applied to Research Question 2, contributions include:

- Demonstration of both SNVS variants using CASpR. Both variants were implemented in real time using actual stereo camera feedback and a desktop computer. Results were presented demonstrating effectiveness under challenging operation conditions including low ambient light and maneuvers with respect to a moving target ROI containing significant nonplanar protrusions.

- Validation of PSN vector estimation procedure and demonstration of the predominant plane tracking method under the realistic conditions described above. Analysis of the effectiveness and accuracy of these methods using actual stereo depth information.

- The aforementioned journal article submission [157] also conveys contributions pertaining to the research question.

7.2 Recommendations for Future Work

With the successful demonstration of the proposed methods in simulation and using a commercial stereo camera sensor under laboratory conditions, future work should focus on implementing practical versions of these methods using flight-capable hardware and conditions as realistic to the on-orbit environment as possible. The limitations associated

with using CASpR and a commercial sensor meant that the methods could only be experimentally demonstrated at comparatively short ranges. Future work should examine use of stereo cameras with longer baselines operating at longer ranges. An analysis of how the accuracy of stereo measurements changes with range and how this would affect the proposed methods is warranted. Furthermore, with space being a resource-constrained environment for maneuvering spacecraft, optimization techniques should be applied to both methods in order to understand and improve fuel use for potential RPO applications. Other areas of future work specific to each method are outlined below.

7.2.1 GNC Method 1 Future Work.

Recommendations for future work regarding GNC Method 1 (IBVS for unknown targets) are discussed in detail in Section 4.4, and are summarized here:

- Examine improving the Feature Extraction and Matching (FEM) subroutine of the image-processing pipeline by developing a method to dynamically update target feature descriptor vectors during maneuvering.

- Improve the ability to maneuver over long distances and time-frames by developing a method to identify new target features during maneuvering (i.e. after the initialization procedure), which would require a procedure to estimate the goal position for newly identified features. Doing so would prevent degradation of maneuvering as more target features would be tracked later into an approach maneuver.

- Develop an improved method to estimate the goal locations of features for nonplanar target ROIs.

- Consider implementation of the method with a zooming stereo camera. This would require modifying the control law to be invariant to focal length, and would allow the method to be implemented during longer range maneuvers since the camera focal length could be changed to keep the target ROI a desired size in the image plane.

7.2.2 GNC Method 2 Future Work.

Recommendations for future work regarding GNC Method 2 (SNVS) include:

- Augment the method in order to enable use without a visible spectrum camera image (i.e. using dense depth maps only, as could be provided by flash LIDAR sensors). In order to achieve this, a method to track a POI using only depth information would need to be developed. This improvement would increase utility of the method during circumstances where little to no ambient light is available.

- Examine methods to increase utility with curvilinear target ROIs. As proposed, SNVS relies on the target ROI containing a predominant surface plane in order to estimate the PSN vector that is used for maneuvering. If some assumptions are made about the curvature of a selected ROI on the target, it may be possible to estimate a navigational vector analogous to the PSN vector for this use-case. Doing so would make the method more broadly applicable for a variety of potential target shapes and types.

Appendix A: Visual Navigation and Servoing Simulation (VNSS)

This appendix details the 6DOF simulation program written to support this research, called the Visual Navigation and Servoing Simulation (VNSS).

A.1 Simulation Overview

A simulation was developed in MATLAB Simulink in order to test the architecture in a simulated setting. The simulation propagates relative 6DOF motion of the chaser and target with respect to the LVLH frame. In addition, a stereo camera on the chaser spacecraft is modeled, including rendering of grayscale camera images and depth maps at user selectable rates.

A.1.1 Dynamics/Plant Model.

The Nonlinear Equations of Relative Motion (NERM) (described in Section 2.3.1) are used to model the translational dynamics of each spacecraft. As both the chaser and target are free to translate with respect to a vritual Local-Vertical-Local-Horizontal (LVLH) origin, the NERM's are propagated independently for each spacecraft. The Earth-Centered Inertial (ECI) orbit of the LVLH frame is defined by a set of classical orbital elements prior to simulation. For the results presented in this work, the orbit of the LVLH frame is circular (eccentricity of 0) in low Earth orbit (altitude of 450 km).

The rotational dynamics of each spacecraft are propagated independently using Euler's rotational equations of motion (see Section 2.3.2). These equations propagate the angular velocity of a spacecraft with respect to the ECI frame ('N' frame), then the angular velocity of the ith spacecraft body with respect to the LVLH frame ('L' frame) is determined by:

$$\boldsymbol{\omega}_{iL} = \boldsymbol{\omega}_{iN} - \boldsymbol{\omega}_{LN} \tag{A.1}$$

To simulate the target body, a 3D model (a stereolithography file) is imported as a set of vertices and faces. The vertices of this model are propagated in simulation using the method described in Section 2.3.3.

A.1.2 Camera Modeling.

The camera sensor is modeled as a pinhole camera as presented in Section 2.4.1. Like other parameters of the spacecraft, the camera intrinsic parameters are user-adjustable to facilitate investigation of realistic systems. Unless otherwise noted, the results presented in this chapter assume the chaser's camera is calibrated (i.e. the actual camera parameters are precisely known), with parameter values as shown in Table A.1. As shown in the table, the frequency of image and depth measurements is set to 10 frames per second (FPS). This value represents the rate at which image/depth measurements are available to the GNC system after image processing is accomplished. As such, it is dependent on a number of factors beyond the camera itself, such as: the speed of the computer using for image processing, which feature extraction and matching algorithms are used, and the number of features tracked. A value of 10 FPS is expected to be be achievable by currently available flight hardware and image processing algorithms.

Table A.1: Camera Intrinsic Parameters Used in Simulation

Camera Parameter	**Value**	**Units**
Resolution	1280×960	pixels×pixels
Focal Length	10	mm
Pixel Size	10×10	$\mu m \times \mu m$
Principal Point	$[640, 480]$	pixels
Skew Coefficient	0	pixels/μm
Stereo Baseline	200	mm
Measurement Update Rate	10	frames/sec

A basic rasterization pipeline [158] is implemented in order to render a grayscale image and a depth map from the camera sensor. Figure A.1 shows an example grayscale

image from the camera (highlights in red manually added for annotation). Figure A.2 shows an example depth map taken at the same instant.

