

Development of Resource-Constrained Sensors and Actuators for In-Space Satellite Docking and Servicing

by

Duncan Lee Miller

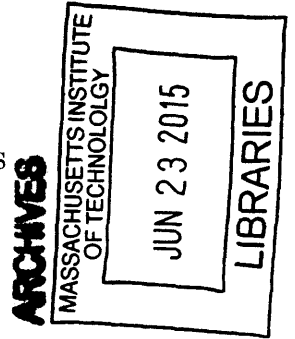
Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

© Massachusetts Institute of Technology 2015. All rights reserved.



Signature redacted

Author
Department of Aeronautics and Astronautics

Signature redacted

Certified by
David W. Miller
Professor of Aeronautics and Astronautics

Signature redacted

Certified by
Alvar Saenz-Otero
Principal Research Scientist

Signature redacted

Accepted by
Paulo C. Lozano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Development of Resource-Constrained Sensors and Actuators for In-Space Satellite Docking and Servicing

by

Duncan Lee Miller

Submitted to the Department of Aeronautics and Astronautics
on May 21, 2015, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Most satellites on-orbit today are not intended to physically approach or interact with other spacecraft. However, the robotic servicing of orbiting assets will be an economically desirable (and often scientifically necessary) capability in future space enterprises. With the right set of tools and technologies, satellites will be able to autonomously refuel, repair, or replace each other. This has the potential to extend mission lifetimes, reduce orbital debris and make space more sustainable. Spacecraft may also assemble on-orbit into larger aggregate spaceflight systems, with applications to sparse aperture telescopes, solar power stations, fuel depots and space habitats. The purpose of this thesis is to address the highest risk elements associated with the docking and servicing of satellites: the sensors, actuators, and associated algorithms.

First, a peripheral agnostic robotics platform is introduced, upon which a suite of technology payloads may be developed. Next, a flight qualified docking port for small satellites is presented, and the results detailing its operation in a relevant environment are discussed. In addition, we review a high precision relative sensor designed to enable boresight visual docking. The measurements from this optical camera are applied to a nonlinear estimator to provide the highly accurate sensing necessary for docking. Finally, a free-flying robotic arm is examined and modeled as an experimental payload for the SPHERES Facility on the International Space Station.

Thesis Supervisor: David W. Miller
Title: Professor of Aeronautics and Astronautics

Thesis Supervisor: Alvar Saenz-Otero
Title: Principal Research Scientist

Acknowledgments

The work in this thesis related to the SPHERES Docking Ports and the SPHERES Halo was supported by NASA and DARPA under the InSPIRE-II contract, #NNH13CJ23C. In addition, the entirety of this research was conducted with Government support under and awarded by the Department of Defense (DoD), Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a. The author gratefully thanks these sponsors for their generous support that enabled this research.

I would also like to thank my advisors Professor David W. Miller and Dr. Alvar Saenz-Otero for giving me the opportunity to participate and contribute to the SPHERES project. I am honored to have worked side-by-side with a first-class team of graduate and undergraduate students throughout my tenure at MIT. I would like to thank both current and past team members for establishing and maintaining such a strong program. The collection of professional engineers at Aurora Flight Sciences also deserve a large amount of credit for their commitment and contributions to the MIT Space Systems Lab. Thank you all for your tireless pursuit of a better future in space.

Contents

1	Introduction	15
1.1	Motivation for In-Space Satellite Cooperation	15
1.2	Existing and Planned Testbeds and Technical Demonstrations	19
1.2.1	SPHERES	20
1.2.2	DARPA Phoenix	22
1.2.3	RSGS	23
1.3	Thesis Approach	24
2	Framework: Peripheral Agnostic Software Architecture for a Satellite Servicing Toolbelt	27
2.1	Overview	27
2.2	Hardware Platform	28
2.3	Software Architecture	30
2.4	Halo Guest Scientist Program	34
2.4.1	Test Project Methods	34
2.4.2	PeripheralCore Methods	35
2.5	Comparison to Existing Robotic Frameworks	36
3	Actuator: Flight Hardware Design of a Rigid Androgynous Docking Port	39
3.1	Overview	39
3.2	Spacecraft Docking Port Design Space	40
3.2.1	Docking Port Classifications	41

3.2.2	Docking Port Attributes	44
3.3	Ground Prototype Design	46
3.4	Flight Requirements Flowdown	48
3.4.1	Design Drivers	48
3.4.2	Requirements Matrix	49
3.5	Flight Design Review	51
3.5.1	Mechanical Overview	54
3.5.2	Electrical and Operational Overview	56
3.5.3	Specifications Overview	57
3.6	Validation in Three Degrees-of-Freedom	57
3.6.1	Ground Test Objectives	58
3.6.2	Concept of Operations	59
3.6.3	Experimental Results	60
3.7	Validation in Six Degrees-of-Freedom	62
3.7.1	RGA Objectives	63
3.7.2	RGA Results	63
3.8	Lessons Learned	74
3.9	Planned ISS Operations	75
4	Sensor: Relative Pose Estimation of a Spin-Stabilized Spacecraft	79
4.1	Overview	79
4.1.1	Requirements	81
4.1.2	Instantiation	83
4.2	Measurements	84
4.2.1	Fiducial Markers	85
4.2.2	Marker Detection and Image Filtering	86
4.2.3	Perspective Four Point Correspondence	88
4.2.4	Camera Calibration	90
4.2.5	Camera Accuracy	93
4.2.6	Measurement Modeling	96

4.3	Dynamics	100
4.3.1	Process Noise Modeling	102
4.4	Multiplicative Extended Kalman Filter	103
4.4.1	Re-parameterize	103
4.4.2	Linearize	104
4.4.3	Propagate	106
4.4.4	Measurement Update	107
4.5	Unscented Kalman Filter	108
4.6	Validation in Simulation	110
4.6.1	Representative Convergence	111
4.6.2	Representative Errors	115
4.6.3	Modeling Variations	117
4.6.4	Montecarlo Simulation	119
4.7	Validation in Three Degrees of Freedom	121
4.7.1	Ground Test Objectives	121
4.7.2	Experimental Results	121
4.8	Validation in Six Degrees of Freedom	124
4.8.1	RGA Objectives	124
4.8.2	RGA Results	124
4.9	Summary and Conclusions	126
5	Actuator: Prototype Design of a Free-Flying Robotic Manipulator	129
5.1	Overview	129
5.2	Literary Research	130
5.3	Hardware Design	138
5.3.1	Lego Arm Overview	139
5.3.2	SPHERES Ambulant Robotic Manipulator (ARM)	140
5.4	Arm Control and Simulation	148
5.4.1	Inverse Kinematics	149
5.4.2	Dynamic Modeling	154

5.5	Verification in Three Degrees-of-Freedom	157
5.6	Future Work	159
6	Conclusion	161
A	Additional Mathematical Definitions	173
A.1	Additional Mathematical Definitions	173
A.1.1	Quaternion	173
A.1.2	Inertia Definition	174
A.1.3	Full Linearization of the Euler Dynamics	174
B	Source Code	177
B.1	Multiplicative Extended Kalman Filter	177
B.2	Unscented Kalman Filter	184

List of Figures

1-1	The Enormous Scientific Return from Hubble Would Not Have Been Possible Without Post-Launch Servicing Missions [1]	16
1-2	There Is A Large Economic Incentive To Mass Producing Aerospace Systems [2]	17
1-3	SPHERES Aboard the ISS [3]	21
1-4	The Principle Systems Involved in the DARPA Phoenix Project [4]	22
1-5	An Overview of the Robotic Servicing of Geostationary Satellites (RSGS) DARPA Mission	24
1-6	Phases of Servicing	25
2-1	Halo Prototype Hardware that Supports Peripherals On Up To Six Halo Ports	28
2-2	Currently Compatible Halo Peripherals Include Docking Ports, Stereo Vision Cameras, Control Moment Gyro's, Thermo-Imagers, Lidars, and Robotic Manipulators	29
2-3	An Object Oriented Approach to Flight Software Supports Multiple Peripherals Simultaneously	31
2-4	Current Peripheral Cores Supported By The Halo Software Framework	32
2-5	Halo Software Flow Diagram (adapted from [5])	33
2-6	Representative Thread Execution During a HaloCore Test Project	34
2-7	An Overview of Where the Halo GSP Platform Fits in the Larger Landscape of Robotics Platforms	38
3-1	The Final Flight Design of the SPHERES Universal Docking Port	40

3-2	The SPHERES Prototype Universal Docking Port Mechanism and Accompanying Avionics Board	46
3-3	The SPHERES SWARM Docking Assembly on the MIT Flat Floor	47
3-4	The SPHERES-VERTIGO-UDP Flight Assembly.	52
3-5	The Secondary In-Space Flight Configurations Through the Mechanical Standoff (left) and the Halo (right).	52
3-6	The Fleet of SPHERES Universal Docking Ports Ready for Shipment.	54
3-7	Exploded view of the internal UDP mechanism.	55
3-8	[Animation] Front View of An Animated Locking Sequence (Press Play to Start Video)	56
3-9	[Animation] Side View of An Animated Locking Sequence (Press Play to Start Video)	56
3-10	Functional Block Diagram of the UDP Electrical System	57
3-11	Glass Table Facility at MIT Used for Ground Testing of the UDP	58
3-12	Glass Table Concept of Operations	59
3-13	Velocity-Controlled Results from Three Degree of Freedom Testing on the MIT Glass Table (Adapted from [6])	61
3-14	[Animation] A Successful Docking Test Achieved by the SPHERES on the Glass Table Using the Proto-Flight Docking Ports (2X Speed; Synchronized Camera Sensing Shown in Upper Right)	62
3-15	Sample Accelerometer Response Throughout an Entire Parabola (Zero-gravity Begins at 31s).	66
3-16	Sample Gyro Response Throughout an Entire Parabola (zero-gravity begins at 31s).	67
3-17	Raw Accelerometer and Gyro Data with Moving Average Filter	67
3-18	Final Unbiased Acceleration and Angular Velocity Response to a Single Thruster Pulse.	68
3-19	Configurations for the System Identification Analysis on Flight 1 (top left and right), Flight 3 (bottom left), and Flight 4 (bottom right).	69
3-20	Portion of the UDP Space Operation Plan	76

3-21	SPHERES Docking Port Test Session Plans Build Toward Halo Operations	78
4-1	Concentric Circle Fiducial Markers Have Previously Been Used on the Exterior of the ISS	80
4-2	The SPHERES Universal Docking Port (left) and Necessary Docking Tolerances (right)	81
4-3	The Linear and Angular Error Constraints Needed to Achieve Docking. Green Is Within the Hole. Orange Is Within the Capture Cone. Red Is Outside The Capture Cone.	82
4-4	The Selected COTS Camera for Integration into the SPHERES UDP.	84
4-5	The Selected Fiducial Markers for the SPHERES UDP.	85
4-6	The Area Ratios of the Fiducial Markers are Spaced Linearly.	86
4-7	The Result of Performing Adaptive Thresholding on a Grayscale Image and Its Inverse	87
4-8	Four Point Correspondence Geometry of Haralick's Iterative Nonlinear Solution to Exterior Orientation [7].	89
4-9	The Camera Calibration Process Used for the UDP Camera	91
4-10	Uncorrected Mean Reprojection Error Anomalies Due to Rolling Shutter	92
4-11	Corrected Mean Reprojection Error Anomalies of a Rolling Shutter Camera	93
4-12	Optical Setup for Measuring Sensing Error	94
4-13	Absolute Error in the X-Direction from 13.9 cm Range	94
4-14	Absolute Error in the Y-Direction from 13.9 cm Range	95
4-15	Absolute Depth Error in the Z-Direction from 13.9 cm Range	95
4-16	This graphic shows higher measurement sensitivity (lower noise) for an x-axis twist (blue is the image plane).	97
4-17	This graphic shows lower measurement sensitivity (higher noise) for a y- or z-axis tilt (blue is the image plane).	98

4-18	Examples of a False Negative Measurement (left) and a False Positive Measurement (right).	99
4-19	[Animation] The Proposed Concept of Operations for Relative Pose Estimation.	100
4-20	Representative tracking of position states with full noise modeling. . .	112
4-21	Representative tracking of velocity states with full noise modeling. . .	113
4-22	Representative tracking of quaternion states with full noise modeling.	114
4-23	Representative tracking of angular velocity states with full noise modeling.	115
4-24	Representative quaternion error with full noise modeling.	116
4-25	Representative angular velocity error with full noise modeling.	117
4-26	Representative angular velocity error with full noise modeling.	118
4-27	Representative angular velocity error with full noise modeling.	119
4-28	The non-dimensional performance of the MEKF and UKF filters (lower is better)	120
4-29	[Animation] Real-Time Fiducial Tracking (Target Is Locked)	122
4-30	Quaternion estimate of the state in the SPHERES hardware demo . .	122
4-31	Angular rate estimate of the state in the SPHERES hardware demo .	123
4-32	[Animation] Real-Time Fiducial Tracking During Undocking (Target Is Locked)	125
5-1	Two views of the Space Station Remote Manipulator System (ie Canadarm 2) [8]	131
5-2	The Canadian Built Dextre has been Operational Aboard the ISS Since 2008 [9]	132
5-3	The Japanese Experiment Module (JEM) Remote Manipulator System (RMS) [10]	132
5-4	The Robonaut Arms Are Capable of Grasping Tools [11]	133
5-5	The Engineering Test Satellite VII Robotic Arm [12]	134
5-6	The Orbital Express OEDMS Robotic Arm During Final Assembly [13]	134

5-7	The European Robotic Arm (ERA) [14]	135
5-8	The Ranger Dexterous Manipulator [15]	136
5-9	The Dynamic Manipulation Flight Experiment (DYMAFLEX) [16]	137
5-10	The Front-end Robotic Enabling Near-term Demonstration (FRIEND) [17]	138
5-11	An Assembly Overview of the SPHERES Lego Robotic Arm	139
5-12	Evolution of the SPHERES-ARM Over Three Generations of Designs	142
5-13	A Mechanical and Electrical Prototypes of the Second Generation SPHERES-ARM	143
5-14	The Design of the Third Generation SPHERES-ARM Prior to Manu- facturing	145
5-15	The Fin Ray Gripper Can Apply Equal Pressure Gently But Firmly to an Egg [18]	146
5-16	The Proposed Fin Ray Gripper Design for the SPHERES ARM	147
5-17	Two SPHERES-ARMS Mounted on the Halo for Satellite Servicing Activities	148
5-18	Conjoined Circle Method (CCM) for Inverse Kinematics	150
5-19	Conjoined Circle Method for 3 DoF	150
5-20	Conjoined Circle Method for 3 DoF	152
5-21	[Animation] A Visualization of the Inverse Kinematics Solver for A Fixed-Base, 3D Robotic Arm	154
5-22	The Proposed Angle Displacement Method for A Two-Segment Body	155
5-23	[Animation] Dynamic Simulation of a Free-Floating Satellite and Pla- nar Manipulator	156
5-24	[Animation] Dynamic Motion Study of a Robotic Arm and a SPHERES Halo	157
5-25	[Animation] Hardware Demonstration of the Lego Robotic Arm Exe- cuting the Same Open Loop Controls as Figure 5-23.	158
5-26	Results from the Lego Arm Hardware Demonstration Including Actu- ator Commands and Rotational Feedback	159

List of Tables

2.1	Virtual Methods Implemented in a Test Project	35
2.2	Necessary Virtual Methods for a Guest Scientist Payload	35
3.1	Docking Port Classification Matrix	41
3.2	Consolidated Requirements Matrix	50
3.3	Specification Summary of the Flight UDP Compared with the Ground Prototype	58
3.4	Summary of Test Configurations and Accomplishments	64
3.5	Experimental and CAD Predictions of the Assembly Centers of Mass	73
3.6	Experimental and CAD Predictions of the Assembly Inertia Matrices	73
4.1	Consolidated Requirements Matrix for the UDP Sensor	82
4.2	Specifications of the uEye XS Camera	84

Nomenclature

AFS	Aurora Flight Sciences
CoM	Center of Mass
COTS	Commercial Off The Shelf
DARPA	Defense Advanced Research Projects Agency
DOF	Degree of Freedom
ESA	European Space Agency
GSP	Guest Scientist Program
ISS	International Space Station
MIT	Massachusetts Institute of Technology
NASA	National Aeronautics and Space Administration
OOP	Object Oriented Programming
OS	Operating System
PADS	Position and Attitude Determination System
RGA	Reduced Gravity Aircraft
RMS	Root Mean Square
RSFS	Robotic Servicing of Geostationary Satellites
SPHERES	Synchronized Position Hold Engage and Reorient Experimental Satellites
SSL	Space Systems Laboratory
USB	Universal Serial Bus
VERTIGO	Visual Estimation for Relative Tracking and Inspection of Generic Objects

Chapter 1

Introduction

1.1 Motivation for In-Space Satellite Cooperation

Spacecraft rendezvous and docking is becoming increasingly frequent with the growing commercialization of space. By far the most common application of docking is in the delivery of crew members or supplies to an orbiting space station. However, the applications of satellite cooperation extend far beyond the transportation of assets into Low Earth Orbit (LEO). In this section, we show that for both economic and scientific purposes, satellite cooperation makes sense for many applications.

The notion of satellite cooperation suggests varying degrees of synergism between spacecraft on-orbit. The technology developed in this thesis (specifically, the requisite sensors and actuators) relates to both the servicing of known and unknown systems, and the assembly of aggregative spacecraft structures .

Satellite servicing on orbit is an emerging technology enabler that can affect a broad class of missions. Once mature, in-space servicers can provide satellite operators the necessary resources to diagnose anomalies on orbit, correct mechanical and electrical problems, and repeatedly enhance our high-value assets over long periods of time. This paradigm of inspecting, repairing and upgrading vehicles exists everywhere on Earth, but not in space [2]. Consider aircraft, ships and automobiles; all are re-furbished and often improved over their lifetime. There is no question that this capability boosts the return on investment in these machines.

Consider the Hubble Space Telescope (HST), arguably the greatest space-based obser-

vatories ever built. Hubble launched in 1990 but required multiple servicing missions in order to reach its full potential as an effective star-gazer. Figure 1-1 shows graphically the enormous effects that servicing had on the HST mission. The Davidson metric is a measure of NASA contributions to worldwide scientific discovery and technological achievement.

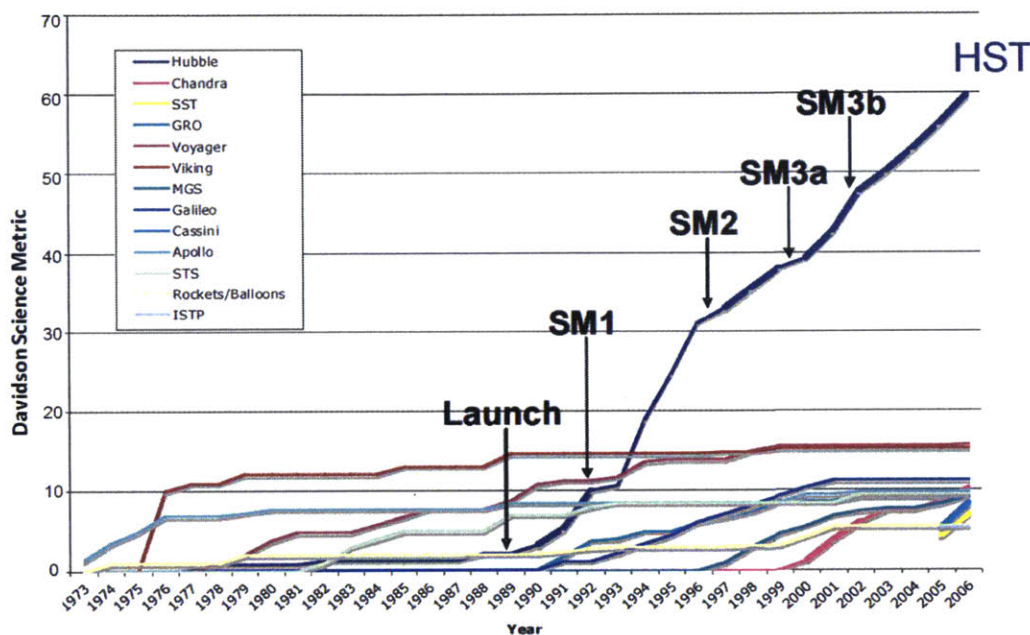


Figure 1-1: The Enormous Scientific Return from Hubble Would Not Have Been Possible Without Post-Launch Servicing Missions [1]

Each HST servicing mission involved high precision rendezvous, delicate component inspection and replacement, and upgraded instrument installation. Once the telescope was redeployed, the shuttle usually re-boosted Hubble's orbit to correct for atmospheric drag and further extend the life of the satellite. From conception, Hubble was designed to be serviced by spacewalking shuttle astronauts, but this is rare. Despite its success, the Hubble servicing scenarios were performed at high cost and risk to human life. Robotic servicers are an alternative methodology promising to provide many of the same (or better) capabilities at lower cost and complexity to human space flight.

Satellite cooperation extends to on-orbit assembly of spacecraft as well. In the same way that organic cells work together to form a larger organism, smaller satellite modules (or satlets) may aggregate together in-space to achieve greater scientific and economic returns. The cellularized approach to satellite construction is a potentially disruptive space

technology promising at least an order of magnitude reduction in cost for equal or greater performance [19]. Such fractionated spacecraft support dynamic reconfigurability on-orbit and allow for spacecraft electronics refreshes more frequently. The economic impacts of this approach can be seen graphically in Figure 1-2. Satellite manufacturers should be taking advantage of the economies of scale through construction of many smaller, nearly-identical satlets. This can reduce the prohibitively large mission costs that come from large monolithic space vehicles.

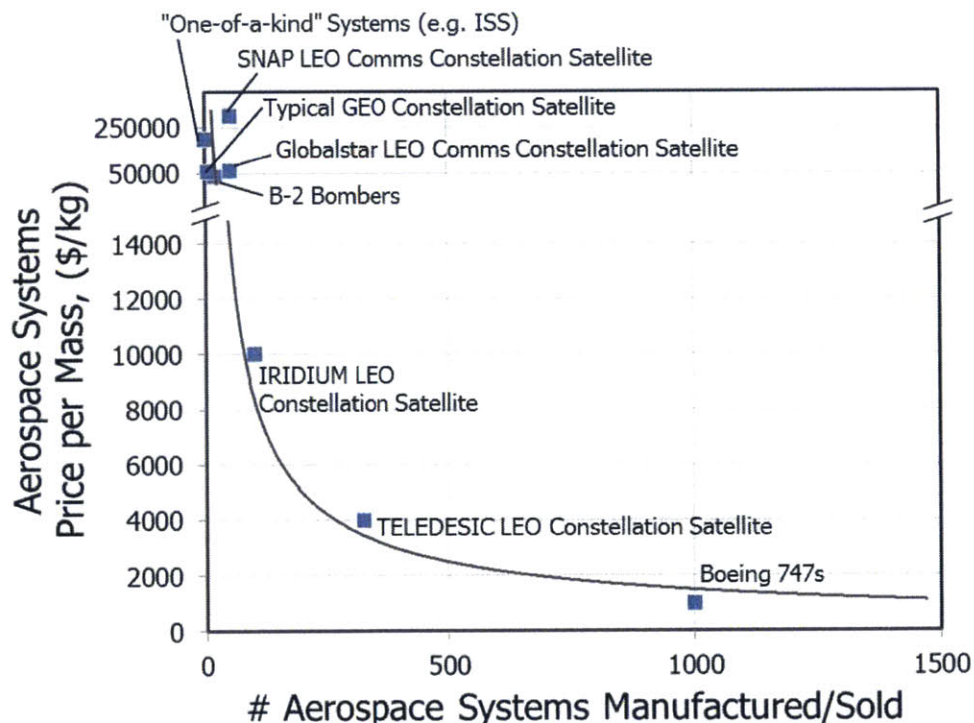


Figure 1-2: There Is A Large Economic Incentive To Mass Producing Aerospace Systems [2]

The robotic capabilities developed in pursuit of satellite cooperation take a variety of forms, and enable advancements in a range of space applications, beyond just fixing satellites. At the core of each of these functions are the sensors and actuators used autonomously by the vehicle (or remotely by operators).

- **On-Orbit Servicing Missions:** With the right set of tools and technologies, satellites can be capable of performing complicated maintenance tasks in orbit. Repurposing damaged or decommissioned spacecraft extends the life of current and fu-

ture satellites by reusing the larger, more massive components and upgrading generic system performance [19]. “Servicing” can encompass a mixture of activities. Opening and closing valves, assisting in the actuation of deployable mechanisms (antennas, solar panels), transferring fluid, swapping in state-of-the-art avionics boxes, recharging exhausted batteries - all are value-added tasks that can open new territories in the satellite domain. Another related concept is the harvesting of parts from retired satellites. A large fraction of the volume to orbit is static spacecraft structures such as antennas or aperture mirrors. A number of special programs have investigated the feasibility of extracting these resources from dead satellites and integrating them into new systems on-orbit.