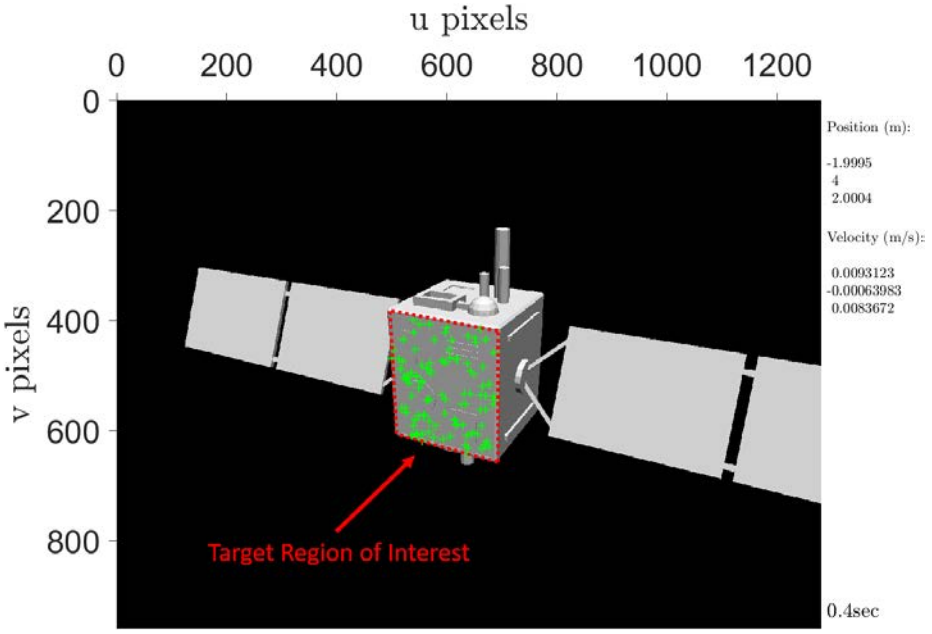

Figure A.1: Example chaser camera view displaying synthetic image features cast onto the target (indicated by green markers)

A.1.2.1 Target Feature Point Generation.

Implementing the full feature-based image processing pipeline used by this architecture would require photo-realistic rendering of image scenes for the feature extraction and matching algorithms to function. Even with high fidelity synthetic images though, there is no guarantee that these algorithms would produce results commensurate with analogous real-life scenarios. Therefore, in lieu of implementing these algorithms, image features are synthetically generated in the simulation and are propagated using the relative motion dynamics/pinhole camera model previously described. In order to increase realism and assess the robustness of the proposed controller, the most common sources of error pertaining to these computer vision algorithm are modeled.

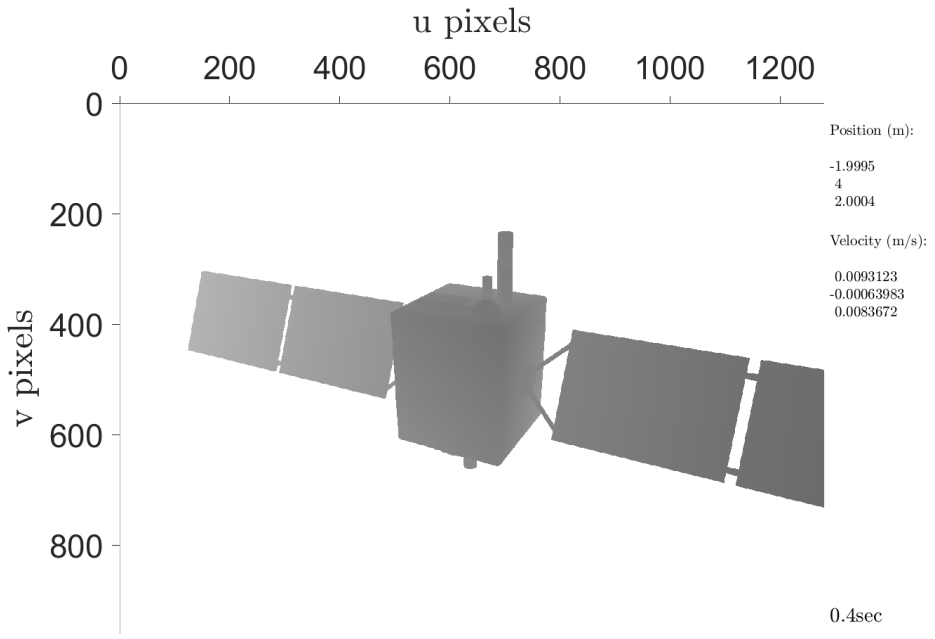

Figure A.2: Example depth map generated by chaser camera sensor (no measurement noise applied)

Feature points are synthetically generated by randomly selecting points on surface of the target body within a region of interest (ROI) specified by the user. Figure A.1 depicts the ROI and synthetic image features cast on the target (shown as green markers). Image measurements of these feature points are generated for each image frame with error modeling applied as described in the next subsection.

A.1.2.2 Modeling Computer Vision Sources of Error.

Since feature extraction and matching algorithms are not implemented in simulation, the error that would occur with real image processing algorithms is approximated by several stochastic mechanisms. Accurately modeling these computer vision error sources is important in order to make an assessment of the robustness of the architecture under realistic or worse-than-realistic conditions.

Though the process of feature identification, matching, and outlier rejection is robust when implemented with RANSAC, it does not perfectly locate matched features in every

image frame. Some sources of error associated with image feature matching are discussed in [159]. For the purposes of this study, image processing error is modeled by four mechanisms:

1. Image feature pixel location (measurement) error
2. Image feature nonidentification
3. Image feature misidentification (outliers)
4. Image feature depth measurement error

Image feature pixel error refers to the fact that derived feature pixel locations are not perfectly consistent from frame to frame. This is a result of the feature detector locating a given feature at a slightly different pixel location in each frame, even if the corresponding 3D point has not moved in relation to the camera. Feature location error is commonly modeled as zero-mean additive white Gaussian noise applied to the pixel location of a feature [132, 160].

Image feature nonidentification refers to the failure of the feature detection algorithm to identify and match a previously known feature in a given frame. This can occur for many reasons, such as a feature being occluded or outside of the field of view of the camera. If scene conditions (such as lighting, contrast, scale, rotation, etc.) change sufficiently, a feature from the target frame may not be able to be matched to a proper correspondence in the current frame. The incidence of feature nonidentification is dependent on many factors unique to a particular situation. Feature nonidentification is modeled in simulation by randomly selecting a subset of target image features to not be identified/matched in the current frame. This subset of features is therefore not available as a measurement for GNC calculations during the corresponding time step.

Image feature misidentification occurs when a feature is identified and matched to an incorrect location in the image (i.e. it is an outlier). While methods like RANSAC are very

good at rejecting outliers, they do not catch all mismatches. It is modeled in the same way as misidentification: a random subset of features are chosen to be outliers for each frame, and their pixel coordinates are randomly changed to a different location in the image.

Finally, image feature depth error occurs as a result of stereo triangulation methods incorrectly estimating the depth to a specific feature. The depth error associated with stereo can be modeled as in [133]:

$$\epsilon_z = \frac{z^2}{bf}\epsilon_d \tag{A.2}$$

where z is the actual depth from the camera to a feature, b is the baseline (distance between cameras), f is the camera focal length (in pixels, assumed to be equal for each camera), and ϵ_d is the matching error in pixels. In simulation, ϵ_z is the standard deviation of white Gaussian noise added to the true value of depth.

A.1.3 Goal Image Generation.

Either of the options for generating goal images discussed in Section 3.3 can be used to generate a goal image for the maneuver in simulation. If the first option is selected, a zoomed-in image of the target is generated with its ROI perpendicular to the optical axis. This image can be rotated or shifted to properly define the goal pose.

It the second option is selected (i.e. the goal image is generated using a perspective warp as proposed in Section 3.3.2), the user selects a quadrangle ROI on the target prior to maneuvering, as depicted in Figure A.1. The user also the selects the desired locations of the corners of this ROI in the goal image. The simulation estimates the homography matrix to perform a perspective warp from the current image view to the desired view. Target feature locations in the goal image are then calculated using (3.14), and maneuvering can commence.

Appendix B: Control and Autonomy Space proximity Robot (CASpR)

This appendix provides a brief overview of the Air Force Institute of Technology (AFIT) Control and Autonomy Space proximity Robot (CASpR).

B.1 Basic Description

CASpR is dual gantry style robotic machine for testing RPO-related research algorithms using representative hardware components. The initial operating capability of CASpR enables 4 degrees of freedom (DOF) motion of two spacecraft bodies in the Local-Vertical-Local-Horizontal (LVLH) reference frame. The overall design intent of CASpR is to be a modular, extensible, low-cost testbed capable of supporting a variety of research pertaining to autonomous spacecraft RPO. A photo of CASpR is shown in Figure B.1.

A number of robotic RPO testbeds exist in the realm of research and academia (a brief survey of several of these testbeds can be found in [161], including a detailed description of the Naval Postgraduate School's POSEIDYN Testbed). All of these testbeds utilize air-bearing surfaces, on which robotic spacecraft simulators float. Due to cost, time, and space limitations, a different approach was taken to develop CASpR. CASpR uses two gantry style machines, similar to that used for 3D printing or CNC, to independently actuate two spacecraft bodies. In fact, the basic design of CASpR was inspired by an open-source CNC machine design, the Mostly Printed CNC (MPCNC) [162].

Relevant coordinate frames for CASpR are shown in Figure B.2. For this work, these frames include: an LVLH frame, chaser and target body frames, and a camera frame. Translational control commands are converted from the spacecraft body frame $\{B\}$ to the LVLH frame $\{L\}$ via a rotation,

$$\boldsymbol{u}^L = \boldsymbol{R}_B^L \boldsymbol{u}^B, \tag{B.1}$$

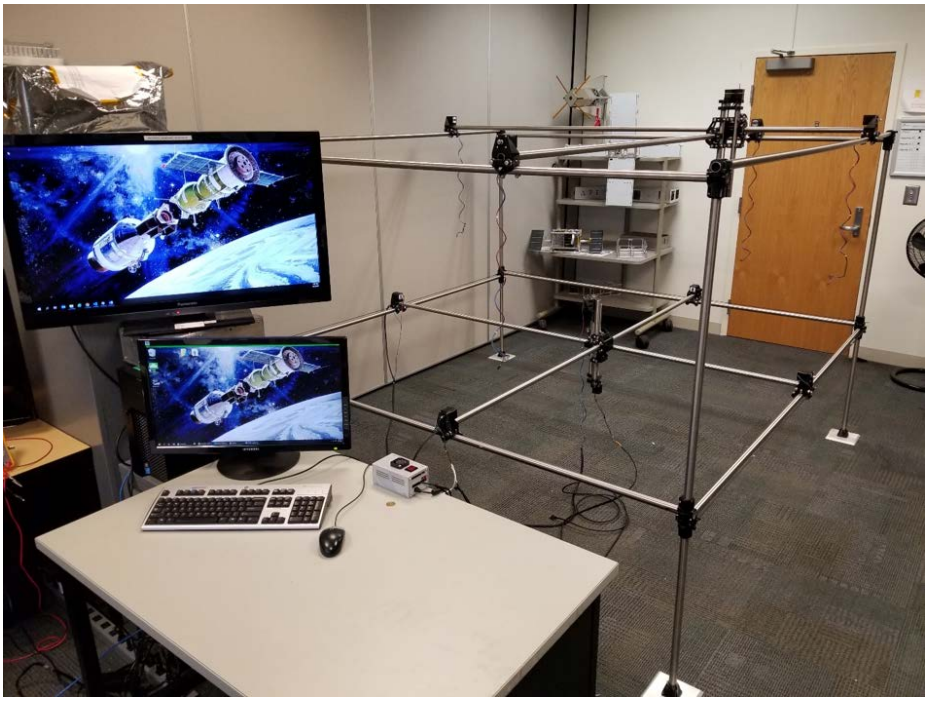

Figure B.1: Picture of CASpR

where $\boldsymbol{R}_B^L$ is a Direction Cosine Matrix (DCM) rotating about the Z axis:

$$\boldsymbol{R}_B^L = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{B.2}$$

If a camera is used in the control loop, as in this work, convention is for the camera's optical axis to be aligned with the camera frame Z axis, as described in Section 2.4.1. Therefore, an additional rotation is necessary to transform from the camera frame $\{C\}$ to the body frame,

$$\boldsymbol{u}^B = \boldsymbol{R}_C^B \boldsymbol{u}^C, \tag{B.3}$$

where $\boldsymbol{R}_B^L$ is the DCM:

$$\boldsymbol{R}_B^L = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}. \tag{B.4}$$

Because the $\{B\}$ frame and $\{L\}$ Z axes are aligned and there is only one rotational degree of freedom, no rotation is necessary for the angular acceleration command, α_c.

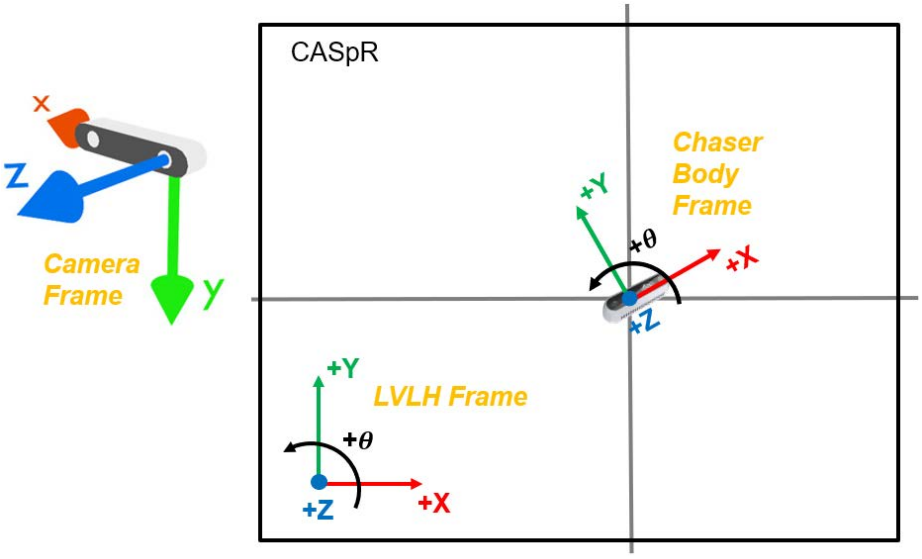

Figure B.2: CASpR coordinate frames

B.2 CASpR Architecture

An overview of the CASpR hardware architecture is shown in Figure B.3. A sensor, such as the D435 stereo camera used for this work, provides measurements to a flight computer, which runs guidance, navigation, and control (GNC) software and generates acceleration commands. The flight computer sends these commands to a Low Level Controller (LLC) for each gantry machine. The primary purpose of the LCCs is to generate hardware control commands for each machine's stepper motors. The LLCs also send back

information on the position of each machine which is utilized in analysis of experimental runs.