- **Orbital Debris Removal:** A large amount of debris orbits the earth at most orbital altitudes. This debris slowly limits the safe use of space. The objects are in most cases of unknown mass and shape, and are tumbling at an unknown rate. Their capture, control, and removal is essential for slowing the effects of the Kessler Syndrome. Broken and drifting satellites take up valuable GEO real estate and pose a risk to their space neighbors. Disposable constellations of small satellites in LEO also do not lend themselves to after launch service; instead mission termination simply produces a great deal of pervasive space junk. While satellite servicing can help make space more sustainable moving forward, there is also existing debris that needs to be classified, inspected and expelled through some strategy involving mechanical docking.
- **Orbital Tug:** One of the most common termination conditions for spacecraft in Low Earth Orbit is orbit decay due to atmospheric drag. Other functions such as station-keeping also burn fuel. If the satellite cannot be re-fueled it would be advantageous for a servicer to rendezvous, dock, and externally provide delta-v to extend the life of the high-value asset. In other cases, the satellite may need to be salvaged from a stranded orbit due to an upper stage failure, or relocate to a new orbit for science purposes. Satellite cooperation may also be needed if the target satellite has been retired but is unable to transition to its graveyard orbit.
- **Robotic Satellite Assembly:** Assemblies of fractionated spacecraft are resilient space systems with an increased flexibility to respond to changing mission objectives and potential threats. The DARPA System F6 program aimed to demonstrate on-

orbit resource sharing and autonomous reconfiguration of networked modules. Most importantly, when spacecraft are designed to be divided into separate modules, failed modules can be quickly replaced without losing the mothercraft [20]. Large assembled satellites systems can also accomplish much greater goals than any one single spacecraft. Consider the International Space Station, which was assembled in-situ from a series of smaller modules.

- **Habitat Resupply:** Humans in space require frequent replenishment in the form of water, oxygen and organics, even in highly efficient nearly-closed systems like the ISS. Re-supply of consumables naturally demands the satellites mechanically dock and transfer cargo between vehicles.
- **Asteroid Sampling:** The knowledge of the composition of near-Earth asteroids and comets allows for unique opportunities to understand the history of the solar system and how to deal with any threats to the planet. However, to obtain those samples, it is necessary to land, soft dock, or grapple tumbling, nutating, and possibly venting structures. This involves tackling ever increasingly difficult dynamics problems, and identifying and controlling suitable docking points.
- **Sparse Apertures:** Precision formation flight and precision metrology are key technologies to enable future long baseline, space-based imaging interferometers. The driving technology innovation for sparse aperture telescopes is the autonomous control of aggregative satellite systems, cooperating in unison.

In each one of these applications, the spacecraft need a unique suite of sensors and actuators to interact with each other and achieve the mission objectives. This thesis focuses on testbed development for advancing the relevant technologies in a risk-reduced, dynamically authentic, rapidly iterable environment.

1.2 Existing and Planned Testbeds and Technical Demonstrations

The value of testbed technology development cannot be understated. Space presents a unique assortment of challenges such as a harsh and remote environment. Access to space

is difficult (but getting cheaper) so it is prudent to invest a fraction of time and resources to (1) rapidly iterate on hardware, software and algorithms in a cheap, easily accessible manner and (2) increase confidence in the performance of the space system prior to launch.

Typically, the technology can be accelerated through its lifecycle through repeated use in a realistic testbed, or through a scaled technology demonstration on-orbit. With any new technology, but especially in the space arena, there exists a variety of technological and logistical barriers to entry. Independent from such barriers, there also exists a varying degree of risk which is a function of the technology readiness of the hardware or algorithm.

Spacecraft docking and servicing has been studied and tested in various forms for decades. Recently there has been a resurgence in interest in the field (due in part to the introduction of the small satellite). As recently at 2015, NASA's Robotic Refueling Mission (RRM) has demonstrated and tested the tools and techniques needed to robotically refuel and repair satellites in-space, especially satellites that were not originally designed to be serviced [21]. One of the most valuable outputs from a demonstration campaign is the validation of ground based simulations used in the modeling and development during the incubation period of the technology.

In this section, we present three relevant testbeds or technical demonstrations that will principally drive the sensors and actuators developed in this thesis.

1.2.1 SPHERES

The Synchronized Position Hold Engage and Reorient Experimental Satellites (SPHERES) testbed is long duration zero gravity platform for advancing high-risk space technologies. The SPHERES were designed and built by the MIT Space Systems Laboratory with support from NASA, DARPA, and Aurora Flight Sciences. In 2006, three units launched to the International Space Station (ISS) and have since conducted over 70 test sessions of space research. Figure 1-3 shows the three satellites operational on-orbit.



Figure 1-3: SPHERES Aboard the ISS [3]

SPHERES is a one-of-a-kind test facility providing a risk-tolerant environment within the confines of the ISS. SPHERES initially was used to study sensor, control and autonomy algorithms for use in satellites, especially in the area of formation flight. Since the satellites are operated by the astronauts, fuel tanks can be replaced and battery packs can be swapped within minutes. If the satellites spin out of control or collide with each other, the astronauts are standing by to regain control and reset the test. This allows the scientists to push the boundaries of traditional control and autonomy algorithms.

SPHERES is a realistic spacecraft testing facility for exercising the full 6 DoF dynamics of close-proximity, multi-satellite scenarios. SPHERES propulsion is provided by 12 cold-gas thrusters and a single propellant tank storing compressed CO₂. Each 4 kg SPHERES is capable of providing 15 m/s of delta-V per tank [22]. SPHERES also employs a suite of sensors directly analogous to the sensors used on an operational space system. A pseudo-GPS strategy based on ultra-sonics provides 1 cm ranging and less than three degrees bearing angle measurements relative to the global ISS frame [23]. With these raw characteristics in mind, it is clear that SPHERES provides a very desirable testing platform that cannot be

replicated on the ground or any sufficiently high fidelity computer simulation.

The SPHERES also poses an expansion port which allows for additional technology payloads to interface with the SPHERES and operate in a zero gravity environment. In the past, these payloads have included electromagnetic RINGS for formation flight, stereo vision cameras for vision-based navigation, and a fluid tank for recording slosh behavior in microgravity.

This thesis proposes additional sensors and actuators that can be operated by the SPHERES in order to advance the technology readiness level of the highest risk elements involved in satellite docking and servicing.

1.2.2 DARPA Phoenix

DARPA Phoenix is a technology demonstration aimed at developing inspection and servicing capabilities for assets at GEO and validating new satellite assembly architectures [24]. If successful, Phoenix will change the traditional design and operational life cycle of GEO birds into one that reduces mission size, complexity, and ultimately cost. The elements of DARPA Phoenix are shown in Figure 1-4.

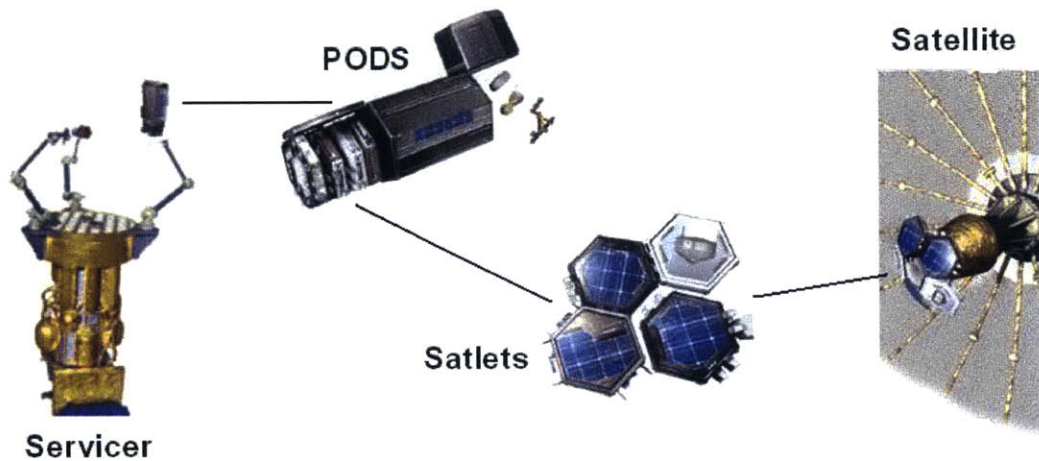


Figure 1-4: The Principle Systems Involved in the DARPA Phoenix Project [4]

The Phoenix mission architecture begins with mini-satellites known as satlets. These independent, modular satellites are designed to share data, power and consumables. The technology innovation they provide is the ability to scale “almost infinitely” [4] both in

production and operation. The satlets are transported into orbit on a Payload Orbital Delivery (POD) system (Figure 1-4). The POD is a packaging structure designed to efficiently carry a variety of mass elements, including satlets, to orbit [24]. Once in orbit, the satlets are gathered and ferried to the target GEO satellite on the servicer satellite's toolbelt. Upon arrival, the satlets are released and assembled into an operating spacecraft, leveraging harvested parts from the retired GEO bird where possible.

In order for Phoenix to be successful, certain key technologies must be demonstrated. Docking ports, robotic manipulators, and relative sensors are all on the critical path to mission success. It is reasonable for DARPA to explore a range of potential designs and algorithms. SPHERES can provide a unique platform for testing this type of experimental technology at low cost and low risk. This thesis explores research that is intended to be directly traceable to DARPA Phoenix. In turn, DARPA Phoenix will enable a broad class of robotic satellite missions and an unprecedented transformation in the satellite business.

1.2.3 RSGS

The Robotic Servicing of Geostationary Satellites (RSGS) is a DARPA mission leveraging much of the research already invested in the Phoenix project. RSGS de-scopes the Phoenix objectives by focusing strictly on the development of a commercial satellite servicer. This servicer will be equipped to assist with mechanical malfunctions on orbit, such as solar array deployment, and provide assistive thrust to extend lifetimes or re-organize constellations [25]. In addition this servicer may employ an optical or thermal camera system to inspect assets in orbit that may be suffering from operational anomalies. RSGS is still in the preliminary Mission Definition phase.

DARPA Goals for GEO Robotics Servicing

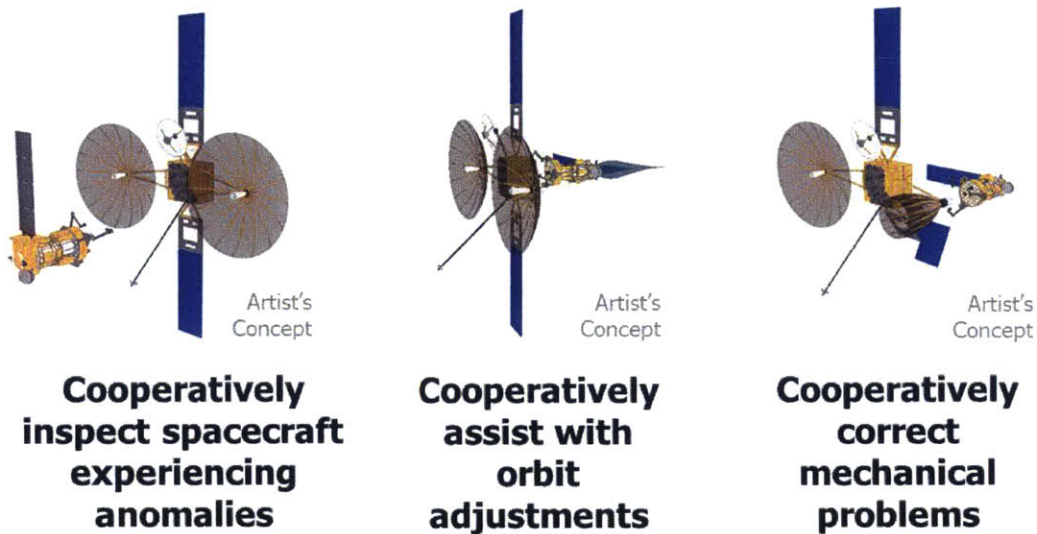


Figure 1-5: An Overview of the Robotic Servicing of Geostationary Satellites (RSGS) DARPA Mission

1.3 Thesis Approach

A robotic servicing mission typically involves four high-risk tasks: (1) rendezvous in orbit, (2) close proximity operations, (3) servicer-target berthing or docking, and (4) target manipulation and re-purposing (Figure 1-6). The software and hardware required for these operations have few precedents in space, and as such the SPHERES-ISS facility provides a long-duration microgravity testbed to develop such technologies in a low-risk environment that affords risk reduction capabilities.

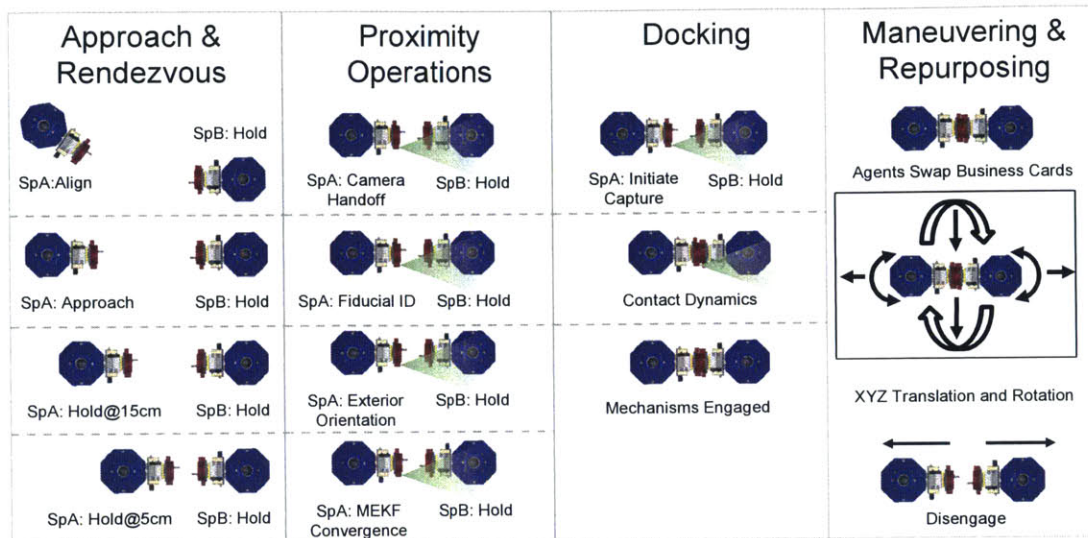


Figure 1-6: Phases of Servicing

This thesis focuses on maturing the sensors and actuators that are active during the proximity operations and capture phases. The six chapters herein present the motivation and framework for three different sensors and actuators that can be exercised in a range of satellite inspection and servicing scenarios. Since these payloads are designed for an existing satellite testbed, the components are subject to the resource constraints of SPHERES, which has been shown to be a realistic analog to authentic spaceflight systems.

Chapter 1 has explored the merits of in-space docking and servicing of complex spacecraft systems. Three state-of-the-art missions have been introduced which will steer the development of many critical technology payloads necessary for effectively cooperative satellites. **Chapter 2** presents a peripheral agnostic software architecture upon which the subsequent payloads will be operated. This platform was initially developed for the docking and servicing research presented herein, but has substantial applications in the broader field of space robotics. **Chapter 3** introduces the first actuator and presents a collection of docking port classifications and designs. Then, the flight revision of the SPHERES docking port is described in detail. The testing performed in a relevant dynamic environment will show a high confidence in the articles's on-orbit capabilities. **Chapter 4** introduces a high precision relative sensor that can be used during proximity operations on the SPHERES. Moreover, a nonlinear estimator is derived and shown to provide sufficiently accurate sensing solutions in highly dynamic docking scenarios. **Chapter 5** presents a prototype actuator in the form

of a free-flying robotic manipulator. Next, the coupled dynamics and controls problem for this satellite appendage is defined and subsequently solved in simulation. **Chapter 6** concludes the thesis by summarizing the results and evaluating the contributions to the academic community.

Chapter 2

Framework: Peripheral Agnostic Software Architecture for a Satellite Servicing Toolbelt

2.1 Overview

This chapter details a peripheral agnostic software architecture that will be applied to the sensors and actuators presented in Chapters 3, 4, and 5. This platform supports a range of peripherals that advance satellite technologies such as inspection, docking, servicing and control. The proposed software has been designed from the beginning to enable object abstraction by collecting implementation details behind a layered interface. Guest scientists can thus focus on technology innovation more than tedious low-level details. In the end, the framework provides a suite of libraries and tools that can be utilized in a dynamically authentic space environment in order to test satellite servicing technologies.

First, the proto-flight toolbelt, known as the SPHERES Halo, is described. This hardware constrains the software architecture to a specific operating system and associated driver capabilities. Section 2.3 presents the system requirements and capabilities of the designed software architecture, known as HaloCore. Section 2.4 introduces the Halo Guest Scientist Program as a platform for present and future researchers to test emerging space technologies in a risk-tolerant environment. Finally, the HaloCore system is compared to

other open-source and commercial robotics platforms in Section 2.5 for perspective.

2.2 Hardware Platform

There are six common conditions that usually characterize a space environment: radiation, thermal, vacuum, orbital dynamics, field of regard, and microgravity [5]. The first three areas can be tested on the ground with relatively high fidelity, and orbital dynamics and field of regard (ie lighting) are fairly low risk. The subject of interest, microgravity, can be exercised on the SPHERES testbed on the ISS. This makes the SPHERES and the associated facilities a one-of-a-kind platform for testing high risk space technologies in a dynamically authentic microgravity environment.

The SPHERES Halo extends the range of possible sensors and actuators that can be operated in this testbed by supplying high speed data processing and increased data storage. The SPHERES Halo is six-port expansion to the SPHERES VERTIGO test facility. As shown in Figure 2-1, the Halo exoskeleton allows scientists to mount and control multiple peripherals within a single facility. The design is modular so that scientists can test as much or as little as desired during a single test. Such a setup is highly conducive to advancing servicing technologies as it allows researchers to rapidly iterate on both payloads and software in a representative testing environment.

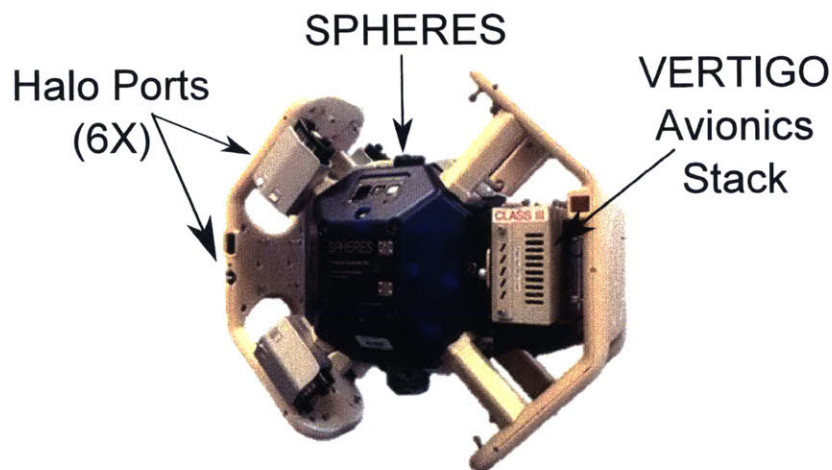


Figure 2-1: Halo Prototype Hardware that Supports Peripherals On Up To Six Halo Ports

All peripherals that interface through the Halo toolbelt are provided with a collection

of power and communication lines for operation. This includes

- 2X USB 2.0
- Gigabit Ethernet
- Regulated 5V power at 1 A
- Unregulated 12V power at 1.5 A

The Halo flight computer resides within the VERTIGO Avionics Stack (Figure 2-1), which runs a Via Pico-ITX P830 processor. The processor includes 4GB of RAM and two 64GB flash drives running a Linux Ubuntu 10.04 LTS environment. Such a system enables researchers to explore most areas of robotic servicing and assembly, including reconfigurable control, high-precision pointing, inspection and docking. Figure 2-2 shows the variety of payloads already compatible with the SPHERES Halo.

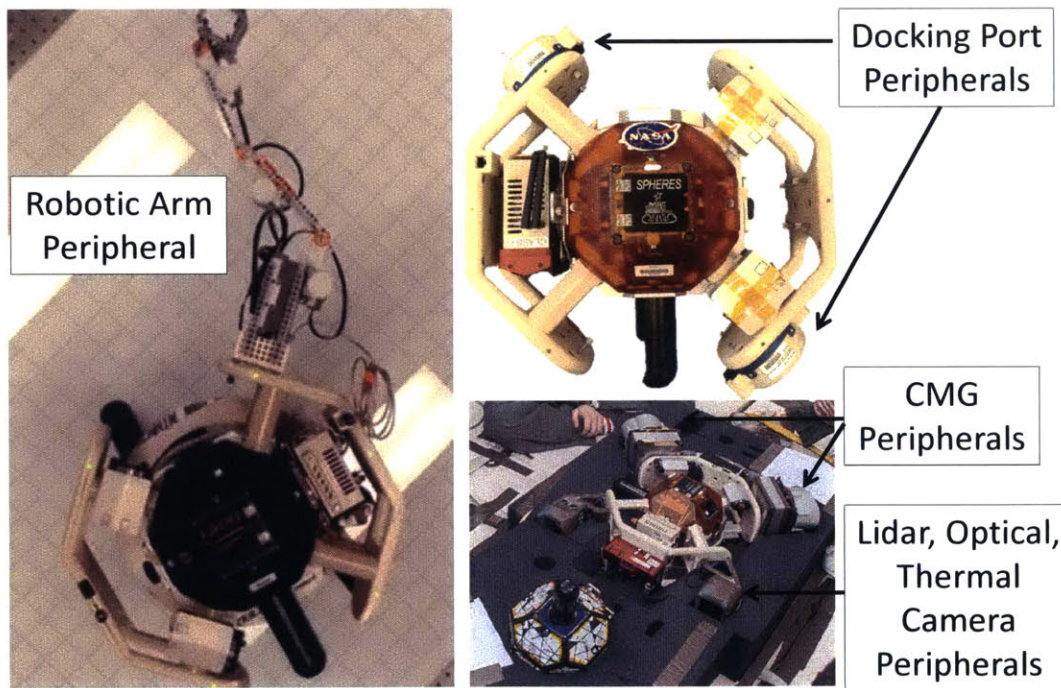


Figure 2-2: Currently Compatible Halo Peripherals Include Docking Ports, Stereo Vision Cameras, Control Moment Gyro's, Thermo-Imagers, Lidars, and Robotic Manipulators

2.3 Software Architecture

This section details the philosophy and structure of the HaloCore software platform so that it may be successfully integrated with any code governing the operation of new peripherals. During development, at least seven ideologies drove the design and evolution of the software architecture. Collectively, this software platform, implemented on the VERTIGO Avionics Stack, extends the capabilities of the SPHERES Halo.

- **Object Oriented Programming:** As shown in Figure 2-3, the HaloCore framework was designed to be as distributed and modular as possible. Since HaloCore is written in C++, the framework improves on the procedural C coding of the SPHERES Guest Scientist Program by abstracting much of the low-level implementation details into object classes. In addition, multiple instances of the same peripheral type (such as docking ports) are easily supported.
- **Multi-threading:** Threading is critical to the process management of the HaloCore system. Background tasks, communication and telemetry are able to run efficiently and independently of the Guest Scientist's research. Moreover, multi-threading allows peripherals to operate at different control cycles and estimation rates which may be critical to meeting science objectives. Figure 2-6 shows a representative threading scenario run through HaloCore.
- **Thread Prioritization:** Threading introduces additional complexities such as shared memory and resources. Thread prioritization is employed so that computing resources are fairly allocated between parallel processes in order to achieve the science objectives.
- **SPHERES Interface:** The HaloCore software architecture has been designed to interface with the existing SPHERES facility so that the propulsion and metrology systems of the SPHERES may be leveraged with the operation of any new peripherals.
- **Robust Data Storage:** Data collection and storage is critical for a testbed like SPHERES because test data is ultimately the science and purpose of the testbed.
- **Adaptability:** The HaloCore framework supports a wide range of present and future peripherals and has been formulated from the ground up to be adaptable to future

mission objectives.

- **Open Source Development Model:** The open source software model is intended to generate an increasingly diverse scope of software capabilities for HaloCore. Like the SPHERES Guest Scientist Program, the Halo software is expected to be released online to potential developers to encourage collaboration and research in the satellite servicing field.

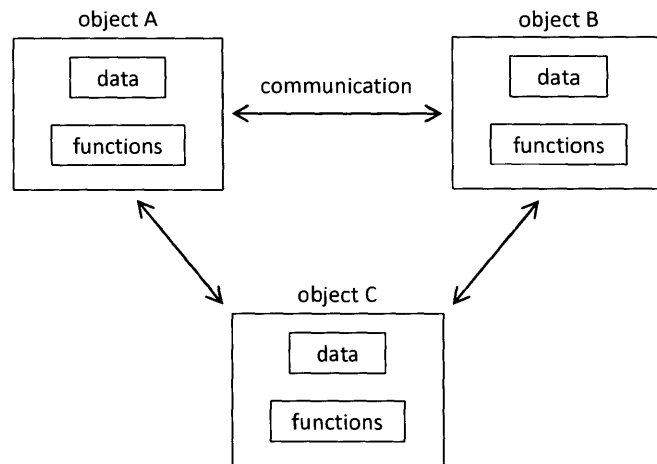


Figure 2-3: An Object Oriented Approach to Flight Software Supports Multiple Peripherals Simultaneously

The purpose of the PeripheralCore is to house all necessary functions and variables that are specific to the behavior of the peripheral. Peripherals are anything that attach to and interface with the Halo, such as docking ports and robotic arms. Figure 2-4 shows the diversity of peripherals currently supported through HaloCore. As shown, each payload class inherits from the parent HaloPeripheral class and can be expanded on from there. Each PeripheralCore contains the basic operational code for the payload plus any external science libraries that may be necessary to accomplish the desired science. Other supporting files may be included, such as a class for cameras connected to the peripheral, or a class to manage data storage. These supporting classes provide better organization by grouping related methods (such as those relating to storing results) in a single class. This added abstraction layer makes instantiation of a single (or multiple) peripheral easy and increases the versatility of the presented satellite servicing toolbelt.