Each LLC is composed of an Arduino Mega 2560 microcontroller and a RAMPS 1.4 shield. The microcontroller propagates linearized relative motion dynamics as discussed in Section 4.2.1.2. The position and velocity calculated by integrating these dynamics are converted into stepper motor commands.

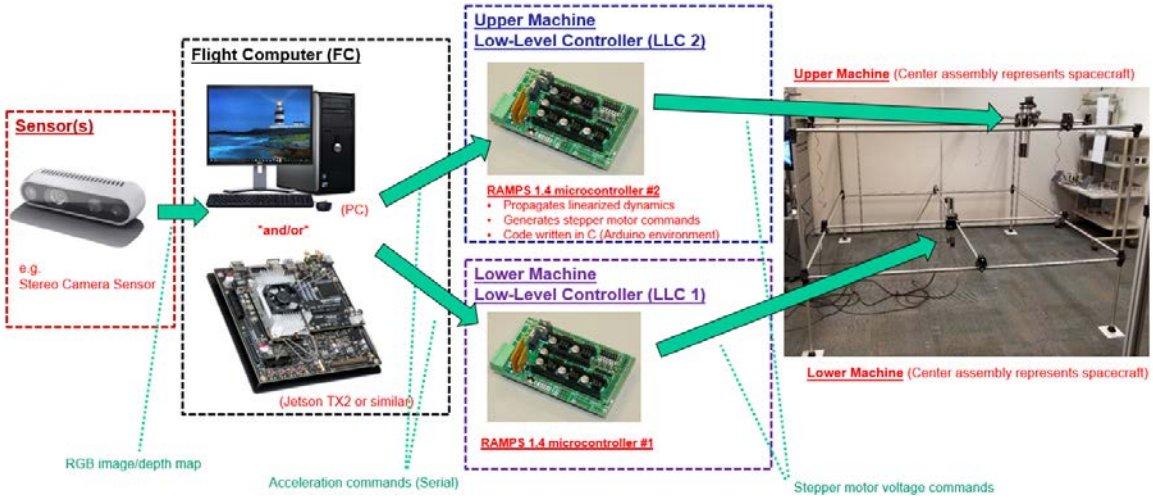

Figure B.3: Hardware architecture of CASpR

Each of CASpR's gantry machines operate using NEMA 17 stepper motors. These motors are used for several reasons: they have good low-speed torque, excellent precision, and high resolution (3,200 steps per revolution or more, depending on driver used).

Code to operate CASpR is written in Python in order to utilize the extensive computer vision functionality of the OpenCV library [37]. A Python class was written to operate CASpR. Two modes of operation are available: 'manual mode' which allows an operator to manually position each machine, and 'GNC mode' which utilizes acceleration commands generated by flight software to control each gantry machine.

Bibliography

[1] Brian Weeden. 2009 Iridium-Cosmos Collision Fact Sheet, 2010.

[2] J. C. Liou. An active debris removal parametric study for LEO environment remediation. *Advances in Space Research*, 47(11):1865–1876, 2011.

[3] Nathan Strout. The Pentagon wants to extend the life of satellites and refuel on orbit.

[4] G. Casonato and G.B. Palmerini. Visual techniques applied to the ATV/1SS rendezvous monitoring. pages 613–625, 2005.

[5] David C. Woffinden and David K. Geller. Navigating the road to autonomous orbital rendezvous. *Journal of Spacecraft and Rockets*, 44(4):898–909, 2007.

[6] Xiaodong Du, Bin Liang, Wenfu Xu, and Yue Qiu. Pose measurement of large non-cooperative satellite based on collaborative cameras. *Acta Astronautica*, 68(11-12):2047–2065, 2011.

[7] Wigbert Fehse. *Automated Rendezvous and Docking of Spacecraft*. Cambridge university press, vol.16 edition, 2003.

[8] David C. Woffinden and David K. Geller. Relative Angles-Only Navigation and Pose Estimation For Autonomous Orbital Rendezvous. *Journal of Guidance, Control, and Dynamics*, 30(5):1455–1469, 2007.

[9] Roberto Opromolla, Giancarmine Fasano, Giancarlo Rufino, and Michele Grassi. A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. *Progress in Aerospace Sciences*, 93(July):53–72, 2017.

[10] Mitsushige Oda. Experiences and lessons learned from the ETS-VII robot satellite. *Proceedings - IEEE International Conference on Robotics and Automation*, 1(April):914–919, 2000.

[11] Marco D'Errico, editor. *Distributed Space Missions for Earth System Monitoring*. 2013.

[12] Thomas M. Davis and David Melanson. XSS-10 microsatellite flight demonstration program results. In *Proc. SPIE 5419, Spacecraft Platforms and Infrastructure*, number doi: 10.1117/12.544316, 2004.

[13] Robert B. Friend. Orbital Express program summary and mission overview. *Sensors and Systems for Space Applications II*, 6958(April 2008):695803, 2008.

[14] Robin Larsson, Simone D Amico, and Jean-sebastien Ardaens. Prisma Formation Flying Demonstrator : Overview and Conclusions from the AAS 12-072 The PRISMA Formation Flying Demonstrator : Overview and Conclusions from the Nominal Mission. (August 2014), 2012.

[15] Per Bodin, Robin Larsson, Fredrik Nilsson, Camille Chasset, Ron Noteborn, and Matti Nylund. PRISMA: An In-Orbit Test Bed for Guidance, Navigation, and Control Experiments. *Journal of Spacecraft and Rockets*, 46(3):615–623, 2009.

[16] T. Grelier, P. Y. Guidotti, M. Delpech, J. Harr, J. B. Thevenet, and X. Leyre. Formation flying radiofrequency instrument: First flight results from the PRISMA mission. *5th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing, NAVITEC*, pages 1–8, 2010.

[17] Simone Damico, Jean Sebastien Ardaens, and Sergio De Florio. Autonomous formation flying based on GPS - PRISMA flight results. *Acta Astronautica*, 82(1):69–79, 2013.