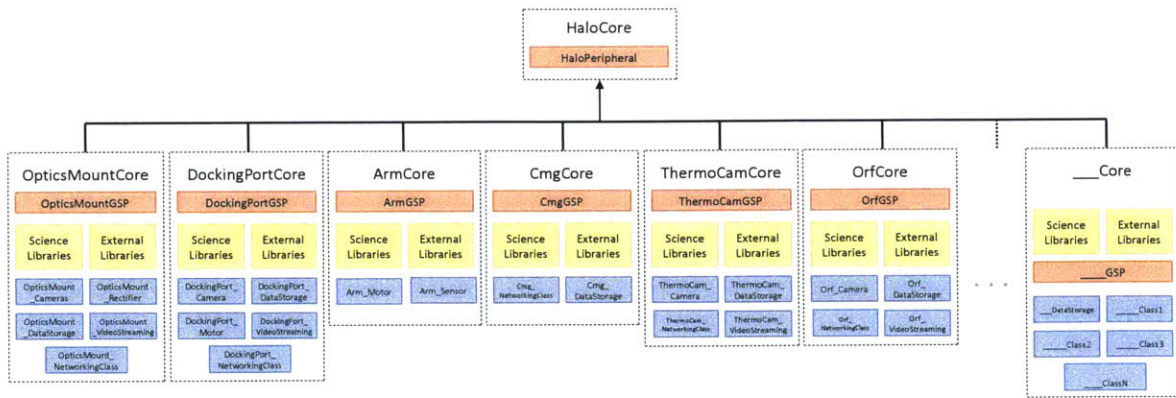


Figure 2-4: Current Peripheral Cores Supported By The Halo Software Framework.

Figure 2-5 documents the flow of software beginning with the `main()` function that gets called during the C++ code execution. The key take away is that the test project specific code can be written without duplicating background initializations for each new project. Instead, the core Halo code is run automatically and the Guest Scientist's code (denoted with the 'GS' prefix) can be appropriately executed from within the test project source code. The functions in blue are intended to be modifiable by the Guest Scientist based on the intended operational research (refer to Table 2.1).

Flow of Halo Linux Software

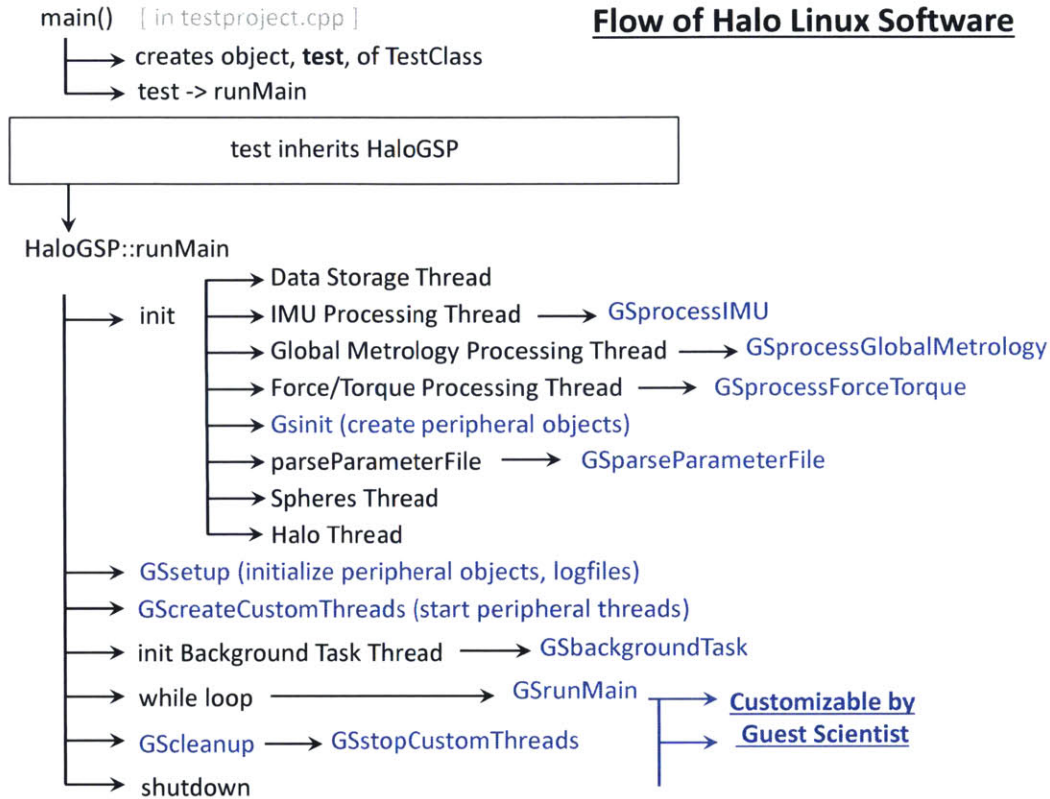


Figure 2-5: Halo Software Flow Diagram (adapted from [5])

There are several POSIX threads, or pthreads, that always run in parallel during the course of the test, in addition to any peripheral specific threading that may be initialized. These threads support data storage, SPHERES and Halo communication among other tasks. Representative thread execution is shown graphically in Figure 2-6. Threading also introduces a number of complexities related to thread prioritization and shared memory access. Mutexes are used in conjunction with threads to provide greater security and prevent different threads from interfering and perhaps attempting to access the same memory at the same time. The mutexes can be locked and unlocked, and a thread will not continue its process if a mutex is locked and currently belongs to a different thread.

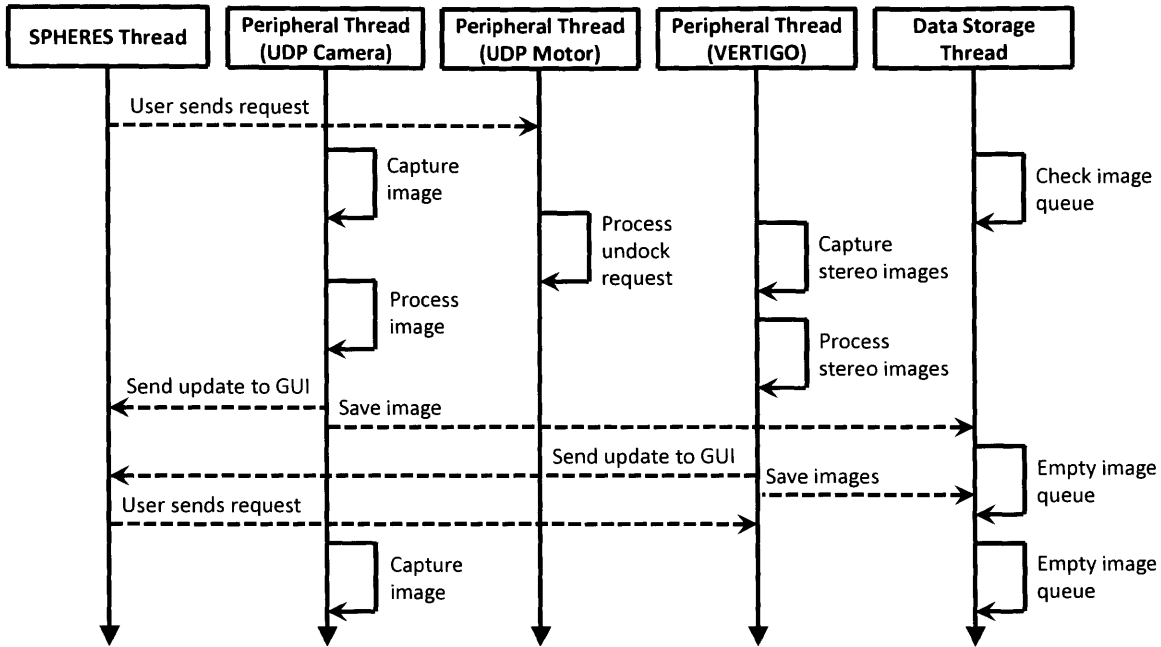


Figure 2-6: Representative Thread Execution During a HaloCore Test Project

2.4 Halo Guest Scientist Program

This section introduces the Halo Guest Scientist Program (GSP) for investigators interested in collaborative research in the field of satellite inspection, assembly and servicing. The Halo GSP extends the capabilities of the existing SPHERES testbed and provides a representative microgravity environment for validating high risk space technologies. This section presents the interfaces to the existing Halo software and provides guest scientists with a framework in which to implement novel algorithms. Two elements of the Halo GSP are detailed herein - first, the test project specific methods that govern the payload activity and second, the PeripheralCore methods that should be written for any new payloads intended for the Halo.

2.4.1 Test Project Methods

The HaloGSP class allows guest scientists to customize initialization processes, background tasks, parameter parsing, and threads. It also provides methods that the test project accesses, since the testproject class inherits the HaloGSP class. These methods, described in Table 2.1, contain the bulk of the code that dictates what happens during the duration

of the test. These are virtual methods that are declared in HaloGSP and then overwritten and defined as needed in the test projects, allowing the guest scientist to customize their test project.

Table 2.1: Virtual Methods Implemented in a Test Project

Function	Description
<code>GSinit()</code>	Initialization method typically used to instantiate object pointers and initialize variables
<code>GSsetup()</code>	Runs the initializing methods for peripheral objects and initializes variables
<code>GSrunMain()</code>	Contains body of the test project outside of any custom threads for each peripheral
<code>GScustomThreads()</code>	Kicks off parallel threads for each peripheral
<code>GSbackgroundTask()</code>	Used for methods that are to be run in the background, or are secondary to the main test
<code>GSparseParameterFile</code>	Parses the parameter file for each peripheral object declared
<code>GScleanup()</code>	Calls shutdown functions for the peripherals and destructors for the peripheral pointers
<code>GSparseCommandLineArgs</code>	Parses additional commandline arguments as it relates to the test project

2.4.2 PeripheralCore Methods

The purpose of the PeripheralCore is to house all necessary functions and variables that are specific to the behavior of the sensor or actuator. Peripherals are anything that attach to and interface with the Halo, such as docking ports and robotic arms. PeripheralCore classes and their methods will be accessed through both calls in the test project code and possibly calls in the HaloGSP code. Here are the essential methods that must be defined in each peripheralGSP class.

Table 2.2: Necessary Virtual Methods for a Guest Scientist Payload

Function	Description
----------	-------------

Continued on next page

Continued from previous page

<code>peripheralGSP(HaloGSP * halo, ...)</code>	Constructor method for the <code>peripheralGSP</code> class. The <code>testproject</code> pointer is passed and used to set shared variables. Peripheral ID may also be passed.
<code>init()</code>	Initialization method called within the <code>Gssetp()</code> method within the <code>testproject</code> . Initializes any cameras or processes that should be running; sets certain variable initial conditions.
<code>parseParameterFile()</code>	Custom parameter parser which reads the parameter file input line by line to set peripheral-specific parameters, prior to the actual initialization of the peripheral. Called within the <code>testprojects GSparsedParameterFile()</code> method.
<code>startPThreads()</code>	Thread initializer called within the <code>GScustomThreads()</code> method. These threads perform the peripheral science, such as processing images or actuating mechanisms.
<code>sleepPThreads()</code>	Thread sleeper method. Pauses the threads when the peripheral is not in use to conserve processing resources
<code>wakePThreads()</code>	Thread waker method. Restarts threads when the peripheral is ready to be used.
<code>getSOHBytes()</code>	State of health method that returns the state information to be passed to the SPHERES.
<code>shutdown()</code>	Pre-destructor method end threads, shut down cameras or processes, ends data storage. This is called at the end of the test, in the <code>GScleanup()</code> method defined in <code>testproject.cpp</code> .
<code>~ peripheralGSP()</code>	Destructor

2.5 Comparison to Existing Robotic Frameworks

It is natural to ask what motivates the creation of the HaloCore software framework when there is no shortage of existing robotics software packages available from the greater robotics community. A variety of platforms were evaluated, chief among them being the Robotic

Operating System (ROS), but ultimately the SPHERES team selected the framework presented in this chapter for three key reasons.

First, the Halo software architecture needs to fall under the open source software license in order to best support remote investigators testing novel space technologies. Open source software benefits from the continued development by multiple users. It also means that HaloCore is expandable and can support any number of new or existing peripherals. For every robotics platform that is open source (such as the Robotic Operating System (ROS), Player and Gazebo) there is a robotics platform that is not (such as VEX or LegoNXT). It was clear from the beginning that commercial packages would not be agreeable in this application.

Second, the Halo software architecture needs to be traceable to future flight systems. SPHERES is a testbed for emerging satellite technologies and algorithms. In order for the testbed to collect valid data, both the hardware and the software must be traceable and scalable to real space applications. Traditional space software teams have full control over their flight operating systems. Some existing robotics platforms may not be stable enough to support the complexities and nuances of spaceflight. It is natural to consider a custom solution, without risking compromising science.

Finally, the Halo software framework needs to be lightweight, clean and simple. Many existing robotics platforms, such as ROS, have a range of capabilities that the Halo software simply does not need. This may add unwanted overhead. Moreover, modern robotics platforms are not nearly as constrained by processing and uplink program size as the SPHERES testbed. The HaloCore solution has been trimmed down to the fundamentals, making it an agreeable solution for this application.

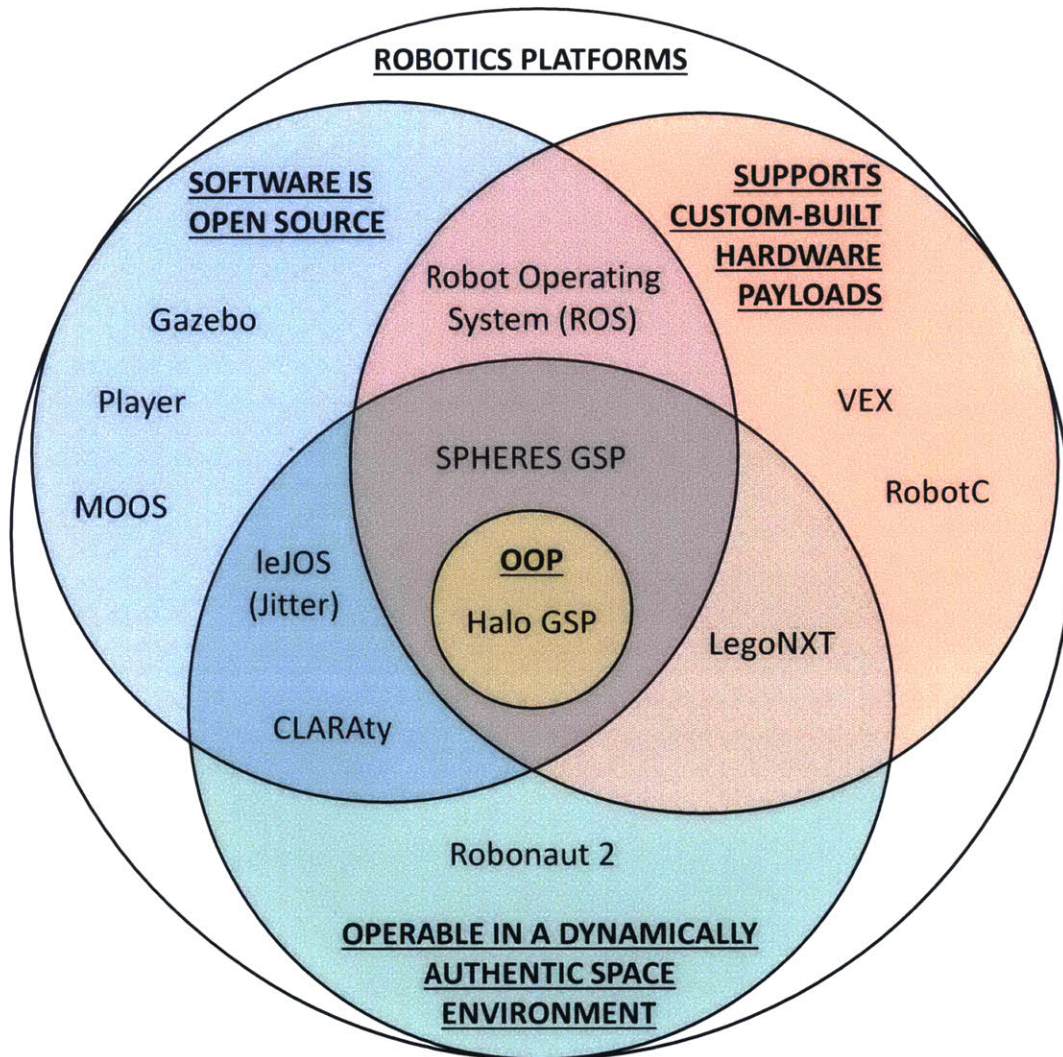


Figure 2-7: An Overview of Where the Halo GSP Platform Fits in the Larger Landscape of Robotics Platforms

Figure 2-7 shows how the Halo GSP system fits into the landscape of robotics platforms. Some robots have already been used in space - Lego Minstorms and Robonaut 2 have flight heritage on the ISS for example. Although the diagram is not comprehensive, it is still clear that the Halo GSP fills a niche of its own as a peripheral agnostic, object-oriented (OOP), software architecture for satellite servicing applications.

Chapter 3

Actuator: Flight Hardware Design of a Rigid Androgynous Docking Port

3.1 Overview

This chapter describes the flight hardware design and validation of a rigid androgynous docking port for the Synchronized Position Hold and Reorient Experimental Satellites (SPHERES) facility aboard the International Space Station (ISS). The addition of Universal Docking Ports (UDPs) to SPHERES is a critical upgrade that provides the satellites with the ability to dock and undock in six Degrees-of-Freedom (DoF). This extension establishes the world's first reconfigurable, on-orbit satellite testbed to address many of the challenges of satellite fractionation. These challenges include performing relative sensing and characterization for docking, adjusting to the new system dynamics of the docked vehicles, and reconfiguring command and control of the aggregated system. In the near-term, the addition of UDPs will help enable the DARPA Robotic Servicing of Geosynchronous Satellites (RSGS) mission, and in the longer-term they will provide an important capability for future studies of reconfigurable spacecraft for new mission architectures involving in-space robotic servicing and assembly.

This chapter begins by reviewing the design space of possible satellite docking architec-

tures (Section 3.2). Specifically, we review the five classifications that traditional satellite docking ports fall under. Then we describe a variety of attributes that may be incorporated into a general docking port design. Section 3.3 presents the first complete iteration of a prototype docking port for use on the SPHERES flat floor test facility. In order to upgrade the prototype technology for space operations, a variety of new design drivers and requirements were evaluated (Section 3.4) and eventually converged upon (Section 3.5). The flight Universal Docking Ports were first tested in three DoF (Section 3.6) and six DoF (Section 3.7), buying down the risk for the upcoming ISS operations.



Figure 3-1: The Final Flight Design of the SPHERES Universal Docking Port

3.2 Spacecraft Docking Port Design Space

The idea of docking ports for spacecraft is not new. Since the 1960s, dozens of satellite docking ports have been successfully designed, built, and operated in-orbit. At the most fundamental level, the objective of a satellite docking port is to provide a mechanism for rigidly docking and holding two spacecraft together. These agents may be moving relative to one another initially, so the docking mechanism must be able to maintain capture despite forces and torques acting on and across the docking interface.

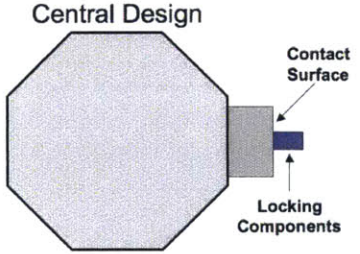
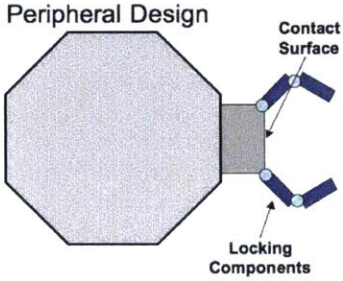
Capture can be achieved through a variety of means, and most designs will be specific

to the mission. However, all docking ports do share common characteristics that can be useful during the system requirements review phase of development.

3.2.1 Docking Port Classifications

In this subsection, we review standardized docking port architectures as a function of mechanism design and geometry. The list of classifications presented is intended to be comprehensive, and is based on both legacy and proposed designs. Table 3.1 provides one example of each class to illustrate the concept, although other instantiations exist.

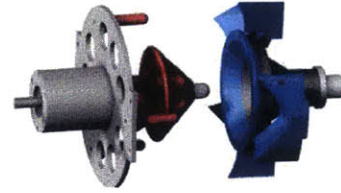
Table 3.1: Docking Port Classification Matrix

Classification	Description	Instantiation
Central	The mechanism of a centrally designed docking port is axially aligned near the origin of the docking face. Typically, a central design has reduced volume as compared to the peripheral design.	 <p>Central Design</p>
Peripheral	A peripherally designed docking mechanism will have locking components radially distant from the docking axis. This can boost docking rigidity and provide volume for capabilities like mass or fluid transfer.	 <p>Peripheral Design</p>

Simplified Mechanism Designs.
 Courtesy of Lennon Rogers [26]

Continued on next page

Gendered
A gendered docking architecture suggests “male” and “female” components on different ports. Specifically, the agents can only dock with modules of the opposite gender.



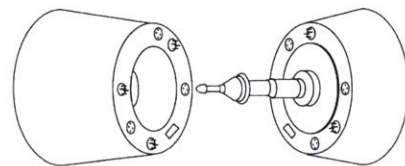
The Autonomous Satellite Docking System (ASDS). Courtesy of Michigan Aerospace

Androgynous
An androgynous docking port can mate with any other similar docking interface, i.e. it is a universal design. A semi-androgynous design is one that can change its configuration depending on the target interface.



An Example Semi-Androgynous Design (Adapted from Oliveri [6])

Symmetric
A radially symmetric docking port design has a degree of freedom during the docking sequence. Specifically, the targets can be at any relative roll angle and still achieve a successful dock. In principle, this could be advantageous during a spin-stabilized docking maneuver.



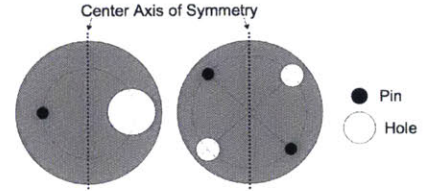
The Soyuz 7K-OK Docking Design. Courtesy of NASA

Continued on next page

Continued from previous page

Asymmetric

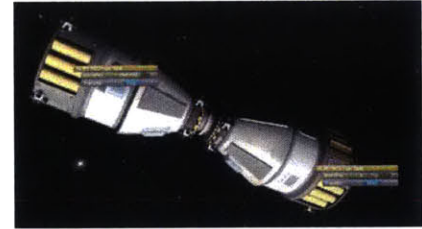
An asymmetric (or inverse-symmetric) design requires that the docking spacecraft be twist-aligned during the docking sequence.



Two Different Inverse-Symmetric Docking Designs. Courtesy of Lennon Rogers [26]

Rigid

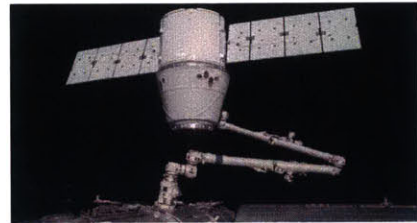
A rigid docking port establishes minimal flexibility between spacecraft, and the docking configuration does not change during the duration of the capture. This can be advantageous when the spacecraft need to maintain a constant relative pose throughout the mate.



Courtesy of Kerbal Space Program

Reconfigurable

A reconfigurable docking port may adjust the relative positions of the agents after contact is initiated. This can be achieved by a robotic arm or some other type of adapting interface. This can be advantageous to achieve safe berthing or pointing.

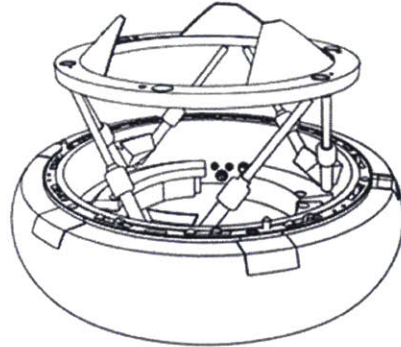


The Dragon Capsule is Relocated After Capture by the Canada Arm. Courtesy of NASA.

Continued on next page

Reusable

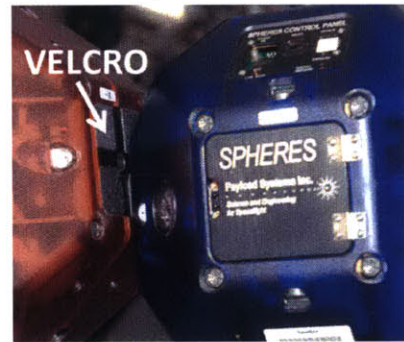
A reusable docking port can be reset after undocking and used again with the same (or another) spacecraft. This could be advantageous for a space station or satellite servicer.



Space Shuttle Docking Port (APAS-95). Courtesy of NASA

Single Use

A single use docking port can only be engaged once during flight (undocking might also not be achievable). This architecture may be desirable due to its simplicity and high probability of success. A single use docking port may employ crushables or pyrotechnics.



The Original SPHERES Used VELCRO for Docking.

3.2.2 Docking Port Attributes

Beyond the generalized classifications, a docking port may also incorporate features that increase the utility of the actuator. This section enumerates capabilities that may be included in the docking port design.

- **Electrical Power Transfer:** When docked, the spacecraft have the ability to share electrical power. This capability could be useful in resurrecting a dead or depleted satellite. The satellites can also optimize peak power point tracking based on solar panel pointing constraints.
- **Data Transfer and Communication:** Hard electrical data lines support high

speed data transfer and control between satellites during the servicing phase. This capability is more power efficient than equivalent wireless transmitters and receivers.

- **Mass and Fluid Transfer:** This allows for the spacecraft to transport fuel, hydraulic fluid, or cargo (including humans). Typically, this necessitates an airlock or pressure equalizer.
- **Thermal Load Transfer:** A thermal conductivity channel between spacecraft facilitates either passive heat sink shunting or a combined active thermal control system. This can be a critical feature considering the updated shading and radiating areas between spacecraft.
- **Electromagnetic Attraction and Repulsion:** Electromagnetics can aid in the docking and undocking sequences by eliminating bounce-back and reducing fuel consumption during the contact phase. During docking, the electromagnets can act as brakes and reduce the contact dynamics, increasing the probability of a successful dock. During undocking, a repulsive force can push the spacecraft apart without using fuel. This can be particularly useful because there is no risk of plume impingement on the target spacecraft during proximity operations.
- **Variable Dampening and Stiffness:** Having control of the rigidity of the docking interface, such as through piezoelectric elements, is useful for vibrationally sensitive payloads, especially those involving optics.
- **Soft Capture Capability:** If docking is to be performed at high relative velocities, it is reasonable to include a certain amount of cushion to reduce the mechanical impulse during capture. Eliminating high mechanical stresses during docking can be a critical capability, especially to protect delicate spacecraft components, such as the solar cells.
- **High Precision Relative Sensing:** Traditional orbit determination on satellites is performed using a global metrology system such as GPS, magnetometers, star trackers or ground based telescopes. Since docking sequences are performed with tolerances that are an order of magnitude below global sensing capabilities, a relative sensor such as a camera or Lidar should be used.

- **Large Docking Tolerances:** If a relative sensor is not used, the docking mechanism should support large docking tolerances considering the agents will have large uncertainties in their relative positions and orientations.

3.3 Ground Prototype Design

The previously presented docking port classifications and attributes introduce an enormous design space for satellite docking mechanisms. One instantiation was realized in 2005 at MIT as part of the Self-assembly Wireless Autonomous and Reconfigurable Modules (SWARM) program. A satellite docking system needed to be developed to advance modular spacecraft architectures that have the ability to autonomously assemble and reconfigure in space [26]. This has academic merit in the fields of robotics, tele-robotics, and automation. These docking ports, herein referred to as the Universal Docking Port (UDP) ground prototypes, helped demonstrate spacecraft assembly in a laboratory environment. Figure 3-2 shows the selected mechanism design and accompanying avionics. These were combined with the SWARM air carriers to achieve successful satellite docking in three degrees of freedom on the MIT flat floor test facility (Figure 3-3). Docking was repeatedly achieved using these ground prototypes and the success of their operation directly drove the design of the flight configuration (Section 3.5).

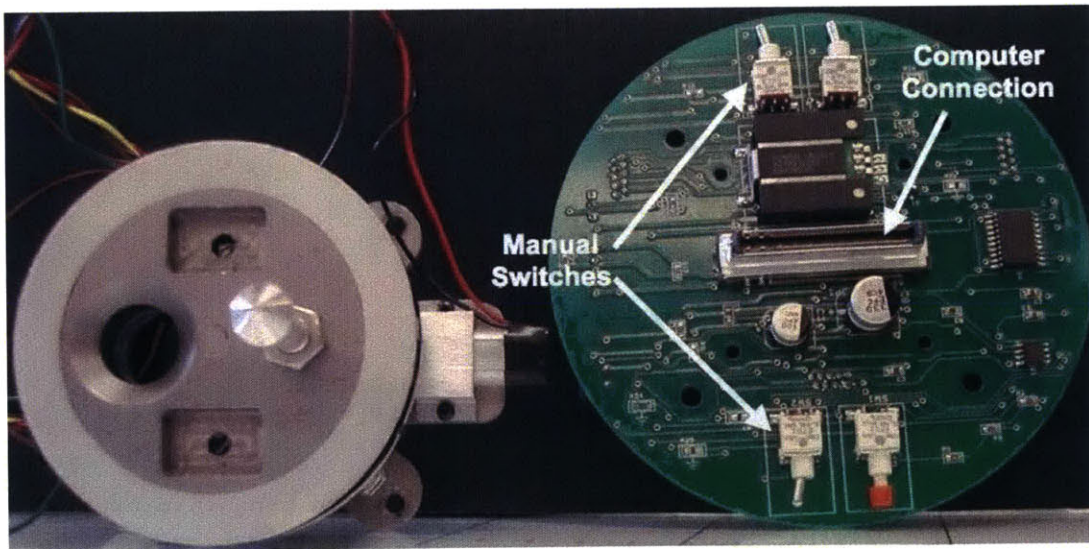


Figure 3-2: The SPHERES Prototype Universal Docking Port Mechanism and Accompanying Avionics Board

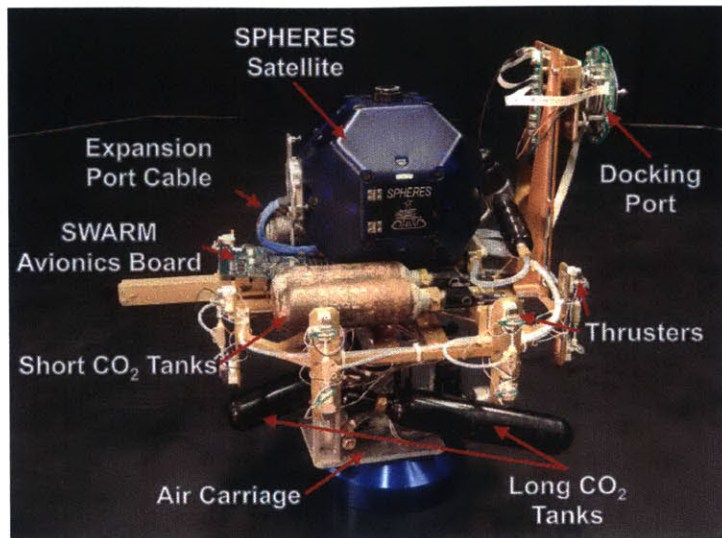


Figure 3-3: The SPHERES SWARM Docking Assembly on the MIT Flat Floor

The distinguishing characteristics of the selected UDP mechanical design were:

- Centralized
- Androgynous
- Inverse-symmetric
- Rigid
- Reusable

With the following attributes:

- Electrical Power Transfer: Although not shown in Figure 3-2, the prototype UDP incorporated electrical contacts on the docking face that supported power transfer between agents.
- Electromagnetic Attraction and Repulsion: An electromagnetic coil was wound about the circumference of the UDP to reduce the contact dynamics. The core of the UDP mechanism was made from iron to amplify the electromagnetic forces.
- Relative Sensing: Four ultrasonic transmitters and receivers were located around the docking face of the UDP. Range information was gathered using the ultrasonic chirps between satellites.

3.4 Flight Requirements Flowdown

This section details the requirements that drove the design of SPHERES Universal Docking Ports (UDPs). The schedule and budget of the InSPIRE-II contract required six flight UDPs to be designed, tested, and approved for launch in less than one year. The first step in the system design process was to evaluate lessons learned from ground testing of the prototype UDPs plus the additional safety and operational requirements levied by the NASA Safety and Human Factors Implementation Teams (HFIT).

3.4.1 Design Drivers

Although the prototype UDPs were compatible with the SPHERES and SWARM testbeds on the ground, the system design and requirements had to be re-evaluated in the context of in-space SPHERES operations. Specifically, there were five strategic design drivers that motivated changes to the prototype units.

- **Power:** The original UDPs were designed to be primarily operated through the SWARM carriage which independently boosted the available output voltage (and current) for the electromagnet. The flight UDP is limited to only 11.1V through the VERTIGO Avionics Stack expansion port, which was shown to be insufficient to generate electromagnetic forces of any significance.
- **Mass:** The added mass of a UDP reduces the rotational and translational control authority of the SPHERES. Moreover, with the addition of the Halo, up to six UDPs could be operated simultaneously from a single SPHERES. With the 6X multiplier on mass, it is desirable to keep the UDPs as lean and light as possible. Together with the power design driver, this motivated the elimination of the electromagnet (and accompanying iron core) from the flight design.
- **Safety:** Since the SPHERES and its payloads are operated inside of the ISS, the hardware is subject to additional safety and integration constraints. At a high level, the payloads must not endanger the health or wellbeing of the astronaut crew and must also pass a litany of tests and analyses including, but not limited to, Electromagnetic Interference (EMI), Electromagnetic Compliance (EMC), flammability, structural integrity (under high-g and kickloads), touch temperature, and acoustics.

- **Traceability:** The purpose of the SPHERES testbed and docking hardware is to advance satellite assembly algorithms that can scale to future space systems, such as the DARPA Robotic Servicing of Geosynchronous Satellites program. Thus, it is important to ensure that the hardware and software do not diverge significantly from architectures that could realistically be implemented in the hard vacuum of space. This partially motivated the removal of ultrasonics for relative sensing (pressure information cannot be transmitted through a vacuum). The addition of the optical camera maintains traceability with NASA's cross-enterprise roadmap documents which call for boresight visual docking in future satellite rendezvous missions.
- **Complexity:** An increase in design complexity compounds risk, schedule, and cost. Although the SPHERES facility is a risk-tolerant testbed for high risk space technologies, the flight UDPs were designed to have as few points of failure as possible to maximize the likelihood of successful docking and in-space operations.

3.4.2 Requirements Matrix

The objective of the UDP is to provide a mechanism for rigidly docking and holding two SPHERES satellites or other free-flyers together. These agents may impart contact impulses initially, so the docking mechanism must be able to maintain rigid capture despite forces and torques acting on and across the docking interface. The interface itself must be rigid so that it does not add additional dynamics to the system. Furthermore, the UDP must provide a capture cone for docking two SPHERES satellites together. This cone allows for slight misalignment in the orientation of two SPHERES satellites approaching docking while still locking into a set position. Correcting for slight misalignment ensures that the intended interface between the satellites is consistently established.

While both SPHERES satellites have global metrology, the UDP must also provide direct sensing between the two docking interfaces. This capability, provided by an onboard camera and visual fiducials, allows the SPHERES satellites to assess their relative pose in the approach phase prior to docking. Using relative sensing to supplement global metrology for docking replicates the approach a robotic servicer would take and provides a more realistic testing scenario. The camera need not be used in all docking maneuvers if global metrology is proven to provide a sufficiently accurate state solution.

Furthermore, the VERTIGO Avionics Box through which the UDP is attached blocks four of the Position and Attitude Determination System (PADS) ultrasound sensors. The UDP must replace these blocked sensors to maintain maximum global metrology capability. These metrology sensors are active whenever the UDP is attached to the VERTIGO Avionics Box for testing.

Table 3.2: Consolidated Requirements Matrix

Parameter	Requirement	Instantiation
Transfer Mechanical Loads	Docked modules shall create one large rigid system, enabling distributed controllers to actuate all connected elements as a single system.	Two counter-rotating disks are used to pinch and wedge the protruding pin of another UDP. By using a wormgear mechanism, the UDP cannot be back-driven and the disks stay locked when the motor is unpowered.
Large Docking Tolerances	Two modules must be able to dock even with misalignment errors.	The base of the lance is a cone and aligns with the chamfered entrance hole, thereby providing initial fine alignment.
Computer Control	The docking port shall be controlled and monitored from an onboard or remote computer.	A circuit board has been developed to interface between the UDP elements and the satellite.
Versatility	The docking port shall be compatible with the VERTIGO Avionics Box and the Halo exoskeleton.	The UDP standoff supports compatibility with both interfaces.

Continued on next page

Continued from previous page

Reusability	The docking port shall be reusable several times throughout its mission life.	The UDP is fully reusable because it uses an electrical motor, threaded rod, and counter-rotating disks for docking and undocking.
Sensor Replacement	The docking port shall replace any blocked sensors.	The UDP incorporates a metrology ring to replace all blocked sensors.
Direct Relative Pose Sensing	The docking port shall enable direct relative pose sensing.	The UDP camera module and accompanying fiducials enable high precision relative sensing.

3.5 Flight Design Review

In this section, we describe the detailed flight design of the UDP components. Figure 3-4 shows the initial flight configuration of the SPHERES docking ports. The docking port is operated by the VERTIGO Avionics Stack through a mechanical and electrical standoff. The flight UDPs have also been designed for two secondary modes of operation: through the SPHERES-Halo exoskeleton and through a purely mechanical standoff with the SPHERES (sans-VERTIGO), as shown in Figure 3-4.

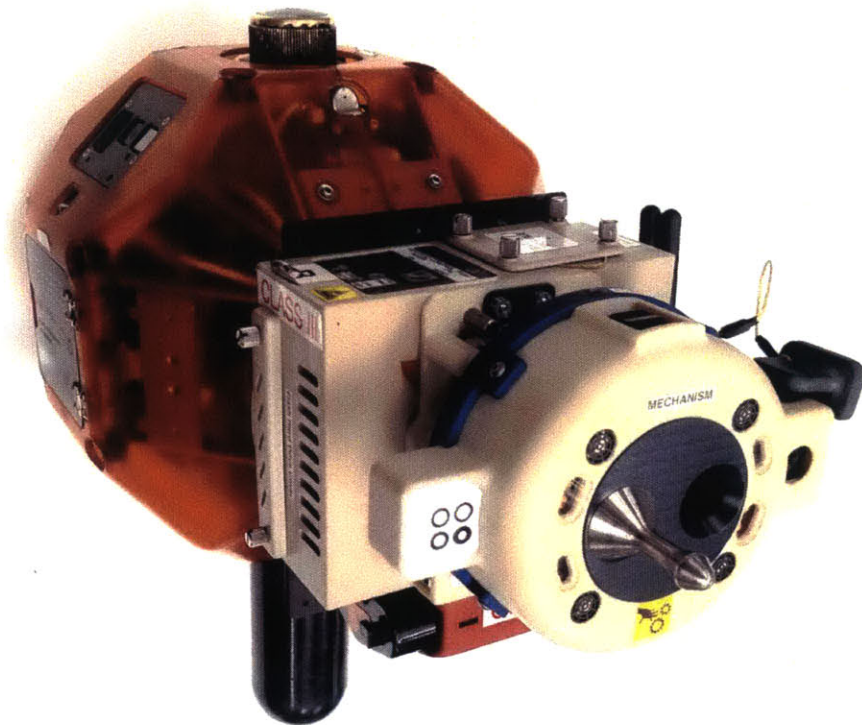


Figure 3-4: The SPHERES-VERTIGO-UDP Flight Assembly.

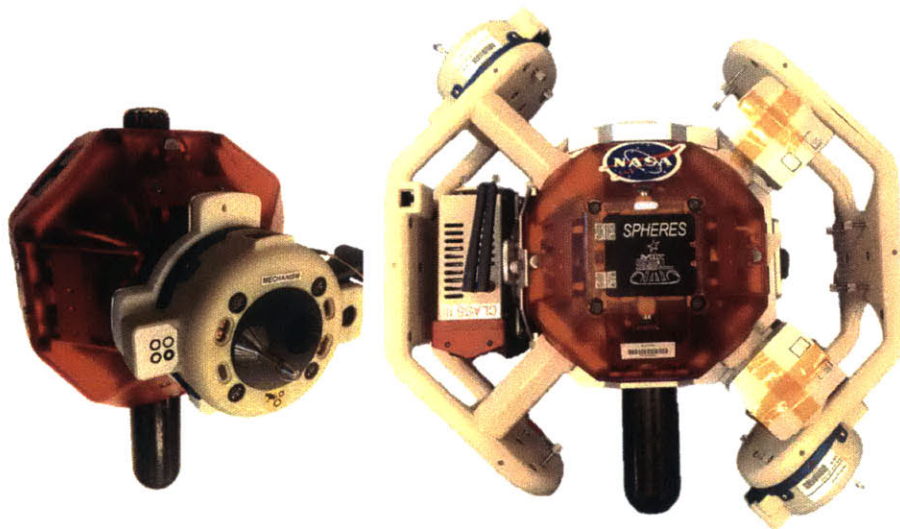


Figure 3-5: The Secondary In-Space Flight Configurations Through the Mechanical Standoff (left) and the Halo (right).

The docking port actuator is the first peripheral built off of the HaloGSP robotics

platform detailed previously in Chapter 2. The Halo exoskeleton supports up to six additional peripherals for a single SPHERES. The Halo supplies power and USB/Ethernet lines to the peripheral expansion port, which means that multi-satellite docking *and reconfiguration* are possible. The mechanical standoff configuration is necessary to perform three-satellite docking tests without a Halo in-space on account of the fact that only two VERTIGO Avionics Stacks exist on orbit. However, the mechanical standoff is particularly advantageous because the SPHERES maintains nearly full controllability (less mass) and the assembly center-of-mass shifts by an order of magnitude less. The UDP attached through the mechanical standoff acts a passive docking receptor, which is sufficient for most docking scenarios.

A total of ten flight-like UDPs have been built (with two pre-prototypes of the flight design), with the finest six being selected for flight to the ISS (Figure 3-6). Flight selection of the best UDPs was determined based on a rigorous unit evaluation including analysis of:

- | | | |
|---------------------------|-------------------------|--------------------------|
| • Operation Time | • Camera Calibration | Docking Repeatability |
| • Motor Strength | Variability | • Aesthetics (Scratches) |
| • Motor Stall Sensitivity | • Fiducial Blemishes | • Personality |
| • EMI Noise | • Fiducial Misalignment | (‘lucky-ness’) |
| • Acoustic Noise | • Optical Sensor | • Connector Behavior |
| • Stall Current | Misalignments | • Communication Drops |
| • Metrology | • Optical Sensor False | (FTDI, Camera) |
| Convergence | Positives/Negatives | • Major Electrical |
| | • ‘Best Family’ for | Component Failures |



Figure 3-6: The Fleet of SPHERES Universal Docking Ports Ready for Shipment.

3.5.1 Mechanical Overview

This subsection reviews the final mechanical design of the UDP. Figure 3-7 shows a side view of the UDP with the key drivetrain components indicated. The UDP mechanical system is focused on transferring rotational energy from an electric motor to a pair of counter-rotating disks around a central shaft. Along the docking bore sight of the UDP is a central axle that aligns the counter-rotating cams, Torrington NTA-411 thrust bearings, and a spacer. The spacer keeps the disks in place while also keeping the threaded motor shaft parallel with the back plate. The shaft pin translates through grooves in the counter rotating disks and the face plate, when the motor is actuated. These grooves in turn impart the required torque to rotate the cams. Both of the counter-rotating disks are almost identical to one another; by inverting one of them, the pin groove and the lance opening are oriented such that a translating shaft pin closes the lance opening. Additionally, the lance opening is teardrop shaped to capture the neck of the opposing lance.

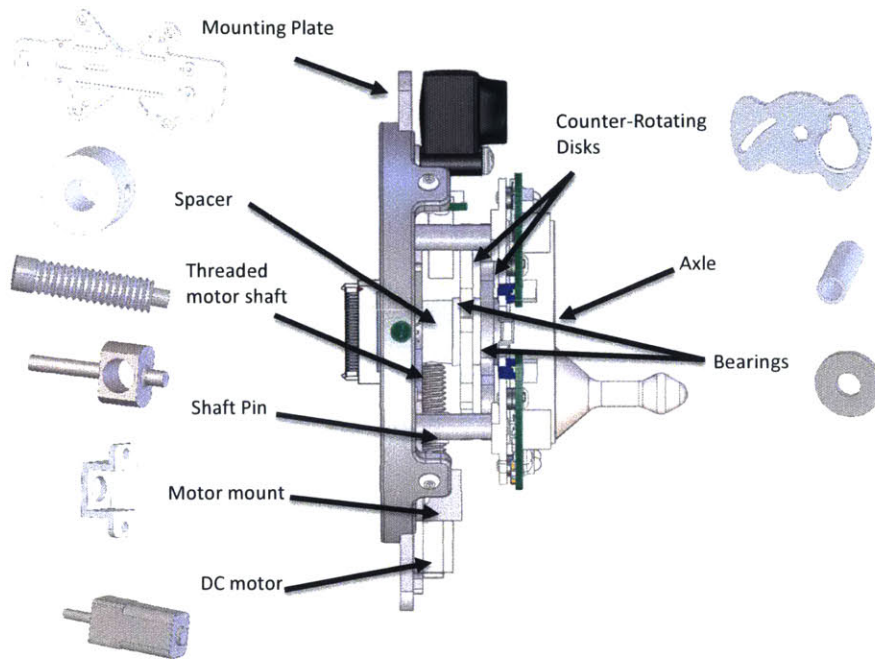


Figure 3-7: Exploded view of the internal UDP mechanism.

The lance itself incorporates a conic head for initial guidance into the opposing UDP, but also possesses an inverted conic base section that the counter-rotating disks can pull against. Therefore, the lance can guide itself into the opposing UDP despite small initial misalignments, and it can be pulled into place by the counter-rotating disks. The satellite to which a UDP is mounted is able to detect docking completion by reading a high current on the motor lines. This high current indicates that the motor has stalled and is no longer able to provide additional torque. In this condition, the lances of each docked UDP are pulled into place by the counter-rotating cams. Because of the enormous friction and high gear ratio present in a worm drive mechanism, power does not need to be supplied continuously to the motor to maintain rigid docking.

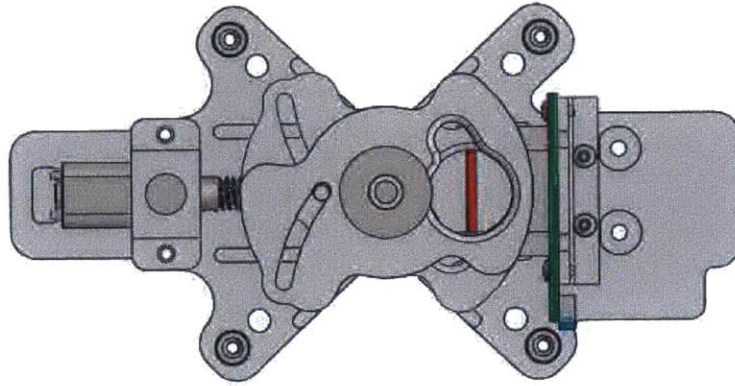


Figure 3-8: [Animation] Front View of An Animated Locking Sequence (Press Play to Start Video)

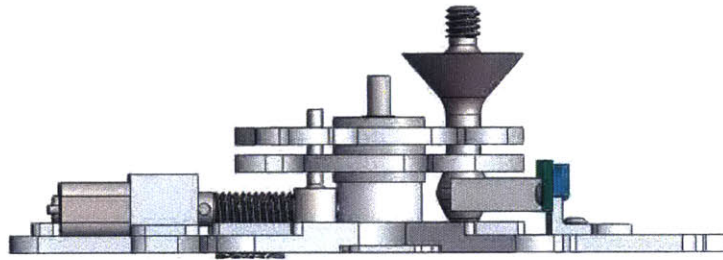


Figure 3-9: [Animation] Side View of An Animated Locking Sequence (Press Play to Start Video)

3.5.2 Electrical and Operational Overview

Electrically, the flight UDP combines the motor driver, the standard US/IR metrology schematics from SPHERES, and a VERTIGO camera module. Communication with the VERTIGO avionics box is through a single 50 pin connector (ERM8-020-09.0-S-DV-K-TR from Samtec) that supports four USB slaves, four ultrasonic bypass lines, and two IR transceiver packages. RS-232 serial communication is needed by the PIC microprocessor and is actually a converted USB line through an FTDI chip.

Capture is triggered when each lance fully enters the opposite hole. This configuration trips the photosensor, which in turn triggers the motor to drive the cam mechanism tight

Table 3.3: Specification Summary of the Flight UDP Compared with the Ground Prototype

	Ground Prototype	Flight UDP
Dimensions	7.6 cm X 3.8 cm	14 cm X 10.7 cm
Mass	0.45 kg	0.533 kg
Max Voltage	24 V	6 V
Peak Power	60 W	5 W
Max Docking Tolerance	± 1 cm, ± 2 deg	± 1 cm, ± 2 deg
Sensing Accuracy	1 cm, 0.5 deg	<2mm, <1 deg

three degree of freedom (DoF) facility for advancing hardware and software development of the SPHERES flight UDPs. This facility is important because it allows the scientists to iterate quickly on developed research algorithms, and the results can clarify and direct future investigations.