[18] Tatsuaki Hashimoto and Takashi Kubota. Vision-based guidance , navigation , and control of Hayabusa spacecraft - Lessons learned from real operation -. In *IFAC Proceedings Volumes*, volume 43, pages 259–264. IFAC, 2010.

[19] Takashi Kubota, Shujiro Sawai, Tatsuaki Hashimoto, and Jun Kawaguchi. Robotics and Autonomous Technology for Asteroid Sample Return Mission. *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics*, pages 31–38, 2005.

[20] Sei-ichiro Watanabe, Yuichi Tsuda, Makoto Yoshikawa, Satoshi Tanaka, Takanao Saiki, and Satoru Nakazawa. Hayabusa2 Mission Overview. *Space Science Reviews*, 208, 2017.

[21] Yuichi Tsuda, Makoto Yoshikawa, Masanao Abe, and Hiroyuki Minamino. System design of the Hayabusa 2 — Asteroid sample return mission to 1999 JU3. *Acta Astronautica*, 91:356–362, 2013.

[22] L. Alfriend, K., Vadali, S. R., Gurfil, P., How, J., & Breger. *Spacecraft formation flying: Dynamics, control and navigation*. Elsivier, 2nd edition, 2009.

[23] W H Clohessy and R S Wiltshire. Terminal Guidance System for Satellite Rendezvous. *Journal of the Aerospace Sciences*, 27.9:653–658, 1960.

[24] Terry Alfriend. Dynamics and Control of Formation Flying Satellites In Earth Orbit.

[25] Bong Wie. *Space Vehicle Dynamics and Control*. American Institute of Aeronautics and Astronautics, 2nd edition, 2008.

[26] Shay Segal and Pini Gurfil. Effect of Kinematic Rotation-Translation Coupling on Relative Spacecraft Translational Dynamics. *Journal of Guidance, Control, and Dynamics*, 32(3):1045–1050, 2009.

[27] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB®*. Springer, 2nd edition, 2017.

[28] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[29] Peter Decker, Dietrich Paulus, and Tobias Feldmann. Dealing with degeneracy in essential matrix estimation. In *International Conference on Image Processing*, number December 2014, 2008.

[30] Anubhav Agarwal, C V Jawahar, and P J Narayanan. A Survey of Planar Homography Estimation. Technical report, Centre for Visual Information Technology, 2005.

[31] Martin A. Fischler and Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with. *Communications of the ACM*, 24(6):381–395, 1981.

[32] Ezio Malis and Manuel Vargas. Deeper understanding of the homography decomposition for vision-based control. Technical report, INRIA, 2007.

[33] R. Taktak, A., Ganney, P., Long, D., & Axell, editor. *Clinical engineering: a handbook for clinical and biomedical engineers*. Academic Press, London, 2019.

[34] E.R. Davies. *Computer and Machine Vision*. Academic Press, 4th edition, 2012.

[35] European Space Agency. Pose Estimation Challenge. https://kelvins.esa.int/satellite-pose-estimation-challenge, 2019.

[36] Peter Janku, Karel Koplik, Tomas Dulik, and Istvan Szabo. Comparison of tracking algorithms implemented in OpenCV. In *MATEC Web of Conferences*. Sciences, EDP, 2016.

[37] G Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[38] Adnan Brdjanin, Nadja Dardagan, Dzemil Dzigal, and Amila Akagic. Single Object Trackers in OpenCV: A Benchmark. *INISTA 2020 - 2020 International Conference on INnovations in Intelligent SysTems and Applications, Proceedings*, (August), 2020.

[39] Alan Lukežič, Tomáš Vojíř, Luka Čehovin Zajc, Jiří Matas, and Matej Kristan. Discriminative Correlation Filter Tracker with Channel and Spatial Reliability. *International Journal of Computer Vision*, 126(7):671–688, 2018.

[40] Shuai Liu, Dongye Liu, Gautam Srivastava, Dawid Połap, and Marcin Woźniak. Overview and methods of correlation filter algorithms in object tracking. *Complex & Intelligent Systems*, (0123456789), 2020.

[41] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, pages 91–110, 2004.

[42] Aleš Leonardis, Horst Bischof, Axel Pinz, Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded Up Robust Features. *Computer Vision – ECCV 2006*, 3951:404–417–417, 2006.

[43] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: and efficient alternative to SIFT and SURF. *2011 IEEE International Conference on Computer Vision (ICCV)*, 2011.

[44] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP 2009 - Proceedings of the 4th International Conference on Computer Vision Theory and Applications*, 1:331–340, 2009.

[45] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[46] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[47] Lucas Bruce D. and Kanade Takeo. An iterative image registration technique with an application to stereo vision.pdf, 1881.

[48] Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. Technical Report Part 3, 1991.

[49] Simon J.D. Prince. *Computer vision: models, learning and inference*. Cambridge university press, 2012.

[50] Ullman S. The Interpretation of Structure from Motion. *Proceedings of the Royal Society of London. Series B.*, 203(1153):405–426, 1979.

[51] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1, 2003.

[52] Jordi Pagès, Joaquim Salvi, Rafael García, and Carles Matabosch. Overview of coded light projection techniques for automatic 3D profiling. *Proceedings - IEEE International Conference on Robotics and Automation*, 1:133–138, 2003.

[53] Yoav Y. Schechner and Nahum Kiryati. Depth from Defocus vs. stereo: How different really are they? *International Journal of Computer Vision*, 39(2):141–162, 2000.

[54] Robert A Raynor. *Range Finding with a Plenoptic Camera*. PhD thesis, 2014.

[55] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Proceedings - IEEE Workshop on Stereo and Multi-Baseline Vision, SMBV 2001*, (1):131–140, 2001.

[56] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[57] Mohit Gupta, Qi Yin, and Shree K Nayar. Structured Light In Sunlight. In *Proceedings of the IEEE International Conference on Computer Vision*, 2013.

[58] H Klaus, W Nassir, and Martin Lingenauber. Benefits of plenoptic cameras for robot vision during close range on-orbit servicing maneuvers Benefits of Plenoptic Cameras for Robot Vision During Close Range On-Orbit Servicing Maneuvers. 2017.

[59] Giovanni Palmerini, Marco Sabatini, and Paolo Gasbarri. Analysis and tests of visual based techniques for orbital rendezvous operations. In *IEEE Aerospace Conference.*, volume 1, 2013.

[60] P Gasbarri, M Sabatini, and G B Palmerini. Ground tests for vision based determination and control of formation flying spacecraft trajectories. *Acta Astronautica*, 102:378–391, 2014.

[61] Deutsches Zentrum. European Proximity Operations Simulator 2 . 0 (EPOS) - A Robotic-Based Rendezvous and Docking Simulator. *Journal of large-scale research facilities*, 3(A107), 2017.