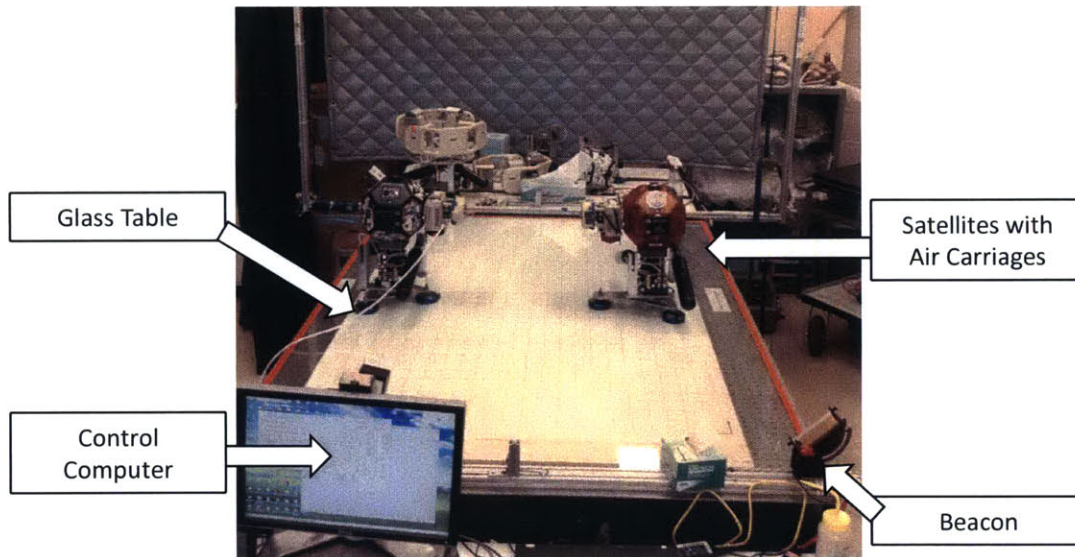


Figure 3-11: Glass Table Facility at MIT Used for Ground Testing of the UDP

3.6.1 Ground Test Objectives

The ground testing campaign was performed with four principle objectives. This provides the groundwork for six degree-of-freedom testing on NASA's reduced gravity aircraft (Section 3.7) and eventual ISS testing.

- A. Perform an integrated functional checkout of the docking port hardware with the VERTIGO

and SPHERES assemblies

- B. Show that the SPHERES global metrology can achieve consistent estimator convergence with the updated ultrasonic sensors from the UDP
- C. Show that SPHERES is controllable and agile with the new hardware, validating the updated SPHERES physical parameters and inertial properties
- D. Apply collision avoidance algorithms to cooperative spacecraft as formulated by visiting student Lorenzo Olivieri [6]

3.6.2 Concept of Operations

The concept of operations of the three DoF scenario was intended to be traceable to a potential on-orbit servicing mission. The overall idea is that during in-space servicing of satellites, there will be a desire to dock multiple spacecraft together and perform servicing tasks while connected. Servicing always begins with a successful docking operation.

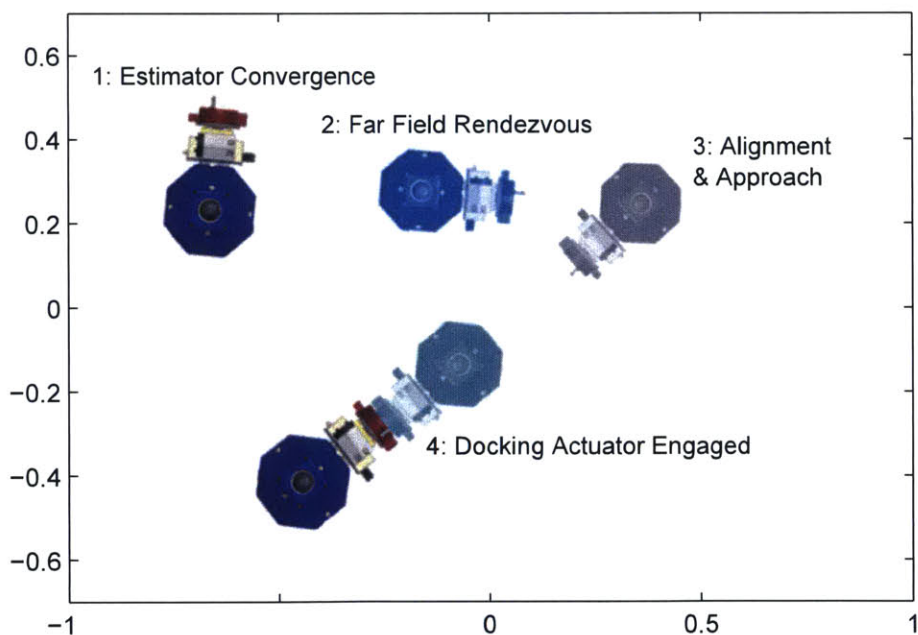


Figure 3-12: Glass Table Concept of Operations

The full docking scenario used for ground testing is shown in Figure 3-12. Phase 1 involves estimator convergence of the global metrology system, satisfying Ground Objective

B. Both satellites have been initialized to random locations in the activity space. This establishes a known state (position, orientation and velocities) for both the deputy and chief. In Phase 2, the active spacecraft pictured performs a waypoint following maneuver to navigate around the target spacecraft. During this maneuver, the satellite is continuously estimating a keep-out boundary around the target SPHERES to mitigate the risk of collision [6]. Next, the primary satellite aligns the face of the docking ports to prepare for docking. In the final maneuver, a series of software gates are passed as the satellite glides in for approach.

Most path-planning and navigation algorithms today for target approaches are based on the glideslope method, which is both a velocity controller and a trajectory planner for straight-line guidance. The idea is to approach the target with decreasing speed that approaches 0 as contact is initiated. Defining ρ as the distance from chief to deputy, and $\dot{\rho}$ as the velocity, the glideslope law is written as Equation 3.1.

$$\dot{\rho} = a \cdot \rho + \dot{\rho}_T \quad (3.1)$$

where $\dot{\rho}_T$ is the target speed (=0) and $a < 0$ is the glideslope. In this scenario, a has been tuned as a function of the initial speed and position, namely:

$$a = -\frac{\dot{\rho}_0}{\rho_0} \quad (3.2)$$

Because the SPHERES Docking Port no longer makes uses of electromagnets to mitigate contact dynamics, this type of control is needed to ensure small relative velocities during docking by reducing the thruster forces at the closest ranges.

3.6.3 Experimental Results

Figure 3-13 shows the glideslope approach in action. Once aligned with the target, the active satellite approached with a linearly decreasing velocity reaching almost 0 as it initiated contact. After a thorough analysis of the telemetry, and a comparison to the control authority of a non-docking port equipped SPHERES, Ground Objective C (controllability) was considered sufficiently satisfied.

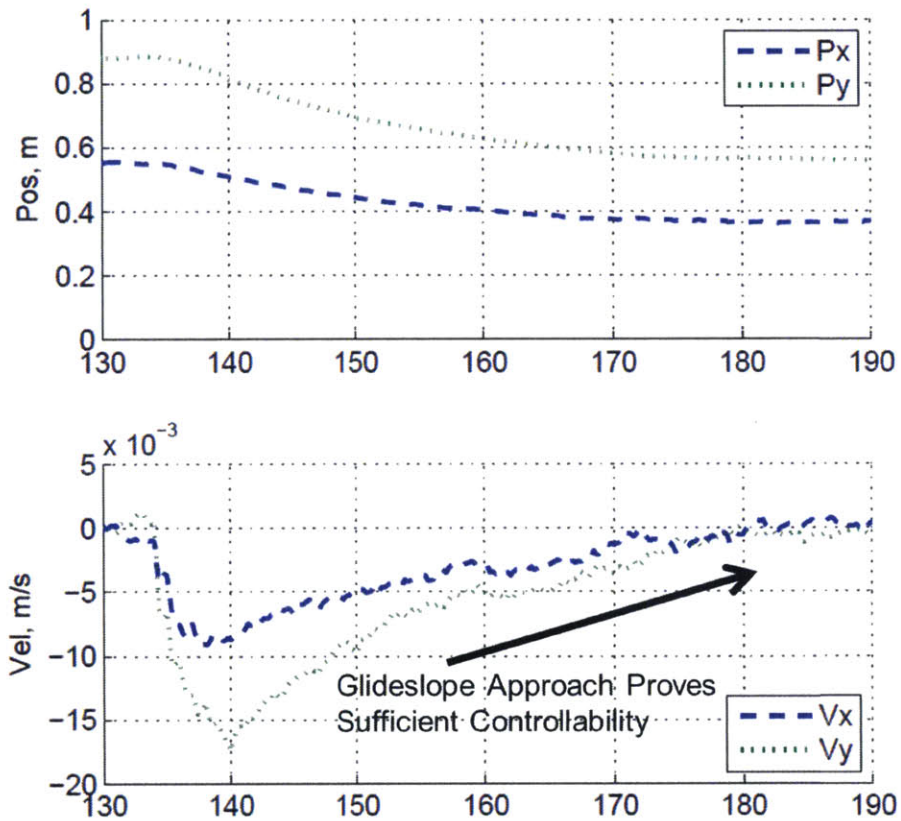


Figure 3-13: Velocity-Controlled Results from Three Degree of Freedom Testing on the MIT Glass Table (Adapted from [6])

In subsequent ground tests, the mechanisms properly engaged when triggered, meeting Ground Objective A. Figure 3-14 shows a sample docking sequence between satellites on the flat floor. A preliminary version of the fiducial tracking video stream is also shown in the top right corner. This specific test demonstrates that the capture mechanism is effective at low velocities in 3 DoF and that the integrated system (including the camera, global sensors, and SPHERES thrusters) functions reliably.

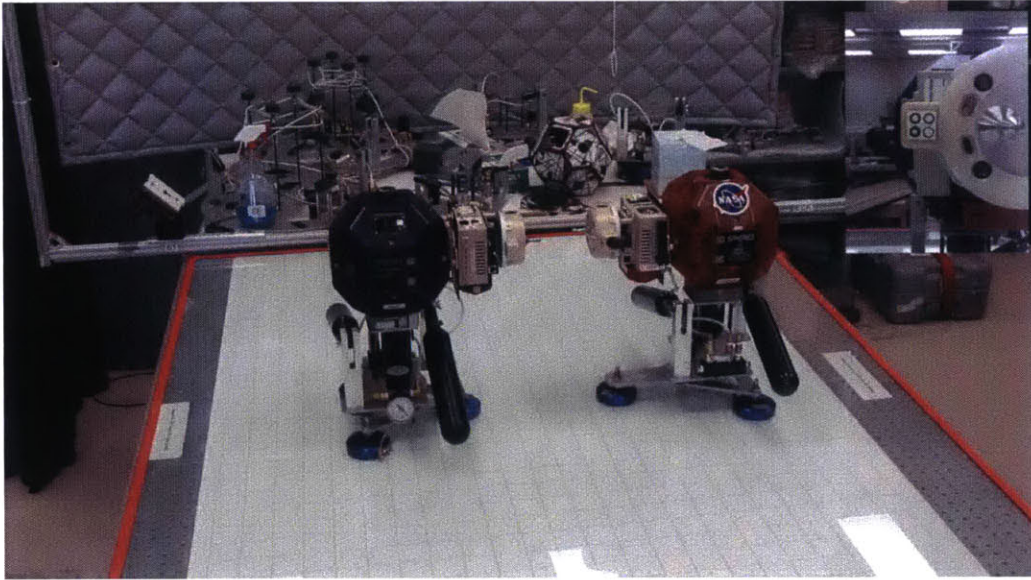


Figure 3-14: [Animation] A Successful Docking Test Achieved by the SPHERES on the Glass Table Using the Proto-Flight Docking Ports (2X Speed; Synchronized Camera Sensing Shown in Upper Right)

3.7 Validation in Six Degrees-of-Freedom

To maximize the ISS test session productivity, a parabolic flight campaign was necessary to bridge the gap between 2D and 3D docking and undocking operations. This section documents the incremental and iterative testing of the UDPs in a short-duration microgravity environment. While the UDPs have been successful in 2D ground testing, only 3 of the 6 degrees of freedom (DoF) were exercised. Six DoF docking simulations at MIT are being developed, but the weightless environment on NASA's microgravity plane is the true benchmark for free-floating flight experiments.

Because the weightless period of NASA's zero-gravity plane has a duration of only about 20 seconds, an end-to-end satellite docking sequence was unable to be performed. Fortunately, a docking maneuver can be conveniently discretized into four phases of flight: far field approach & rendezvous, proximity operations, docking, and maneuvering & repurposing. Each test flight day focused on a particular phase of the docking sequence. Reconstructing the data from all four flight days has provided confidence in our hardware and software

design for complete docking on the ISS.

3.7.1 RGA Objectives

The primary flight objectives *specific to the docking port actuator* on the Reduced Gravity Aircraft (RGA) flight were derived from the end goal of performing successful, repeatable in-space docking maneuvers. These were:

- A. Confirmation of the mechanical docking mechanism, including the updated drivetrain and motor, in a relevant environment using appropriate masses (no ground support equipment)
- B. Mitigation of the contact dynamics encountered between SPHERES during capture to ensure robust docking (the air carriages used on the ground misrepresent the inertia and friction ratios)
- C. Acceptable performance of the Resource Aggregated Reconfigurable Controller (RARC) during maneuvering of the conjoined spacecraft
- D. Validation that SPHERES can repeatedly undock in the relevant environment and flight configuration

The flight testing ensures that sufficient hardware testing has been performed on the payloads prior to operation aboard the ISS. Any adjustments, in either hardware or software, could be made to ensure the necessary precision and structures are in place for docking aboard the ISS. This approach, which has extensive heritage as part of the SPHERES program (the SPHERES satellites themselves were flown on a parabolic flight campaign, as was the Resonant Inductive Near-field Generation System, or RINGS, system), has saved valuable time aboard the ISS. With a manifested launch scheduled for Spring 2015, the timing of this July 2014 campaign allowed the team to verify the design of the hardware and software in time to make all necessary adjustments for improved ISS experimentation.

3.7.2 RGA Results

In this subsection, we first summarize the high level milestones achieved through the zero-gravity testing. Collectively this raises the Technology Readiness Level (TRL) of the

proposed actuator. Then, we present the methodology used in performing the augmented system identification of the SPHERES satellites with the proposed actuators installed.

Flight Summary and Experimental Hypotheses

This July 2014 flight campaign for SPHERES extended the capabilities of the satellites by validating four new configurations that had never been tested in a microgravity environment previously. The test configurations accomplished over the four flight days were: (1) a free flying SPHERES satellite with a Universal Docking Port (UDP), (2) a free flying satellite with a Halo and UDP, (3) two docked SPHERES and UDP assemblies, (4) two docked SPHERES, UDP and Halo assemblies. These configurations were designed to advance a modular approach to satellite docking and raise the TRL of the proposed docking actuator. Table 3.4 summarizes the major accomplishments.

Table 3.4: Summary of Test Configurations and Accomplishments

Flight Day	Test Configuration	Science Accomplished	Parabola Count
7/28	Fully Detached: SPHERES+UDP, SPHERES+Halo+UDP	<ul style="list-style-type: none"> → Inertial ID of SPHERES+UDP assembly → Inertial ID of SPHERES+Halo+UDP assembly → Plume impingement characterization of the Halo structure → Global metrology performance with a Halo in 6DOF 	Successful: 13 Minor Anomaly: 13 Major Anomaly: 9
7/29 Morning	Detached and Docked: SPHERES+UDP, SPHERES+UDP	<ul style="list-style-type: none"> → Capture mechanism demonstration in 6DOF → Undocking demonstration in 6DOF → Dual free-floating relative sensing using UDP cameras 	Successful: 28 Minor Anomaly: 12 Major Anomaly: 0
7/29 Afternoon	Docked: SPHERES+UDP, SPHERES+UDP	<ul style="list-style-type: none"> → Inertial ID of the docked assembly → Controllability of the aggregated system 	Successful: 4 Minor Anomaly: 35 Major Anomaly: 1
7/30	Docked: SPHERES+UDP, SPHERES+Halo+UDP	<ul style="list-style-type: none"> → Inertial ID of the docked assembly with Halo → Controllability of the aggregated system with Halo 	Successful: 28 Minor Anomaly: 20 Major Anomaly: 2

Using both quantitative and qualitative results from the zero gravity testing, we were able to confirm the following hypotheses about the docking port actuator.

Hypothesis: The space vehicles can successfully dock provided their relative approach velocity is small enough. The original realization of the Universal Docking Port incorporated an electromagnet that activated during the contact dynamics phases of docking and undocking.

Once the lance entered the target’s hole, both electromagnets would actuate and draw the satellites together. This feature ensured there would be no bounce-back, i.e. once the lance entered the hole, a successful docking was virtually guaranteed. However, this was de-scoped from the flight design (see Section 3.4).

During RGA flight testing, we demonstrated a successful capture repeatedly, roughly 6 times. Bounce-back was not observed when the satellites had sufficiently small approach velocities (less than 1 cm/s). The capture mechanism was able to grab the lances without compromising the dock. This increases our confidence for on-orbit testing.

Hypothesis: The SPHERES satellites are able to successfully undock without the electro magnetic force. The previously baselined electromagnet’s secondary function was to provide a magnetic repulsion force during undocking. Without it, undocking is performed solely with cold gas thrusters. There was thought to be a risk of lance-mechanism interference that the thrusters may not be able to overcome. During RGA flight testing we were able to confirm that the thrusters alone are capable of performing a successful undocking maneuver. No anomalies were observed.

Hypothesis: The physical satellite centers of mass and inertias match expected results from CAD. Most numerical models suffer from the same fault known as GI-GO (Garbage In, Garbage Out). If our model is any way in error, then our control and performance will be negatively affected. Thus, we performed system identification maneuvers on Days 1, 3 and 4. Not only will this accelerate on-orbit testing, it also pushed us to advance our software to support this type of testing. This analysis is detailed in Subsection 3.7.2. In general, our results matched our expectations, although we will use this information to tune our controllers in the future.

System Identification Methodology

In this subsection, we present the analysis methodology used for mass identification with the new docking port actuator installed on the SPHERES. Our analysis shows (1) a statistical agreement between predicted and experimental mass properties, and (2) that the proposed actuator does not prohibitively decrease the controllability of the satellite. However, there were significant lessons learned that will be further investigated during ISS operations, namely, a characterization of the rigidity of the dock.

Most of the key data received from Flights 1, 3 and 4 was in the form of high-frequency

measurements from the accelerometers and gyroscopes onboard the SPHERES satellites. For each test that was run (i.e., each parabola), the SPHERES satellites recorded this acceleration and angular velocity data for processing afterward. An example of one parabola of acceleration data is shown in Figure 3-15. The accelerometers saturate at approximately 0.25 m/s^2 (25 milli-Gs), so one can see where the satellite is in free-fall fairly easily. The noisy data that seems to jump randomly between -0.25 m/s^2 and $+0.25 \text{ m/s}^2$ represent areas where the satellites are being handled by the team, and the areas where the data is roughly zero show when the satellite is floating. It is these floating sections where the interesting measurements are recorded.

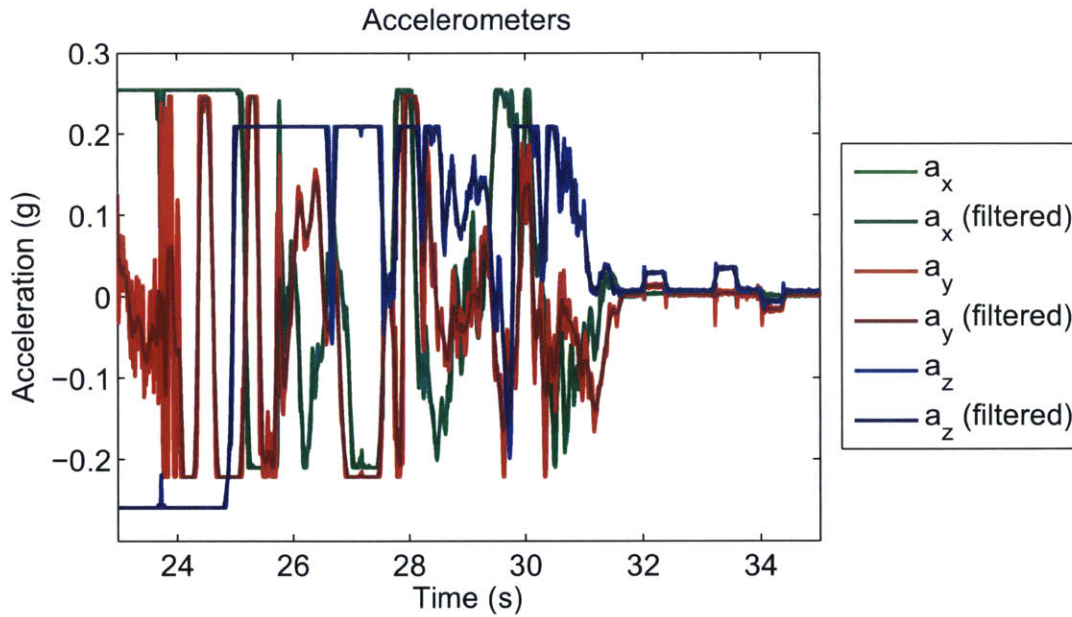


Figure 3-15: Sample Accelerometer Response Throughout an Entire Parabola (Zero-gravity Begins at 31s).

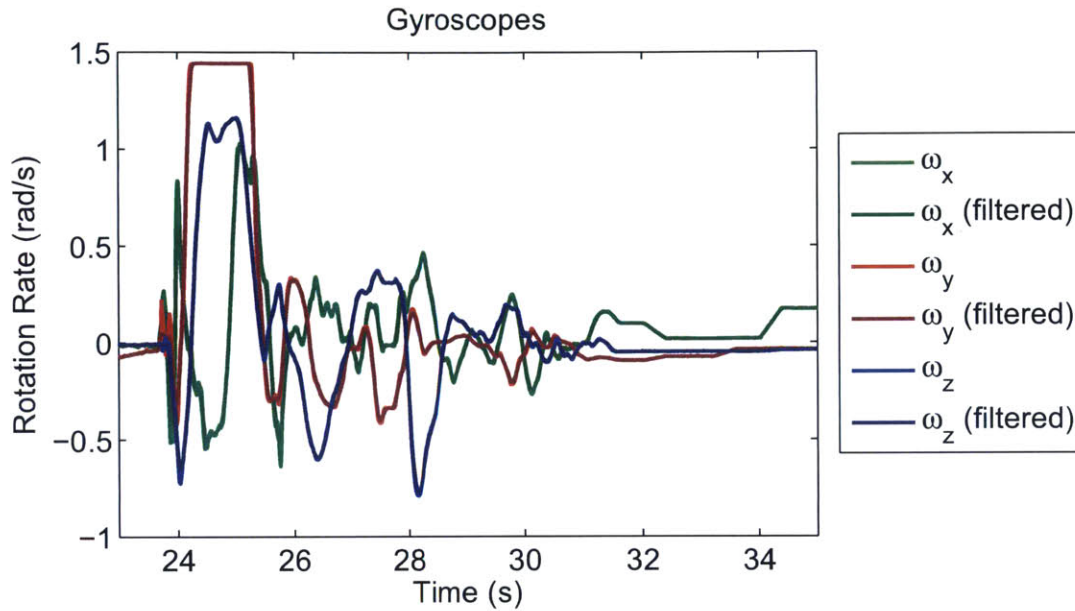


Figure 3-16: Sample Gyro Response Throughout an Entire Parabola (zero-gravity begins at 31s).

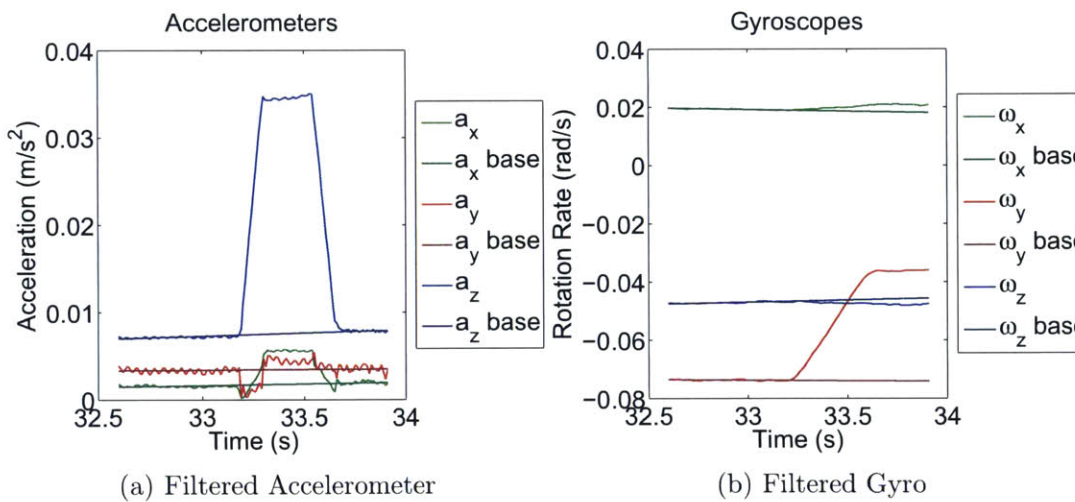


Figure 3-17: Raw Accelerometer and Gyro Data with Moving Average Filter

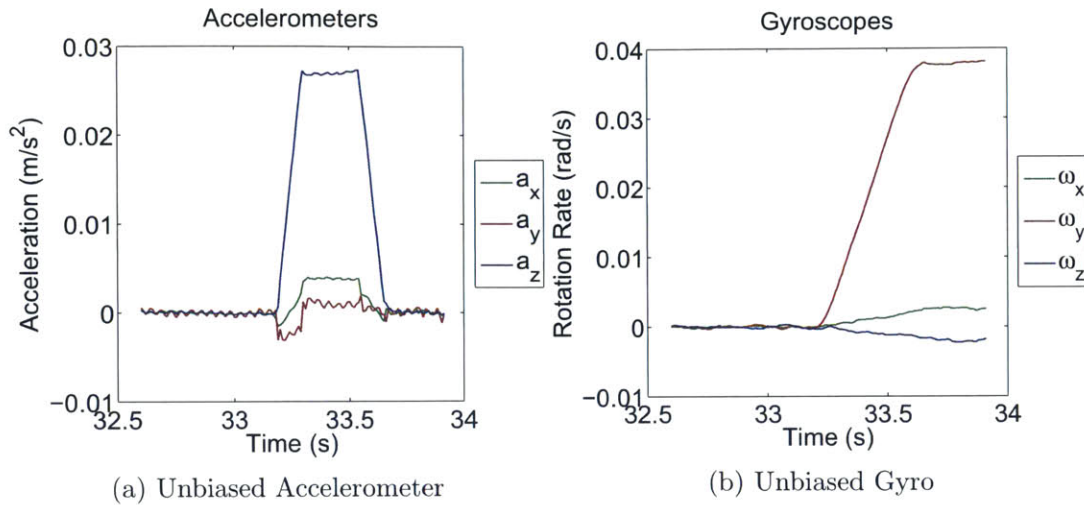


Figure 3-18: Final Unbiased Acceleration and Angular Velocity Response to a Single Thruster Pulse.