[62] Heike Benninghoff, Toralf Boge, and Tristan Tzschichholz. Hardware-in-the-Loop Rendezvous Simulation Involving an Autonomous Guidance, Navigation and Control System. *Advances in the Astronautical Sciences*, 145(April 2014):953–972, 2012.

[63] Sumant Sharma and Simone D'Amico. Reduced-dynamics pose estimation for non-cooperative spacecraft rendezvous using monocular vision. In *38th AAS Guidance and Control Conference*, pages 361–385, Breckenridge, Colorado, 2017.

[64] Antoine Petit, Keyvan Kanani, and Eric Marchand. Vision-based Detection and Tracking for Space Navigation in a Rendezvous Context. *Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, i-SAIRAS˜2012*, 2012.

[65] Simone D'Amico, Mathias Benn, and John L. Jørgensen. Pose estimation of an uncooperative spacecraft from actual space imagery. *International Journal of Space Science and Engineering*, 2(2):171, 2014.

[66] Antoine Petit, Eric Marchand, and Keyvan Kanani. Vision-based Space Autonomous Rendezvous: A Case Study. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 619–624. IEEE, 2011.

[67] Peter Corke, Rohan Paul, Winston Churchill, and Paul Newman. Dealing with Shadows : Capturing Intrinsic Scene Appearance for Image-based Outdoor Localisation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[68] Graham D Finlayson, Steven D Hordley, and Mark S Drew. Removing Shadows from Images. In *European conference on computer vision*, pages 1–14, Springer, Berlin, Heidelberg, 2002.

[69] Graham D Finlayson, Mark S Drew, and Cheng Lu. Intrinsic Images by Entropy Minimization. In *European conference on computer vision*, pages 582–595, Springer, Berlin, Heidelberg, 2004.

[70] Warunika Ranaweera and Mark S Drew. Shadow Removal Using Illumination Invariant Image Formation. 2015.

[71] Peter S Maybeck. *Stochastic models, estimation, and control*. Academic press, vol. 1 edition, 1982.

[72] Yaakov Bar-Shalom, X.-Rong Li, and Kirubarajan Thiagalingam. *Estimation with Applications To Tracking and Navigation*. John WIley & Sons, Inc, 1st edition, 2001.

[73] Roberto Opromolla, Giancarmine Fasano, Giancarlo Rufino, and Michele Grassi. Pose Estimation for Spacecraft Relative Navigation Using Model-Based Algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 53(1):431–447, 2017.

[74] Sebastian Thrun. *Probabilistic robotics*. MIT Press, Cambridge, MA, 2005.

[75] Hugh Durrant-Whyte and Tim Bailey. Simultaneous Localization and Mapping: Part I. *IEEE Robotics & Automation Magazine*, 13(2):1066–1069, 2006.

[76] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proc. AAAI Nat. Conf. Artif. Intell.*, pages 593–598, 2002.

[77] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. *IJCAI*, pages 1151–1156, 2003.

[78] Tim Bailey and Hugh Durrant-Whyte. Simultaneous Localization and Mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(September):108–117, 2006.

[79] Giorgio Grisetti, K Rainer, and Cyrill Stachniss. A Tutorial on Graph-Based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[80] Arnaud Doucett, Nando De Freitast, and Stuart Russent. Rao-Blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intel- ligence*, pages 176–183, 2000.

[81] Arne Sonnenburg, Marcel Tkocz, Klaus Janschek, and Technische Universit. *EKF-SLAM based Approach for Spacecraft Rendezvous Navigation with Unknown Target Spacecraft*, volume 43. IFAC, 2010.

[82] Sean Augenstein and Stephen M Rock. Improved Frame-to-Frame Pose Tracking during Vision-Only SLAM / SFM with a Tumbling Target. In *2011 IEEE International Conference on Robotics and Automation*, 2011.

[83] Frank Schnitzer, Klaus Janschek, and Georg Willich. Experimental Results for Image-based Geometrical Reconstruction for Spacecraft Rendezvous Navigation with Unknown and Uncooperative Target Spacecraft. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5040–5045, 2012.

[84] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

[85] Brent Edward Tweddle. *Computer Vision-Based Localization and Mapping of an Unkown, Uncooperative and Spinning Target for Spacecraft Proximity Operations*. Doctoral dissertation, Massachusetts Institute of Technology, 2013.

[86] Brent E. Tweddle, Alvar Saenz-Otero, John J. Leonard, and David W. Miller. Factor Graph Modeling of Rigid-body Dynamics for Localization , Mapping , and Parameter Estimation of a Spinning Object in Space. *Journal of Field Robotics*, 32(6):897–933, 2015.

[87] Michael Kaess. *Incremental smoothing and mapping*. PhD thesis, Georgia Institute of Technology, 2008.

[88] Shai Segal, Avishy Carmi, and Pini Gurfil. Stereovision-Based Estimation of Relative Dynamics Between Noncooperative Satellites: Theory and Experiments. *IEEE Transactions on Control Systems Technology*, 22(2):568–584, 2014.

[89] Vincenzo Pesce, Michèle Lavagna, and Riccardo Bevilacqua. Stereovision-based pose and inertia estimation of unknown and uncooperative space objects. *Advances in Space Research*, 59(1):236–251, 2017.

[90] Vincenzo Capuano, Kyunam Kim, Juliette Hu, Alexei Harvard, and Soon Jo Chung. Monocular-based pose determination of uncooperative known and unknown space objects. *Proceedings of the International Astronautical Congress, IAC*, 2018-Octob(October):1–5, 2018.

[91] Christopher G Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 762–818, 1988.

[92] Richard Zappulla, Hyeongjun Park, Josep Virgili-Llop, and Marcello Romano. Experiments on Autonomous Spacecraft Rendezvous and Docking Using an Adaptive Artificial Potential Field Approach. 2016.

[93] Qinglei Hu, Yueyang Liu, and Youmin Zhang. Control of non-cooperative spacecraft in final phase proximity operations under input constraints. *Control Engineering Practice*, 87(June 2018):83–96, 2019.

[94] R Bevilacqua, T Lehmann, and M Romano. Development and experimentation of LQR/APF guidance and control for autonomous proximity maneuvers of multiple spacecraft. *Acta Astronautica*, 68(7-8):1260–1275, 2011.

[95] Shawn Baxter McCamish, Marcello Romano, and Xiaoping Yun. Autonomous distributed control of simultaneous multiple spacecraft proximity maneuvers. *IEEE Transactions on Automation Science and Engineering*, 7(3):630–644, 2010.

[96] Costantinos Zagaris. *Autonomous Spacecraft Rendezvous with a Tumbling Object: Applied Reachability Analysis and Guidance and Control Strategies*. PhD thesis, Naval Post Graduate School, 2018.

[97] Nicoletta Bloise, Elisa Capello, Matteo Dentis, and Elisabetta Punta. Obstacle avoidance with potential field applied to a rendezvous maneuver. *Applied Sciences (Switzerland)*, 7(10), 2017.