The thrusters of the satellite fire for a set amount of time and impart acceleration to the system. The thruster pulses can be seen in the data as square acceleration pulses shown in red in Figure 3-17a and detailed further in Figure 3-18a. The angular velocity of the satellite changes linearly while the thruster is on and remains constant when the thruster is off. The resulting gyro behavior will read a change in angular velocity from before to after the thruster was fired. Figure 3-17 shows an example of data from accelerometers and gyros for a single thruster pulse.

Because the underlying raw data can be quite noisy at times, a moving average filter is applied to remove the high-frequency noise. After this moving average is extracted from the raw data, the bias of each sensor needs to be subtracted. This bias is approximated by a linear regression as shown in Figure 3-17. The linear fit is then subtracted from the processed data and the final zero-base acceleration and angular velocity measurements are known. From Figure 3-18, the height of the acceleration square pulse from each accelerometer tells the amount of acceleration the thruster imparted on the system while it was on. The change in angular velocity can be seen in the gyro plot as the difference between the pre- and post-pulse heights.

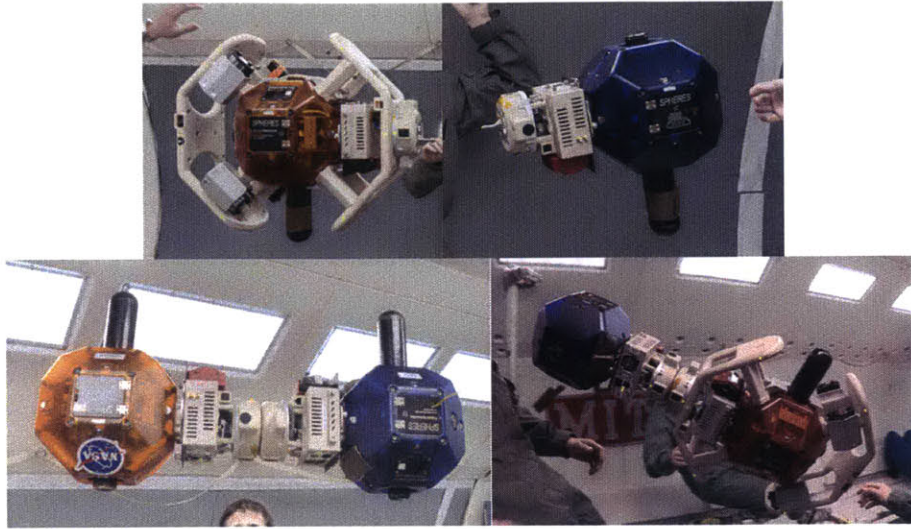


Figure 3-19: Configurations for the System Identification Analysis on Flight 1 (top left and right), Flight 3 (bottom left), and Flight 4 (bottom right).

This pulse analysis process is performed on all thruster firings for each of the flights. With this large collection of data and strategically designed thruster firings, the center of mass and moments of inertia of the systems are able to be computed. Essentially, the dynamic equations of motion can be rearranged to form a least-squares minimization problem, with the location of each accelerometer from the center of mass as an output. Because the locations of the accelerometers with respect to the geometry of the satellite are known, the position of the center of mass can be extracted from this data. Once the center of mass is known, the theoretical torque exerted on the system by each thruster can now be computed as the cross-product between the lever arm of each thruster and the force of the thruster. Given enough data points, comparing the theoretical torque to the output angular acceleration will yield the moments of inertia of the system. Additionally, the magnitude of the acceleration from each pulse can be compared for impinged and unimpinged thrusters to estimate a percentage of impingement and achieve another goal of the testing (not discussed herein).

The first realization in processing the accelerometer data is that the sensor is in a rotating reference frame and follows the well-known kinematics equation for acceleration in a non-inertial frame:

$$\mathbf{a}_r = \mathbf{a}_i - 2\boldsymbol{\omega} \times \mathbf{v}_r - \boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{r} - \boldsymbol{\alpha} \times \mathbf{r} \quad (3.3)$$

Here, \mathbf{a}_r is the acceleration measured by the accelerometer, \mathbf{a}_i is the linear acceleration of the center of mass of the assembly, $-2\boldsymbol{\omega} \times \mathbf{v}_r$ is the Coriolis acceleration which is 0 in this scenario (the accelerometer is fixed in the body frame), $-\boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{r}$ is the centripetal acceleration caused by the rotation of the assembly about the center of mass, and $-\boldsymbol{\alpha} \times \mathbf{r}$ is the Euler (or transverse) acceleration. In order to form a linear model for solving the least square estimate of \mathbf{r} , the center of mass, Equation 3.3 has been linearized and expanded into the array Equations 3.4. This linearization is valid because of the small rotation rates induced by a single thruster pulse. The measurement noise from the sensor is neglected in this analysis.

$$\begin{aligned}
a_x &= \frac{F_x}{m} + \alpha_z r_y - \alpha_y r_z \\
a_y &= \frac{F_y}{m} + \alpha_x r_z - \alpha_z r_x \\
a_z &= \frac{F_z}{m} + \alpha_y r_x - \alpha_x r_y
\end{aligned} \tag{3.4}$$

We note that the linear acceleration of the center of mass is described by $\frac{F_i}{m}$ and has been substituted accordingly. Next, the accelerometer readings from each thruster firing was recorded and stacked into matrices according to the familiar $Ax = b$ form. From this, x can be solved for as simply $x = A^{-1}b$.

$$\begin{aligned}
\begin{bmatrix} \vdots \\ a_x - \frac{F_x}{m} \\ \vdots \end{bmatrix} &= \begin{bmatrix} | & \vdots & \vdots \\ 0 & +\alpha_z & -\alpha_y \\ | & \vdots & \vdots \end{bmatrix} \begin{bmatrix} r_{x,a_x} \\ r_{y,a_x} \\ r_{z,a_x} \end{bmatrix} \\
\begin{bmatrix} \vdots \\ a_y - \frac{F_y}{m} \\ \vdots \end{bmatrix} &= \begin{bmatrix} \vdots & | & \vdots \\ -\alpha_z & 0 & +\alpha_x \\ \vdots & | & \vdots \end{bmatrix} \begin{bmatrix} r_{x,a_y} \\ r_{y,a_y} \\ r_{z,a_y} \end{bmatrix} \\
\begin{bmatrix} \vdots \\ a_z - \frac{F_z}{m} \\ \vdots \end{bmatrix} &= \begin{bmatrix} \vdots & \vdots & | \\ +\alpha_y & -\alpha_x & 0 \\ \vdots & \vdots & | \end{bmatrix} \begin{bmatrix} r_{x,a_z} \\ r_{y,a_z} \\ r_{z,a_z} \end{bmatrix}
\end{aligned} \tag{3.5}$$

The inertia tensor could similarly be estimated using the batch least squares. Euler's equation for rotational dynamics in the body-fixed frame is given as shown in Equation 3.6. In the Body Fixed Frame (BFF), the inertia tensor is constant for a rigid body, which is

convenient for this application.

$$\boldsymbol{\tau} = \mathbf{I}\boldsymbol{\alpha} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) \quad (3.6)$$

For our linear estimator, we could neglect the $\boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega})$ term, leaving us with the familiar Newton's Second Law ($F = ma$) for rotational dynamics. With an appropriate data set, this could be solved using an equivalent methodology as proposed for the center of mass, but this time for the six element, symmetric inertia tensor. Knowing the input torque ($\boldsymbol{\tau}$) from each thruster pulse and measuring the slope of each gyroscope during the pulse (Figure 3-18), the three simultaneous dynamics equation for each axis can be re-arranged to solve for the inertia tensor as follows. Note that the thruster pulses are assumed to be perfectly aligned with the geometric axes of the SPHERES.

$$\boldsymbol{\tau} = [\mathbf{I}]\boldsymbol{\alpha} = \begin{pmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{pmatrix} \begin{pmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{pmatrix} = \begin{pmatrix} I_{xx}\alpha_x - I_{xy}\alpha_y - I_{xz}\alpha_z \\ I_{yy}\alpha_y - I_{yx}\alpha_x - I_{yz}\alpha_z \\ I_{zz}\alpha_z - I_{zx}\alpha_x - I_{zy}\alpha_y \end{pmatrix} \quad (3.7)$$

A single x-axis pulse would generate the following rotational dynamics.

$$\begin{pmatrix} \tau_x \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} I_{xx}\alpha_x - I_{xy}\alpha_y - I_{xz}\alpha_z \\ I_{yy}\alpha_y - I_{yx}\alpha_x - I_{yz}\alpha_z \\ I_{zz}\alpha_z - I_{zx}\alpha_x - I_{zy}\alpha_y \end{pmatrix} \quad (3.8)$$

Compiling pulses from all three axes provides the necessary number of equations to solve for the 3x3 (but 6 element) inertia tensor, with similar systems of equation for the I_{yy} and I_{zz} components (the products of inertia should be symmetrical). Additional thruster pulses reduces the experimental uncertainty and can be stacked on the 'A' matrix from Equation 3.9.

$$\begin{pmatrix} \tau_x \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha_{x,x} & -\alpha_{y,x} & -\alpha_{z,x} \\ \alpha_{x,y} & -\alpha_{y,y} & -\alpha_{z,y} \\ \alpha_{x,z} & -\alpha_{y,z} & -\alpha_{z,z} \end{pmatrix} \begin{pmatrix} I_{xx} \\ I_{xy} \\ I_{xz} \end{pmatrix} \quad (3.9)$$

where $\alpha_{i,j}$ corresponds to the angular acceleration in the i^{th} axis from a j^{th} thruster pulse. Unfortunately, our data set did not possess statistically significant cross-coupled accelerations to estimate the products of inertia. As evident from Figure 3-18, the induced accelerations due to the products of inertia are an order of magnitude smaller than the rotation rates about the principle body fixed axis. Moreover, the induced accelerations were often dirty (due to aircraft dynamics) or not measurable all together. Thus a simpler method is proposed for estimating the products of inertia by scaling the CAD results by a factor equal to the percent difference between the experimentally derived principle moments (Equation 3.10) and the CAD.

$$\begin{aligned} I_x &\approx \tau_x / \alpha_x \\ I_y &\approx \tau_y / \alpha_y \\ I_z &\approx \tau_z / \alpha_z \end{aligned} \quad (3.10)$$

System Identification Results

Overall, on Flight 1, 64 thruster pulses were recorded for the Halo-equipped satellite and 19 from the UDP-equipped satellite. This amount of data was sufficient to estimate the center of mass and principal moments of inertia of the system. On Flight 3, only 17 total thruster pulses were recorded, so only a rough estimate of inertia was possible for two of three axes. However, the center of mass was estimated fairly well. On Flight 4, 128 combined pulses were recorded, which also provided sufficient data to determine center of mass and inertia values.

The testing campaign also exposed a few additional problems that we may or may not experience on the ISS. First, there seemed to be a noticeably higher noise level on the Halo-equipped satellite system, than the UDP-only-equipped system. This noise could simply arise because the sensors on that satellite are worse, but could also be due to vibrations induced in the Halo structure and UDP actuator, either from collisions, or from the acoustic noise on the plane. Regardless, the increased noise level on the system made the data

Table 3.5: Experimental and CAD Predictions of the Assembly Centers of Mass

Configuration	X_{cg} (cm)	Y_{cg} (cm)	Z_{cg} (cm)
UDP (Test)	4.028	-1.009	-0.902
UDP (CAD)	4.313	0.201	-0.171
<i>Absolute Difference</i>	<i>0.285</i>	<i>1.21</i>	<i>0.731</i>
UDP + UDP* (Test)	11.007	1.081	-0.238
UDP + UDP* (CAD)	25.09	0	-0.171
<i>Absolute Difference</i>	<i>14.083</i>	<i>-1.081</i>	<i>0.067</i>
Halo (Test)	0.477	-0.755	-0.775
Halo (CAD)	-0.913	0.012	-0.051
<i>Absolute Difference</i>	<i>-1.39</i>	<i>0.775</i>	<i>0.724</i>
Halo + UDP (Test)	27.953	0.073	0.563
Halo + UDP (CAD)	15.814	-0.039	-0.048
<i>Absolute Difference</i>	<i>-12.1393</i>	<i>-0.112</i>	<i>-0.611</i>

Table 3.6: Experimental and CAD Predictions of the Assembly Inertia Matrices

Configuration	I_{xx} (kg-m ²)	I_{yy} (kg-m ²)	I_{zz} (kg-m ²)	I_{xy} (kg-m ²)	I_{xz} (kg-m ²)	I_{yz} (kg-m ²)
UDP (Test)	3.360 E-02	5.720 E-02	5.020 E-02	-6.272 E-04	-3.480 E-04	2.609 E-04
UDP (CAD)	3.023 E-02	6.406 E-02	5.671 E-02	-6.601 E-04	-3.662 E-04	2.746 E-04
<i>Percent Difference</i>	10.02%	-11.99%	-12.96%	-	-	-
UDP + UDP (Test)	6.800 E-02	1.063 E+00	1.055 E+00	-	-	-
UDP + UDP (CAD)	6.055 E-02	1.502 E-01	1.355 E-01	-4.443 E-02	-4.472 E-02	-4.472 E-02
<i>Percent Difference</i>	10.959%	85.872%	87.158%	-	-	-
Halo (Test)	1.005 E-01	1.735 E-01	1.557 E-01	9.876 E-05	-3.025 E-03	-9.558 E-04
Halo (CAD)	1.039 E-01	2.115 E-01	1.670 E-01	1.108 E-04	-3.394 E-03	-1.072 E-03
<i>Percent Difference</i>	-3.42%	-21.91%	-7.27%	-	-	-
Halo + UDP (Test)	1.360 E-01	1.563 E+00	1.534 E+00	-	-	-
Halo + UDP (CAD)	1.342 E-01	2.421 E-01	2.793 E-01	-2.928 E-02	-2.694 E-02	-2.685 E-02
<i>Percent Difference</i>	1.31%	84.51%	81.80%	-	-	-

analysis much more difficult than it has been on the ISS in the past. A second discovery that arose from this testing was that thruster forces from one satellite did not reliably appear in the accelerometers of another docked satellite. This discovery implies that the docking is not perfectly rigid and that the force either is damped out below the noise floor or translated into an impulsive spike from contact dynamics. The latter seems more likely, as spikes in acceleration are seen at intervals that agree with when the other satellite is firing its thrusters. The effects of this are shown in the poor estimations of the C.M. and moments of inertia of the docked configurations in Table 3.5 and Table 3.6. These errors have been attributed to the high stresses absorbed by the assemblies during the 2-g pullout maneuver of the aircraft, which had a tendency of loosening thumbscrews. This is not expected to be an issue during ISS testing.

The moments of inertia presented in Table 3.6 are taken about the center of mass of the assembly and aligned with the geometric axes of the master SPHERES. The author has highlighted the rows in Table 3.6 corresponding to the *best estimate* of the moment of inertia of the configuration. The experimental values are suggested to be used for the single assembly only configuration, using scaled products of inertia derived from the CAD estimate's products of inertia and the percent difference from experimental to CAD. In the docked configuration, the CAD estimates are recommended to be used until experimental data is available from ISS test sessions.

3.8 Lessons Learned

A variety of complex problems were overcome throughout the development of the SPHERES Universal Docking Port. Three of the most significant lessons learned should be kept fresh during the development of future spaceflight systems for SPHERES and others.

- **Requirements Tradeoffs:** The final flight design of the UDP is not perfectly optimized for science. Instead, there were numerous instances when ISS safety and integration requirements took priority. Although some non-ideal requirements were challenged, there were also losses that were absorbed (via cost, schedule, or science). The lesson learned is understanding that while there maybe conflicting requirements or interests on specific design features, the payload developer must know when to push back and when to concede in order to achieve an agreeable solution for all parties.

- **Importance of End-to-End Integrated Testing:** Only through end-to-end integrated testing can there be full confidence in the performance of the product. Full integrated testing was not performed early enough on the UDP prototypes to catch the effect of the camera cover magnets on a failed docking maneuver. In retrospect, the camera cover should have used Velcro instead of magnets (which create a well of attraction with the steel lance). Similarly, the camera should have been positioned a bit farther radially from the UDP center to better avoid visual interference from the UDP core during poor angular alignments. Finally, it was not predicted ahead of time that the conformal coating on the flight units would disrupt the VERTIGO-to-UDP communication lines. Fortunately, these three design flaws were able to be overcome or mitigated through software or operational procedures.
- **Electromagnetic Interference (EMI) Requirements:** It was not predicted ahead of time that the embedded DC motor inside of the UDP would significantly jeopardize the EMI emissions profile of the payload. As a result a novel EMI filter had to be designed and installed on the flight units what should have been just weeks before final delivery. This cost the program significant money and schedule.

3.9 Planned ISS Operations

Thus far, this chapter has documented the flight hardware design and validation of a rigid, androgynous docking port for satellite servicing operations. This section specifically provides a sampling of the research capabilities enabled by this hardware upgrade to the SPHERES facility. Once on station, two-satellite test sessions can begin immediately. Figure 3-20 shows the proposed assembly scenarios that can be further studied on-orbit.

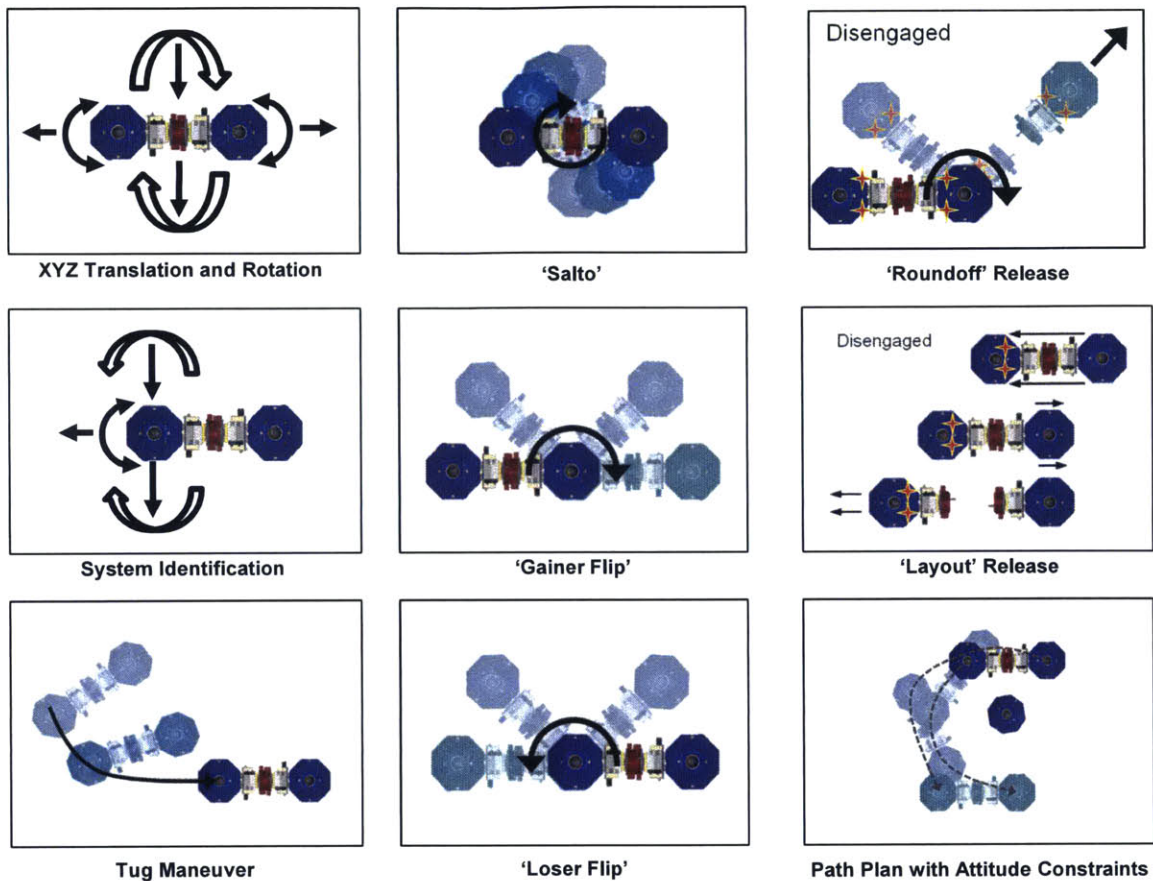


Figure 3-20: Portion of the UDP Space Operation Plan

- **XYZ Translation and Rotation:** The combined satellite system has the ability to translate linearly forward and backward in the X-Y-Z dimensions as well as rotate about the X-Y-Z axes.
- **System Identification:** An active satellite may dock to a passive target and perform a system identification maneuver before adapting its control gains to support the new inertial properties of the system.
- **Tug Maneuver:** After computing and adapting to the additional mass, the servicing satellite may tug the passive agent into a new orbit for an extended mission life or retirement.
- **Salto:** In testing the performance of the novel Resource Aggregated Reconfigurable Controller (which engages after docking), a series of acrobatic maneuvers can be

investigated. The Salto maneuver, a classical acrobatic roll, can be initiated which rotates about the combined center of mass.

- **Gainer Flip:** The Gainer is a traditional acrobatic flip consisting of a backwards somersault while still moving forward. The axis of rotation should be about the master SPHERES, and has applications to reusable launch vehicles and spacecraft.
- **Loser Flip:** The Loser is the mirror trick of the gainer, wherein the agent ends the maneuver behind its initial position due to backward momentum.
- **Roundoff Release:** A series of interesting thruster failures can be investigated to advance the field of Fault Detection Isolation and Recovery (FDIR). In one scenario, the +X axis thruster of both satellites can be faulted in software. This allows the SPHERES to be accurately navigated to achieve docking. However, they do not possess the ability to undock in the traditional sense (linearly). Through the proposed Roundoff release, an innovative use of the fictional centrifugal force, the agents may still achieve a successful separation.
- **Layout Release:** The Layout release may be employed when only one agent's +X axis thrusters are faulted.
- **Path Planning with Attitude Constraints:** A third SPHERES satellite may be utilized as a representative debris sample. This presents a research problem of high intellectual merit. In many cases, satellites must maintain sun- or Earth-pointing during operations. Pointing constraints may be active during these debris avoidance activities, which poses a non-trivial optimization problem.

Test session time on the ISS is limited, especially considering that much of the allocated time for a SPHERES session is necessary time for setup and shutdown procedures (Figure 3-21). Although much of the science proposed in Figure 3-20 can be accomplished within the three allocated UDP test sessions, Halo operations with up to six UDPs can enable unprecedented advances in satellite reconfiguration that is not possible with two UDP-only assemblies.