[98] Qinglei Hu, Xiaodong Shao, and Wen-hua Chen. Robust Fault-tolerant Tracking Control for Spacecraft Proximity Operations Using Time-varying Sliding Mode. *IEEE Transactions on Aerospace and Electronic Systems*, 2017.

[99] Feng Zhang and Guang-ren Duan. Integrated Relative Position and Attitude Control of Spacecraft in Proximity Operation Missions Feng. *International Journal of Automation and Computing*, 9(August):342–351, 2012.

[100] Y. Shirai and H Inoue. Guiding a Robot by Visual Feedback in Assembling Tasks. *Pattern Recognition*, 5:99–108, 1973.

[101] J. Hill and W.T. Park. Real time control of a robot with a moble camera. In *Proc. 9th ISIR*, pages 233–246, Washington, D.C., 1979.

[102] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[103] Peter I. Corke. *Visual Control of Robots : High-Performance Visual Servoing*. Research Studies Press, Taunton, UK, 1996.

[104] Seth Hutchinson, Gregory D. Hager, and Peter Corke. A Tutorial on Visual Servo Control. *IEEE Transactions on Robotics and Automation*, 12(6):651–670, 1998.

[105] William J Wilson, Carol C Williams Hulls, Student Member, and Graham S Bell. Relative End-Effector Control Using Cartesian osition Based Visual Servoing. *IEEE Transactions on Robotics and Automation*, 12(5), 1996.

[106] Fran¸cois Chaumette. Visual servoing. In K. Ikeuchi, editor, *Computer Vision: A Reference Guide*, pages 869–874. Springer, 2014.

[107] Fraṇois Chaumette and Setrh Hutchinson. Visual servo control. I. Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90, 2006.

[108] Francois Chaumette and Seth Andrew Hutchinson. Visual Servo Control, Part II: Advanced Approaches. *IEEE Robotics and Automation Magazine*, 14(March):109–118, 2007.

[109] François Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. *The Confluence of Vision and Control, LNCIS Series*, (No 237):66–78, 1998.

[110] Ezio Malis, Francois Chaumette, and Sylvie Boudet. 2-1 / 2-D Visual Servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, 1999.

[111] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. *Proceedings - IEEE International Conference on Robotics and Automation*, 2004(2):1843–1848, 2004.

[112] François Chaumette, Seth Hutchinson, and Peter Corke. *Visual Servoing*. Springer, 2nd edition, 2016.

[113] Lingfeng Deng, F. Janabi-Sharifi, and W. J. Wilson. Stability and robustness of visual servoing methods. *Proceedings - IEEE International Conference on Robotics and Automation*, 2(May):1604–1609, 2002.

[114] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[115] Nicholas R. Gans, Seth A. Hutchinson, and Peter I. Corke. Performance tests for visual servo control systems with application to partitioned approaches to visual servo control. *International Journal of Robotics Research*, 22(10-11):955–981, 2003.

[116] Peter I Corke and Seth A Hutchinson. A New Partitioned Approach to Image-Based Visual Servo. *IEEE Transactions on Robotics and Automation*, 17(4):507–515, 2001.

[117] Ezio Malis. Visual Servoing Invariant to Changes in Camera. *INRIA*, RR-4309, 2001.

[118] Koichiro Deguchi, Mathematical Engineering, and Information Physics. Optimal Motion Control for Image-Based Visual Servoing by Decoupling Translation and Rotation. In *1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, number Vol. 2, 1998.

[119] Mohammad Keshmiri, Wen Fang Xie, and Abolfazl Mohebbi. Augmented image-based visual servoing of a manipulator using acceleration command. *IEEE Transactions on Industrial Electronics*, 61(10):5444–5452, 2014.

[120] Ye Shi and Bin Liang. Modeling and Simulation of Space Robot Visual Servoing for Autonomous Target Capturing. In *2012 IEEE International Conference on Mechatronics and Automation*, number 60805033, pages 2275–2280. IEEE, 2012.

[121] Javier Pérez Alepuz, M Reza Emami, and Jorge Pomares. Direct image-based visual servoing of free-floating space manipulators. *Aerospace Science and Technology*, 55:1–9, 2016.

[122] Noriyasu Inaba, Mitsushige Oda, and Masato Hayashi. Visual Servoing of Space Robot for Autonomous Satellite Capture. *Transactions of the Japan Society for Aeronautical and Space Sciences*, 46(153):173–179, 2003.

[123] Guoliang Zhang, Hong Liu, Jie Wang, and Zainan Jiang. Vision-based system for satellite on-orbit self-servicing. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, (2006):296–301, 2008.

[124] Naoya Yanagi and Fuyuto Terui. Visual servo motion control of a spacecraft around an asteroid using feature points. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 17(1 PART 1):13012–13015, 2008.

[125] Sungwook Cho, Sungsik Huh, and David Hyunchul Shim. Visual Detection and Servoing for Automated Docking of Unmanned Spacecraft. *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan*, 12(APISAT-2013):a107–a116, 2015.

[126] P. Mithun, Harit Pandya, Ayush Gaud, Suril V. Shah, and K. Madhava Krishna. Image Based Visual Servoing for Tumbling Objects. *IEEE International Conference on Intelligent Robots and Systems*, pages 2901–2908, 2018.

[127] Jorge Pomares, Leonard Felicetti, Javier Perez, and M. Reza Emami. Spacecraft visual servoing with adaptive zooming for non-cooperative rendezvous. *IEEE Aerospace Conference Proceedings*, pages 1–8, 2018.

[128] John G Allen, Richard Y D Xu, and Jesse S Jin. Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces. In *Proceedings of the Pan-Sydney area workshop on Visual information processing*. Australian Computer Society, Inc, 2004.

[129] Jorge Pomares, Leonard Felicetti, Javier Pérez, and M. Reza Emami. Concurrent image-based visual servoing with adaptive zooming for non-cooperative rendezvous maneuvers. *Advances in Space Research*, 61(3):862–878, 2018.

[130] Eric R. Prince. *Optimal Finite Thrust Guidance Methods for Constrained Satellite Proximity Operations Inspections Maneuvers*. Dissertation, Air Force Institute of Technology, 2019.

[131] F. Chaumette and A. Santos. Tracking a Moving Object by Visual Servoing. *IFAC Proceedings Volumes*, 26(2):643–648, 1993.

[132] Stefano Soatto, Ruggero Frezza, and Pietro Perona. Motion estimation via dynamic vision. *IEEE Transactions on Automatic Control*, 41(3):393–413, 1996.

[133] David Gallup, Jan Michael Frahm, Philippos Mordohai, and Marc Pollefeys. Variable baseline/resolution stereo. *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.

[134] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

[135] Intel® RealSense™ Depth Camera D435. https://www.intelrealsense.com/depth-camera-d455/, 2020.