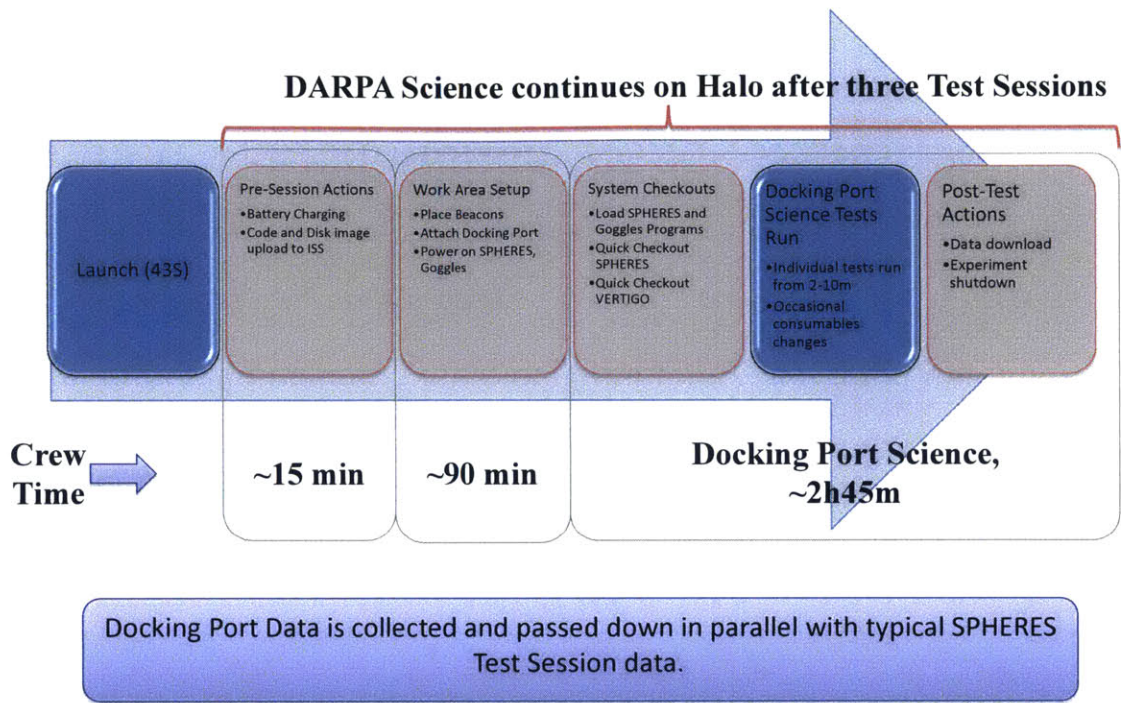


Figure 3-21: SPHERES Docking Port Test Session Plans Build Toward Halo Operations

Chapter 4

Sensor: Relative Pose Estimation of a Spin-Stabilized Spacecraft

4.1 Overview

This chapter describes the integration of a relative pose estimation sensor to be used for satellite proximity operations. In general, determining with confidence the relative state of a free-floating body has a wide variety of applications. The inspection of drifting or tumbling satellites is of particular interest to the Defense Advanced Research Projects Agency (DARPA). The target could be a military asset, an asteroid, space debris, a comet, or an uncooperative agent. Once characterized the target can be subsequently docked to, serviced or otherwise studied. For example, DARPA Phoenix has proposed a satellite assembly architecture that requires high precision relative sensing in order to aggregate small modules into larger operable satellites.

Visual sensing using known geometries already has heritage in space but is still a problem of high interest. Identifying known geometries through image processing of fiducial markers is well known - indeed it has been in use on the International Space Station (ISS) for several years (Figure 4-1). However, traditionally the problem is formulated in the context of a dynamic observer and a static target (for navigation). The applications of a static observer and a dynamic (e.g. spin-stabilized) target are less well known. Therein lies the research gap which is the focus of this chapter.

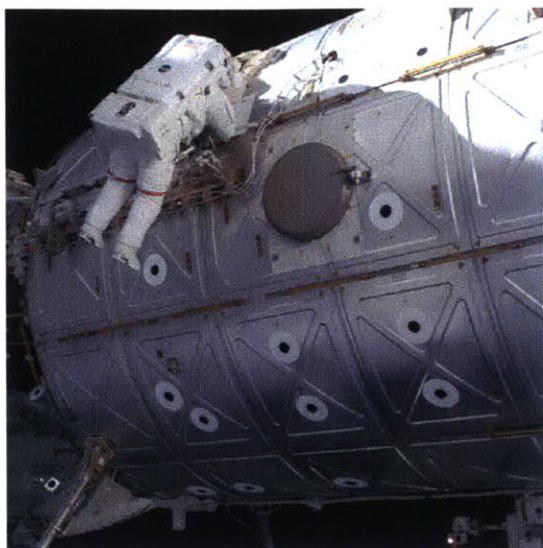


Figure 4-1: Concentric Circle Fiducial Markers Have Previously Been Used on the Exterior of the ISS

This chapter begins by reviewing the requirements and selection of the flight hardware sensor integrated within the Universal Docking Ports for SPHERES. Then we show how this optical camera can be used to generate real time estimates of a target state for docking, presenting the methodology, calibration, and estimated accuracies. Finally, in this chapter we develop a six degree of freedom stochastic simulator of a rigid body spacecraft. With this we explore the performance of two filters in the presence of nonlinear dynamics and non-Gaussian random noise in both simulation and real hardware:

1. A Multiplicative Extended Kalman Filter
2. An Unscented Kalman Filter using the same Multiplicative parameterization of quaternions

The results presented herein consist of Montecarlo simulations and testbed data (three and six degree-of-freedom) to show that the designed estimation filters work effectively. Using this knowledge, the sensing techniques are being applied directly to the SPHERES testbed on the International Space Station during on-orbit operations to increase the probability of a successful docking maneuver.

4.1.1 Requirements

This sub-section reviews the requirements that drove the selection and integrated design of the SPHERES UDP relative sensor. SPHERES utilizes a global metrology system consisting of ultrasonics and infrared LEDs. This metrology system provides an estimated bias accuracy of about ± 1 cm and ± 3 deg with a variance of ± 0.2 cm and ± 1 deg [23]. While this is sufficient for most SPHERES operations and may be possible for docking, it cannot provide a high confidence of successful, repeatable docking using the SPHERES UDP.

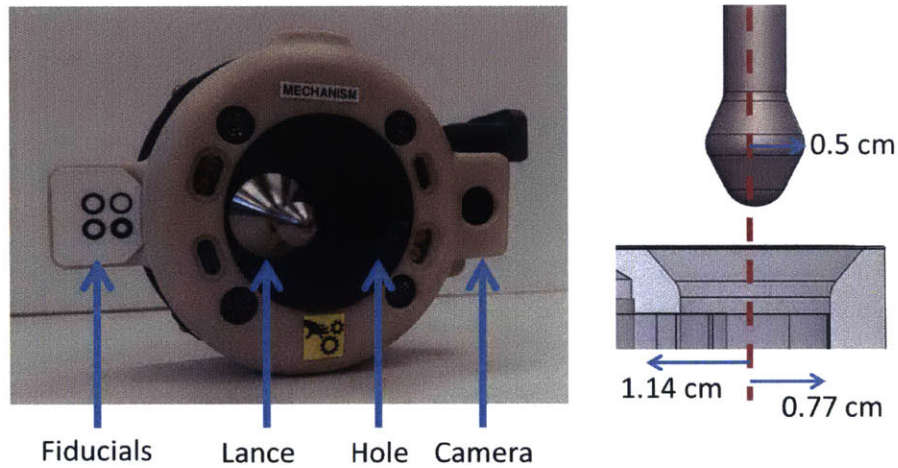


Figure 4-2: The SPHERES Universal Docking Port (left) and Necessary Docking Tolerances (right)

Consider Figure 4-2. This graphic shows the docking tolerances necessary for the lance to enter the hole. The acceptable linear and angular errors are borderline what the SPHERES global metrology provides. To be more precise, the linear and angular constraints are coupled together and follow the relationship.

$$e_{\theta} = \sin^{-1} \left(\frac{d/2 - e_l}{L} \right) \quad (4.1)$$

Where e_{θ} is the angular error, e_l is the linear error, d is the diameter of the docking hole, and L is the distance from the center of mass of the satellite to the docking lance. As described in Chapter 3, the SPHERES UDP design includes a capture cone to help guide the

satellites together. Figure 4-3 shows that, at a minimum, the relative sensor must provide accuracies of less than 1 cm (linear) and less than 2 degrees (angular) in order to have a chance of a successful dock. It should be recognized that these accuracies are necessary just before the satellites initiate contact. It will be shown (Section 4.2) that using an optical relative sensor, the accuracy increases as the agent separation distance decreases.

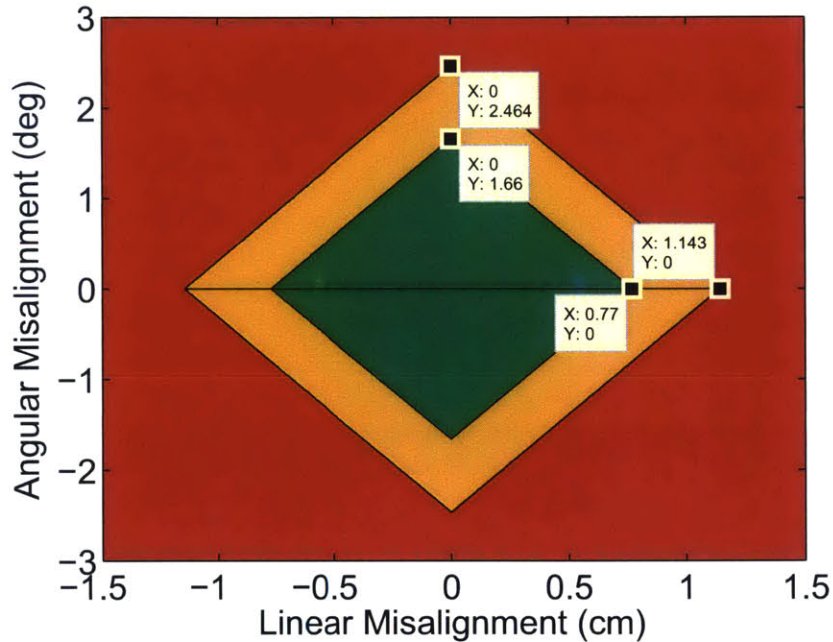


Figure 4-3: The Linear and Angular Error Constraints Needed to Achieve Docking. Green Is Within the Hole. Orange Is Within the Capture Cone. Red Is Outside The Capture Cone.

It is important to note that there is a marked difference between *sensing* accuracy and *control* accuracy. A satellite that cannot reliably control to less than 1 cm would be unable to achieve docking even with a perfect sensor. Similarly, a satellite that cannot reliably sense to less than 1 cm would be unable to achieve docking even with perfect actuators. Thus, the sensor proposed herein provides necessary, but not sufficient, conditions for satellite docking.

Table 4.1: Consolidated Requirements Matrix for the UDP Sensor

Parameter	Requirement	Instantiation
-----------	-------------	---------------

Continued on next page

Continued from previous page

Size	The docking port sensor shall be small enough to be integrable within the docking port cover and less than 5% of the docking port mass.	The selected docking port sensor meets these physical constraints.
Compatibility	The docking port sensor shall be operated over the VERTIGO expansion port. This requires an operating power no more than 5W at 5V, a communication protocol of USB 2.0 or Ethernet, and Linux drivers.	The selected docking port sensor is a 2.5W camera that uses USB 2.0 with Linux drivers.
Linear/ Angular Sensing Accuracy	The docking port sensor must be capable of meeting the linear and angular requirements specified in 4-3.	After solving the exterior orientation problem, this requirement was shown to be met (Section 4.2).
Traceability	The docking port shall include a flight-traceable relative sensor that can applied to spacecraft external to the ISS (no ultrasonics).	The selected docking port sensor is an optical camera, as would be used in the vacuum of space.

4.1.2 Instantiation

The selection of the camera was a fairly straightforward process. The uEye XS camera made by IDS Imaging was found to meet all of the requirements, in addition to providing a wide range of software configurable specifications. Table 4.2 summarizes many of these characteristics. The camera uses the same driver package and API as the VERTIGO goggles which significantly decreased development and integration time. Just like the VERTIGO Goggles, a CMOS camera was selected rather than a CCD due to their inherent resistance to the shader effect under high contrast lighting that may occur on orbit [7]. However, as noted in Section 4.2.4, the CMOS rolling shutter introduces additional complexities. The XS camera also possesses a reconfigurable frame-rate which is useful for maximizing the estimation rate based on available processing power.



Figure 4-4: The Selected COTS Camera for Integration into the SPHERES UDP.

Table 4.2: Specifications of the uEye XS Camera

Sensor	5MP CMOS
Shutter	Rolling
Resolution and Frame Rate	15 fps @ 2592x1944 pixels 30 fps @ 640x480 pixels
Focus	Autofocus from 10 cm to inf
Power Consumption	2.5W
Size	26.5 mm x 23.0 mm x 21.5 mm
Mass	12 g
Interface	USB 2.0 (Mini B)
Pixel Size	1.4 μ m

4.2 Measurements

In this section, we describe the measurement technique for a visual navigation algorithm used to achieve satellite docking on-orbit. This design extends the work presented by Dr. Brent Tweddle in [7], and is applied to the specific SPHERES-UDP payload. The details of the measuring system are presented, including the fiducial marker design, the measurement estimation step, the calibration procedure and estimated accuracies. This information is subsequently used in Section 4.2.6 to properly model the system and evaluate the proposed estimator.

4.2.1 Fiducial Markers

The proposed relative state estimation algorithm uses fiducial targets of known geometry to acquire measurements. Traditionally, these markers will consist of easily identifiable edges, lines or points as reference. High contrast shapes can offer some of the most easily detectable components in a variety of lighting environments, especially space. The selection of concentric contrasting circles for the SPHERES UDP has been based on the success of Gatrell et al. [27] who developed point targets specifically designed for the lighting environments of space. The most important property of these concentric circles is that the centroids of the circles remain collocated under translations and rotations. Moreover, the area ratio of the circles remains constant under translation and rotation. This makes unique identification of markers significantly simpler.

With this knowledge, four sets of concentric circles were chosen as the fiducial targets for the SPHERES UDP relative sensor. Fischler and Bolles [28] have shown that given four points on a plane, the exterior orientation problem can be solved with no ambiguities. As described in Section 4.2.3, the solution to the exterior orientation problem provides the relative position and orientation of the target spacecraft - these are the measurements provided by the UDP sensor.

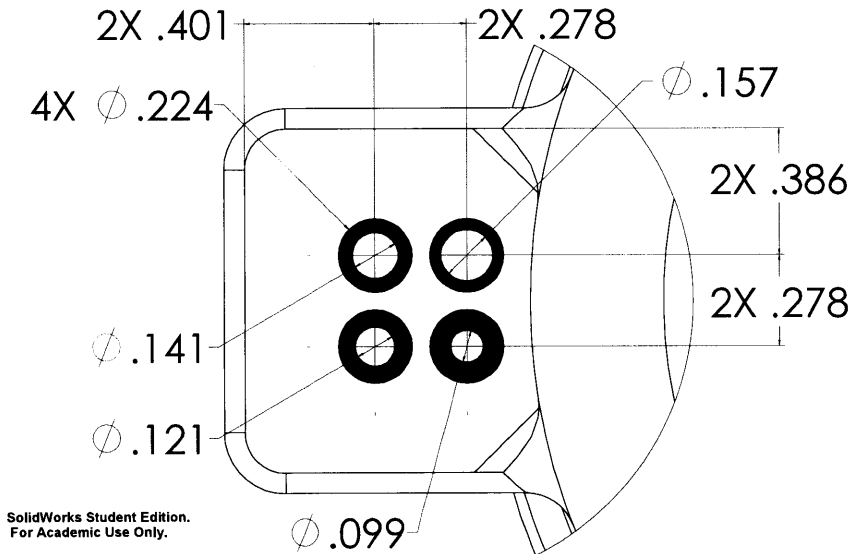


Figure 4-5: The Selected Fiducial Markers for the SPHERES UDP.

The precise geometry of the fiducial markers on the SPHERES UDP is shown in Figure

4-5. The fiducials were designed to be large enough to be detected at far distances, and small enough to fit within the full field of view of the camera when the SPHERES are docked together. Moreover, the area ratios of the fiducials were chosen to be linearly related, to make blob differentiation as easy as possible (Figure 4-6).

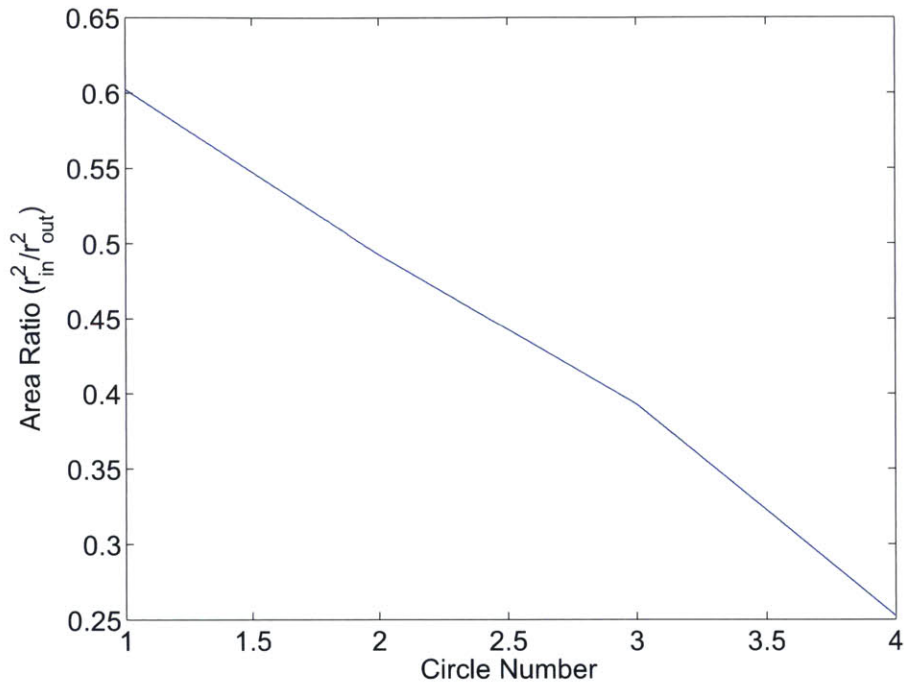


Figure 4-6: The Area Ratios of the Fiducial Markers are Spaced Linearly.

4.2.2 Marker Detection and Image Filtering

The measurements are obtained by visually identifying the fiducial markers on the target spacecraft, opposite the camera on the UDP. A multi-step algorithm is used to detect these targets, and is based on that proposed by Tweddle [7].

1. Image Masking
2. Image Segmentation using Adaptive Thresholding
3. Blob Identification and Classification
4. Blob Filtering
5. Search for Collocated Blobs

6. Solving the Perspective 4-Point Problem (P4P)

The algorithm has been designed to intelligently seek out the fiducial markers in the highest probability region first, relative to the previous measurement. This involves masking out a percentage of the image based on the target's distance away as computed during the previous iteration. This can significantly reduce computation time because roughly half of the pixels need not be processed. If a complete set of fiducials is not identified, no masking is performed in the subsequent step.

The image is then segmented into black and white using an adaptive thresholding technique from OpenCV. An adaptive window threshold is applied to every pixel in the image. While a static threshold value would not likely be robust to variations in lighting conditions, an adaptive window compensates by using a threshold value that is a function of the local intensity of light around each pixel. As a result, the output is a purely black and white image consisting of a set of unconnected components, or blobs.



Figure 4-7: The Result of Performing Adaptive Thresholding on a Grayscale Image and Its Inverse

An open-source library `cvBlobsLib` was used to identify and uniquely classify each region in the segmented image. From there, the blobs can be organized and searched in order to locate the fiducial markers of interest. Four cascaded filters have been tuned and are used to achieve robust fiducial tracking.

- a. **Collocation Filtering:** `cvBlobsLib` detects blobs using a falling edge gradient on image intensity. By matching blob centroids within a tolerance (approximately 2 pixels) from both the thresholded and inverted images, we throw out a majority of inoperative blobs.

- b. **Area Ratio Bounding:** The working assumption is that concentric circle area ratios are invariant to translation and rotation in the image plane. Thus, knowing the designed fiducial area ratio, we can reject a number of false positives. We note that there is an error tolerance needed to account for the quantization of pixel error (i.e. pixels are not fractionated).
- c. **Eccentricity Check:** We can check the circularity of the blobs by measuring the height and width of the bounding box. Knowing that the relative pose will be bounded within approximately 45 degrees in any axis means that we can impose an eccentricity check to reject inoperative blobs.
- d. **Perimeter-to-Area Ratio Comparison:** The innovative filter in eliminating false positives is the perimeter-to-area ratio check. It is shown in Sections 4.7.2 and 4.8 that the ultrasonic sensors closely resemble the size and shape of the fiducial markers. Fortunately, the ultrasonic sensors do not pass through the thresholding step cleanly. The ragged edges of the ultrasonic sensors help differentiate them from the clean contrast of the fiducial markers. Thus, the ultrasonic sensors have a much higher perimeter-to-area ratio and now have a high probability of being rejected as a false positive.

The final step in determining the relative pose of the target is to estimate a solution to the exterior orientation problem

4.2.3 Perspective Four Point Correspondence

At a high level, the exterior orientation problem solves for the translation vector and orientation quaternion from the camera image plane to a fixed coordinate in 3D space. This is also known as the hand-eye problem in machine vision and robotics [29]. The generalized perspective projection maps a 3D world coordinate (x, y, z) to the 2D image plane (u, v) . The exterior orientation problem is estimating the reverse, but the mapping is not one-to-one. Each pixel represented by (u, v) defines a straight line ray extending from the center of projection into the scene. Although each image plane point does not map to a unique point in the 3D world, known information about the scene geometry may be used to estimate the absolute world coordinates of a target.

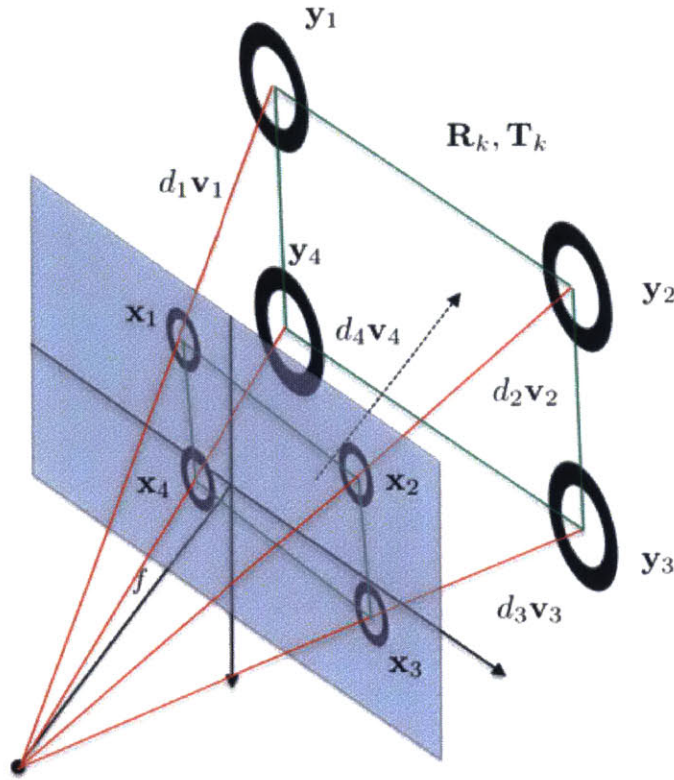


Figure 4-8: Four Point Correspondence Geometry of Haralick's Iterative Nonlinear Solution to Exterior Orientation [7].

Figure 4-8 illustrates the proposed measurement method including the camera geometry (pinhole model). The problem can be restated mathematically into solving for the rotation \mathbf{R} and translation \mathbf{T} between the two frames with:

$$\mathbf{x}_n = \mathbf{R}\mathbf{y}_n + \mathbf{T} \quad (4.2)$$

In this sense, \mathbf{R} is the rotation matrix that transfers a vector from world coordinates (the \mathbf{y} in Figure 4-8) to the image frame (the \mathbf{x}) and \mathbf{T} is the displacement vector between frame origins. For the purposes of relative navigation with spacecraft, it is common to represent orientation using quaternions rather than rotation matrices (refer to Appendix A.1.1). Not only is the number of elements reduced from nine (really six unique) to four, quaternions also do not suffer from the infamous gimbal lock. The (non-unique) quaternion can be found by numerically solving the system of equations provided by the direction-cosine

matrix to quaternion relationship (Equation 4.3).

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \quad (4.3)$$

4.2.4 Camera Calibration

The process of calibrating a camera to determine the precise internal geometry is known as solving the *interior* orientation problem. Knowledge of these camera parameters is essential in obtaining accurate position and orientation solutions during the perspective four point correspondence. The four elements of interest are:

1. **Camera Constant:** This is also known as the focal length and is the distance to the image plane from the center of projection.
2. **Principal Point:** This defines the origin of the image plane coordinate system.
3. **Lens Distortion Coefficients:** These nonlinear terms correct for the ‘fish-eye’ lens effect which can be attributed to optical imperfections in the camera.
4. **Scale Factors:** This correction factor applied to pixel rows and columns would be needed if the pixels are non-square.

The principal point (or camera center, c_x, c_y) and focal length (f_x, f_y) make up the intrinsic “camera matrix” (Equation 4.4). In projective geometry, this linear transformation from world coordinates into the projective plane is called a homography, which includes the intrinsic matrix (left) and extrinsic rotation plus translation matrix (right).

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (4.4)$$

Every camera model, indeed every serial number, will have a unique set of intrinsic calibration parameters. The calibration procedure is fairly straightforward for traditional robotics applications where the user has direct access to the payload. A calibration target

of known geometry (usually a checkerboard) can be sampled by the imager and an iterative, converging solution can be found. The more images that are sampled, the higher confidence in the calibration. Figure 4-9 shows the calibration process to be used on the UDP camera sensor during ISS operations. The astronaut positions the calibration target in specific regions of the field of view, collecting a large number of data points for a complete calibration.