[136] Anders Grunnet-Jepsen, John N Sweetser, Paul Winer, Akihiro Takagi, and John Woodfill. Projectors for Intel ® RealSense™ Depth Cameras D4xx. 2019.

[137] Steven W. Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Pub, San Diego, CA, 1st edition, 1997.

[138] Anders Grunnet-jepsen and Dave Tong. Depth Post-Processing for Intel ® RealSense ™ D400 Depth Cameras. *Report*, 2018.

[139] Eduardo S.L. Gastal and Manuel M. Oliveira. Domain Transform for Edge-Aware Image and Video Processing. *ACM Transactions on Graphics*, 30(4):1–12, 2011.

[140] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel corporation*, pages 1–10, 20001.

[141] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. *Proceedings - International Conference on Pattern Recognition*, pages 2756–2759, 2010.

[142] Grapple Fixture, Wikipedia. https://en.wikipedia.org/wiki/Grapple_fixture, 2020.

[143] Xiangtian Zhao, M. Reza Emami, and Shijie Zhang. Image-based control for rendezvous and synchronization with a tumbling space debris. *Acta Astronautica*, 179(December 2020):56–68, 2021.

[144] Youcef Mezouar and François Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, pages 534–549, 2009.

[145] Aidin Foroughi. *Stochastic Optimal Control with Application in Visual Servoing*. PhD thesis, University of Tehran, 2010.

[146] Stefan Hinterstoisser, Cedric Cagniart, Student Members, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient Response Maps for Real-Time Detection of Texture-Less Objects. In *IEEE transactions on pattern analysis and machine intelligence*, pages 876–888. IEEE, 2011.

[147] Klaas Klasing, Daniel Althoff, Dirk Wollherr, and Martin Buss. Comparison of surface normal estimation methodsfor range sensing applications. *Proceedings - IEEE International Conference on Robotics and Automation*, (May 2014):3206–3211, 2009.

[148] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab. Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. *IEEE International Conference on Intelligent Robots and Systems*, pages 2684–2689, 2012.

[149] WJ Tam, G Alain, L Zhang, and T Martin. Smoothing depth maps for improved steroscopic image quality. *Three-Dimensional TV, Video, and Display III*, 5599, 2004.

[150] Karl Johan Åström and Richard M. Murray. *Feedback systems*. Princeton university press, Princeton, NJ, 2010.

[151] Bong Wie and Peter M. Barb. Quaternion feedback for spacecraft large angle maneuvers. *Journal of Guidance, Control, and Dynamics*, 8(3):360–365, 1985.

[152] B. Wie, H. Weiss, and A. Arapostathis. A quaternion feedback regulator for spacecraft eigenaxis rotations. *Journal of Guidance, Control, and Dynamics*, 12.3:375–380, 1989.

[153] Wyatt J Harris, Richard Cobb, and Costantinos Zagaris. Visual Servoing Using Predominant Surface Normals for Spacecraft Final Approach. In *2020 AAS/AIAA Astrodynamics Specialist Conference*, pages 4067–4084, Lake Tahoe, CA, 2020.

[154] Wyatt Harris, Dax Linville, Joshuah Hess, and Richard Cobb. Development of GNC for Optimal Relative Spacecraft Trajectories. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1476–1487, Portland, Oregon, 2020.

[155] Wyatt J Harris, Richard G Cobb, and Clark N Taylor. Visual Servoing for Final Approach Phase of Spacecraft Proximity Operations with Unknown Targets. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1606–1617, Portland, Oregon, 2020. IEEE.

[156] Wyatt Harris, Richard Cobb, Clark Taylor, and Costantinos Zagaris. Visual Servoing for Spacecraft Proximity Operations with Unknown Targets. *Manuscript accepted for publication to Journal of DoD Research and Engineering*, 2021.

[157] Wyatt Harris, Richard Cobb, Clark Taylor, and Costantinos Zagaris. Surface Normal Visual Servoing for Maneuver with Moving Unknown Targets. *Manuscript submitted for publication to IEEE Transactions on Robotics*, 2021.

[158] Rasterization: a Practical Implementation. https://www.scratchapixel.com/., 2020.

[159] Chun Yang, Ananth Vadlamani, Andrey Soloviev, Michael Veth, and Clark Taylor. Feature matching error analysis and modeling for consistent estimation in vision-aided navigation. *Navigation, Journal of the Institute of Navigation*, 65(4):609–628, 2018.

[160] Tomer Shtark and Pini Gurfil. Tracking a non-cooperative target using real-time stereovision-based control: An experimental study. *Sensors (Switzerland)*, 17(4), 2017.

[161] Richard Zappulla, Josep Virgili-Llop, Costantinos Zagaris, Hyeongjun Park, and Marcello Romano. Dynamic air-bearing hardware-in-the-loop testbed to experimentally evaluate autonomous spacecraft proximity maneuvers. *Journal of Spacecraft and Rockets*, 54(4):825–839, 2017.

[162] V1 Engineering. Mostly Printed CNC. https://www.v1engineering.com/, 2020.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
16-09-2021	Doctoral Dissertation	Sep 2018-Sep 2021

4. TITLE AND SUBTITLE

Visual Navigation and Control for Spacecraft Proximity Operations with Unknown Targets

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Harris, Wyatt J., Major, 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-21-DS-094

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

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

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

Many current and future spacecraft missions must conduct rendezvous and proximity operations (RPO) with resident space objects (RSOs). An important subset of spacecraft RPO that is yet to be demonstrated on-orbit involves final approach maneuvers with respect to RSOs where no information (such as geometry, inertia, relative velocity, etc.) is known about the target a priori, and no information is actively provided by the target during maneuvering. Such operation with respect to 'unknown' targets represents an important possible mission set for Department of Defense spacecraft and is the subject of this research. Two visual servoing frameworks capable of autonomously controlling complex RPO maneuvers with respect moving unknown targets are proposed. Computer vision pipelines for each framework are developed and demonstrated. Both pipelines assume a stereo camera is the only sensor available for relative navigation during maneuvering and require only a single training image of the target to initiate maneuvering. A six degree of freedom simulation and the Air Force Institute of Technology's robotic RPO simulator Control and Autonomy Space proximity Robot (CASpR) are used to demonstrate the methods during complex maneuvers with respect to moving unknown targets. It is shown that both frameworks are capable of being implemented in real time using low cost, size, weight, and power hardware and require minimal pre-maneuver inspection of the target. The methods presented in this work represent compelling alternatives to other proposed methods for RPO with unknown targets, as they do not require estimation of target inertia or rely on complex simultaneous localization and mapping algorithms to generate potentially brittle relative pose estimates.

15. SUBJECT TERMS
Computer vision, rendezvous and proximity operations (RPO), satellite inspection

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Dr. Richard G. Cobb, AFIT/ENY
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	U	297	**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U			(937) 255-3636, x4559