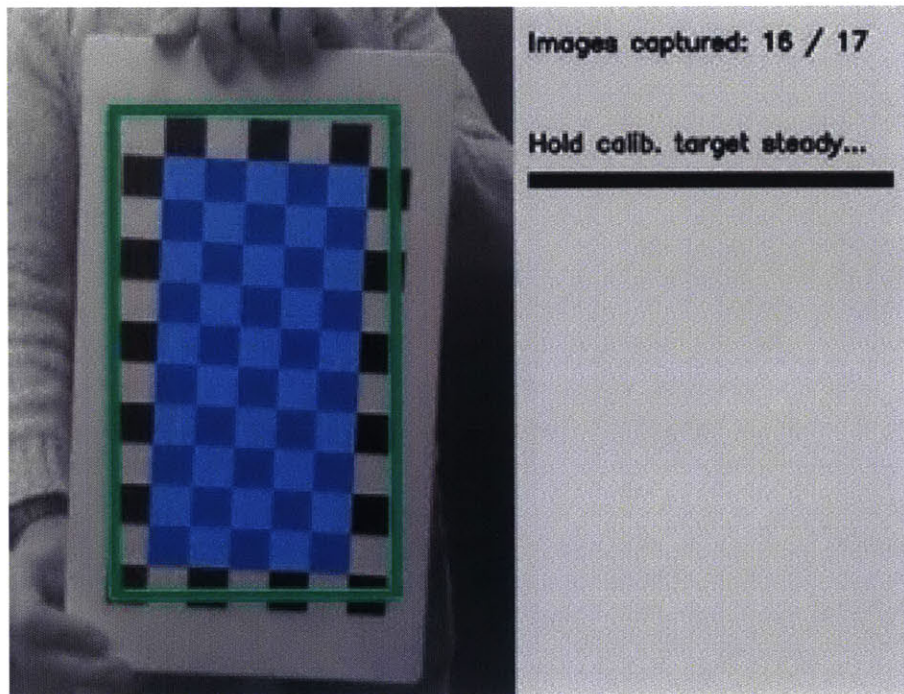


Figure 4-9: The Camera Calibration Process Used for the UDP Camera

One particular nuance with the selected sensor is that the UDP camera employs a rolling shutter, rather than a global shutter. The vast majority of embedded imagers utilize rolling shutters on account of their small size, simplicity and capacity for high frame rates. However, the CMOS sensor inside of a rolling shutter records the light exposure one row at a time. For fast moving objects, this can introduce unwanted artifacts including wobble, skew, smear, partial exposure and spatial and temporal aliasing. Global shutter addresses these issues by recording every pixel in a frame simultaneously. However, these sensors, which are typically CCDs, have an effective frame rate of about half that of rolling shutters and are also significantly larger in size.

The consequence of having a rolling shutter for camera calibration is shown graphically in Figure 4-10. An ideal calibration will have small and near uniform error between the set of images. This implies that each of the images provides independent information that improves the overall calibration estimate. Instead, if the calibration target is moving during image capture, the blurred checkerboard can lead to a poor and inconsistent intrinsic solution.

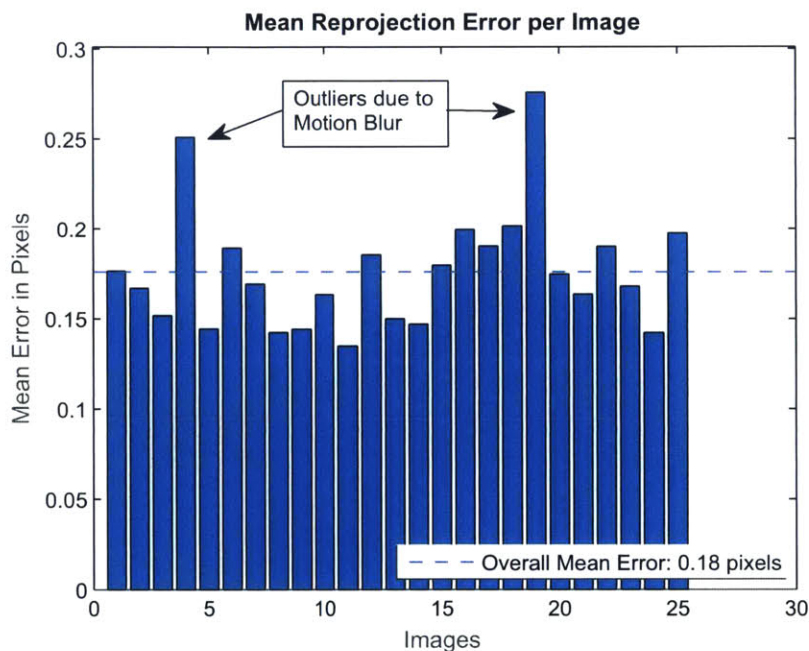


Figure 4-10: Uncorrected Mean Reprojection Error Anomalies Due to Rolling Shutter

After all samples of the calibration target are acquired, a least squares minimization is applied to obtain a mean intrinsic solution. Using the estimated calibration parameters, the reprojection error is calculated as the square of the Euclidian difference between an original 3D point (projected onto the image frame) and a triangulated estimate (reprojected onto the image frame) using the calibration solution. Figure 4-10 motivates the need to discard outliers on account of motion blur. When this is done, the range and variance of the intrinsic camera matrix is greatly reduced (Figure 4-11).

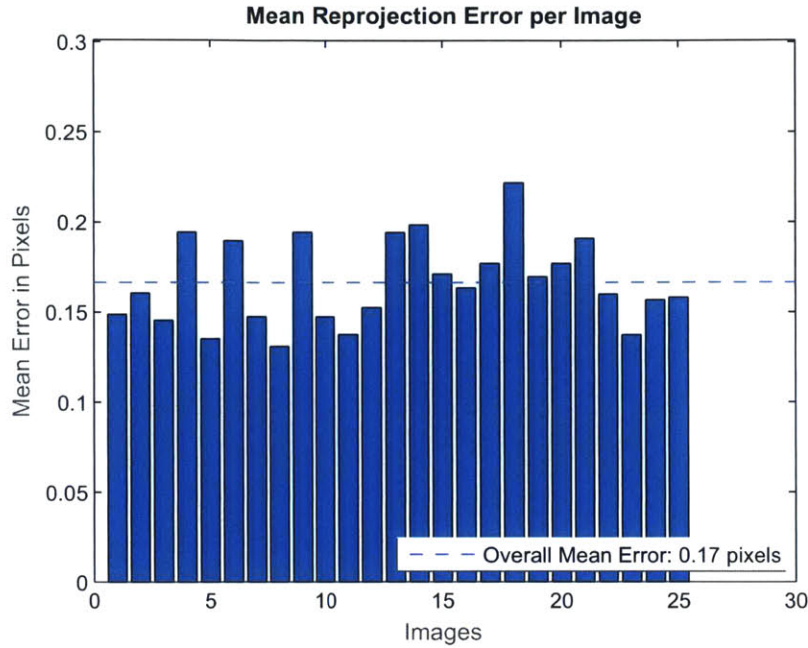


Figure 4-11: Corrected Mean Reprojection Error Anomalies of a Rolling Shutter Camera

The SPHERES testbed is fortunate in that astronaut assistance is available for camera calibration. Authentic spacecraft rendezvous scenarios must employ a different methodology for camera calibration. Two approaches can be taken. Either calibration can be performed once prior to launch, with the premise that launch vibrations and fluctuating thermal conditions have a negligible effect on the intrinsic matrix. Alternatively, a novel on-orbit calibration using stellar imagery or earth-originating targets could be used.

4.2.5 Camera Accuracy

The accuracy of the exterior orientation solution was judged experimentally on a high precision optical table at the MIT Space Systems Laboratory. The results from this testing indicate the expected accuracy and repeatability of the calibrated UDP sensor. This was achieved by fixing the camera and fiducials at known relative locations (Figure 4-12) and recording the estimated solution, using the unique calibration parameters for each camera. The side-to-side and depth accuracies are shown in Figures 4-13, 4-14, 4-15 (this coordinate frame is centered and oriented about the camera specifically).

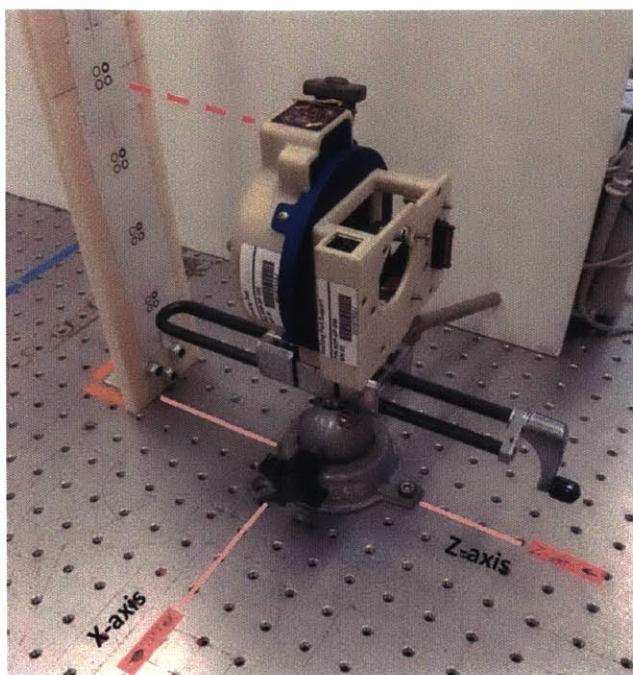


Figure 4-12: Optical Setup for Measuring Sensing Error

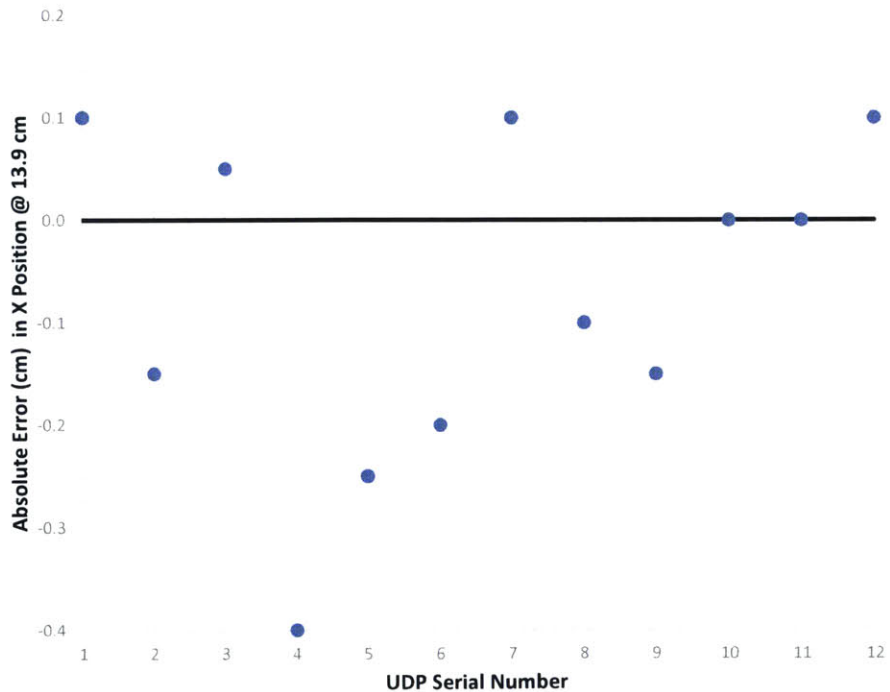


Figure 4-13: Absolute Error in the X-Direction from 13.9 cm Range

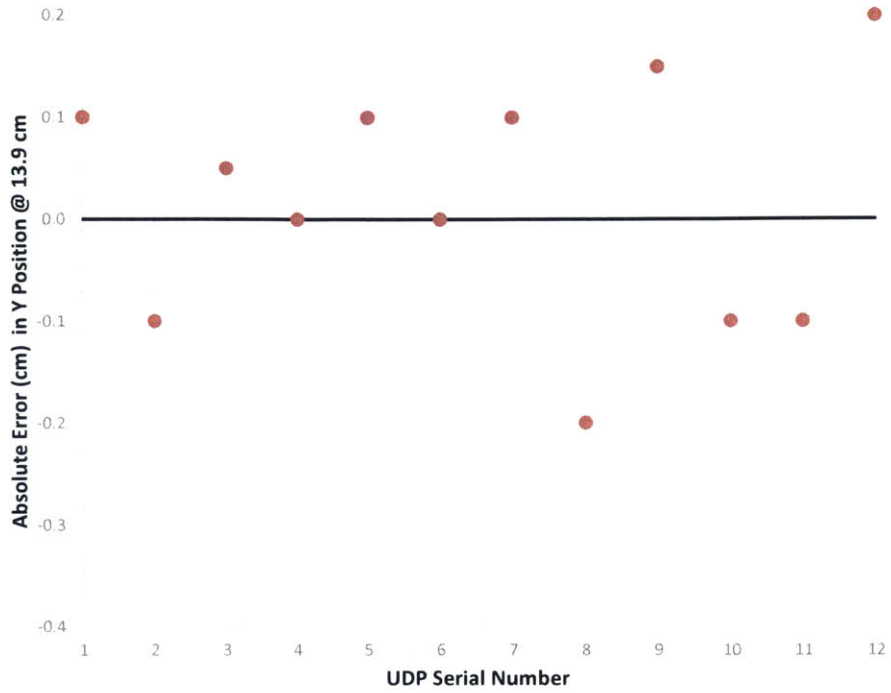


Figure 4-14: Absolute Error in the Y-Direction from 13.9 cm Range

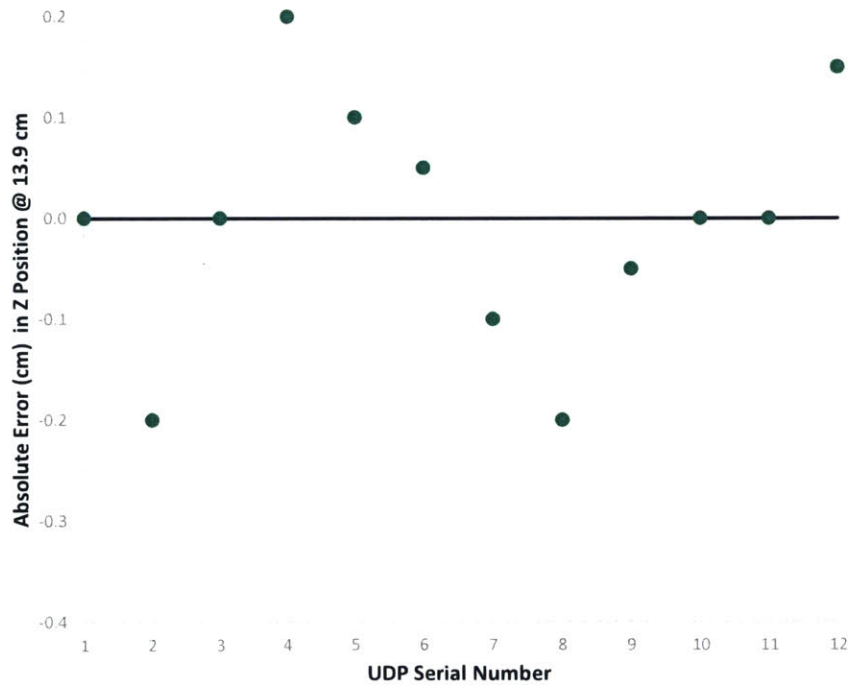


Figure 4-15: Absolute Depth Error in the Z-Direction from 13.9 cm Range

The critical insight is that these figures show absolute error at a given distance. This relative sensing error is attributed to an imperfect calibration. As the range to the target is reduced, the size of the fiducials in the image plane increases, effectively reducing observed absolute error. For a docking and servicing application, this is preferred because high precision sensing is only necessary at the closest ranges. It was shown through repeated testing that the error variance of a specific serial number was below the quantization error in the solution. Absolute quaternion accuracies of approximately 1 degree were also shown to be possible when the camera was properly calibrated.

4.2.6 Measurement Modeling

With the measurement technique sufficiently analyzed and understood, measurements may now be modeled accurately in simulation. This is necessary in order to observe realistic performance of the two proposed state estimators in simulation, which can provide an unambiguous truth state. The measurement inputs to the filter are the position and quaternion of the target relative to the observer’s camera frame. The observer is assumed to be inertially still and has the role of identifying and tracking the tumbling spacecraft prior to docking.

$$\mathbf{y}_k = \begin{bmatrix} \bar{\mathbf{r}}_k \\ \bar{\mathbf{q}}_k \end{bmatrix} \rightarrow \begin{bmatrix} \bar{\mathbf{r}}_k \\ \bar{\boldsymbol{\sigma}}_k \end{bmatrix} \quad (4.5)$$

As described in Section 4.4.1, the raw quaternion measurements are immediately converted into Modified Rodrigues Parameters for mathematical convenience. It is important to note that these measurement are always defined *from* one frame *to* another. Typically, the raw output measurements of the exterior orientation solution are defined as the translation and rotation *from* the target reference frame *to* the camera reference frame (Equation 4.2). In the scope of this chapter, the measurements obtained are defined as the displacement *from* the observer *to* the target and the quaternion transform *from* the target frame *to* the observer’s frame. This definition is maintained in order to keep the actual C code easier follow.

Random Measurement Noise Modeling

Three families of measurement noise have been implemented in order to achieve the most realistic performance of the filters. First, the traditional random noise has been included in the measurements. However, the noise on the measurements has been modeled as nonlinear/non-Gaussian. Due to non-linearities in both the dynamics and the measurements it will be shown that the Multiplicative Extended Kalman Filter (MEKF) will not be an unbiased estimator for this problem.

The derivation for the nonlinear noise measurements begins with assumption that the positions of the fiducial markers on the image plane suffer from small Gaussian noise perturbations. That is, the true pixel position of the concentric circles can be distributed normally with some variance that is a function of the atmospheric lighting, image blur, software thresholding and CMOS noise. However, we can conclude that there is a nonlinear transform from pixel noise to quaternion measurement noise in two of the three axes.

Figure 4-16 depicts the noise resulting from the twist case. w represents the pixel noise error which is applied in the image plane around the first Euler angle (x-axis twist). Using small angle approximations, the transformed noise in the first Euler angle (ϕ) is

$$\phi_w = \sin^{-1} \frac{w}{d} \approx \frac{w}{d} \quad (4.6)$$

Where d has been defined to be the distance between the center of the concentric circle (black dot) and the center of the fiducial plane. This is a linear transform to the first Euler angle.

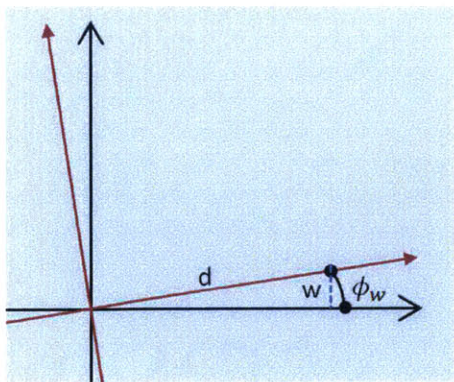


Figure 4-16: This graphic shows higher measurement sensitivity (lower noise) for an x-axis twist (blue is the image plane).

However, the two tilt degrees of freedom are much more sensitive to Euler angle noise. Although the pixel noise is the same, Figure 4-17 shows that the same pixel noise w results in a larger angle noise in the second and third Euler angles. A small angle approximation provides the following relation.

$$\cos \theta_w = \frac{d - w}{d} = 1 - \frac{w}{d} \quad (4.7)$$

$$\cos \theta_w \approx 1 - \frac{\theta_w^2}{2} = 1 - \frac{w}{d} \quad (4.8)$$

$$\theta_w, \psi_w \approx \sqrt{\frac{2w}{d}} \quad (4.9)$$

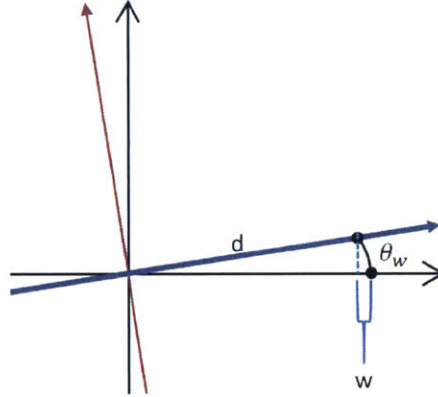


Figure 4-17: This graphic shows lower measurement sensitivity (higher noise) for a y- or z-axis tilt (blue is the image plane).

Since the relation w/d is smaller than 1, it is clear that the measurement noise on the second (and third) Euler angles are subject to a nonlinear amplifying transform. After the Euler angle noise has been generated, it is further transformed into an error quaternion as derived in Appendix A.1.1. This error quaternion is added to the measurement according to the multiplicative outer product.

$$\mathbf{q}_{meas}(k) = \delta \mathbf{q}_w \otimes \mathbf{q}_{truth}(k) \quad (4.10)$$

Thus, in modeling \mathbf{R} within the filters, we have weighted the measurement states according to a variable η^2 which is a measure of the confidence of the measurement. In

the hardware implementation, this is an output of the iterative solution to the exterior orientation problem and is an input for every filter step.

Type I Errors

In probability theory, Type I errors (or false positives) can lead to a large divergence of the measurement state. In the scope of this analysis, it is conceivable (in fact, occasionally expected) that the camera tracking algorithm falsely identifies a blob as a fiducial marker. This can result in a measurement that may be significantly offset from the current state estimate. In the simulation, this has been emulated by increasing the variance of the discrete random noise by a factor of 5 approximately 5% of the time.

Although the filters presented herein do not actively perform outlier rejection, I chose to model this in the system in order to make conclusions on the robustness of the filter.

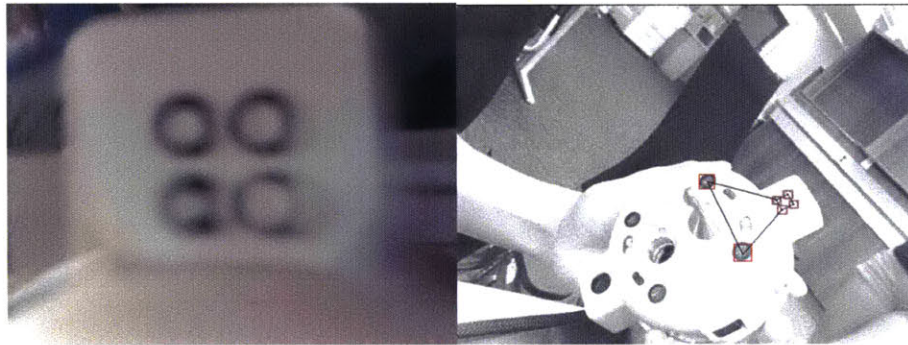


Figure 4-18: Examples of a False Negative Measurement (left) and a False Positive Measurement (right).

Type II Errors

In probability theory, Type II errors (or false negatives) arise when a measurement exists but the algorithm chooses not to use it. In the scope of this problem, it is conceivable that a measurement may not be acquired every time step. This can be caused by camera disturbances, motion blur, or external objects obscuring the markers. As a result, the filter propagates for longer time periods between measurements. This has been implemented in the simulation by dropping measurements approximately 5% of the time to get a more realistic performance from the filters.

4.3 Dynamics

A high level concept of operations that the estimation filters enable is shown in Figure 4-19. In this scenario, the target SPHERES B is in a stabilized spin about the camera axis so the fiducials are always in sight by SPHERES A.

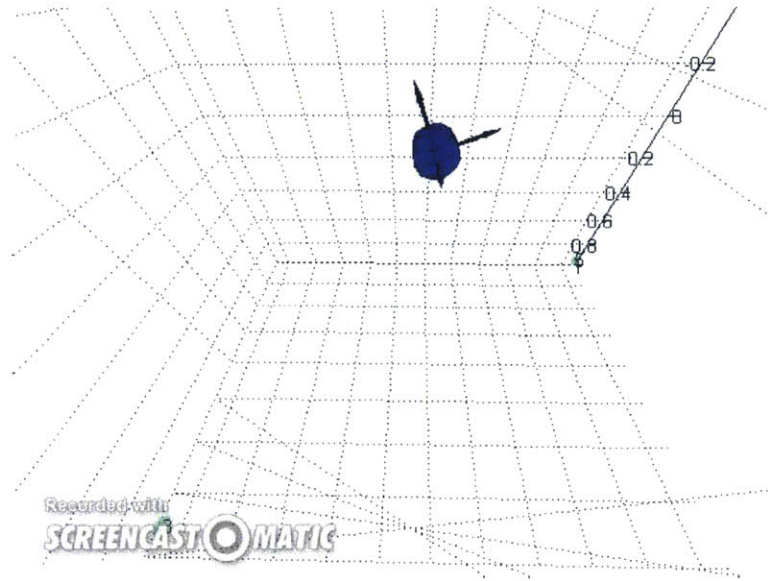


Figure 4-19: [Animation] The Proposed Concept of Operations for Relative Pose Estimation.

The state of a spacecraft in inertial space can be represented as a 13 element state vector consisting of position, velocity, attitude and rotation rate elements. For a six degree-of-freedom spacecraft, this representation is sufficient to model trajectories, disturbances, and control inputs without any ambiguity. In the scope of the chapter presented herein, the state to be estimated is a relative state. The position, velocity, attitude and rotation rates are defined relative to an inertially fixed observer.

Thus, \mathbf{r} is the x,y,z displacement vector from the observer to the target. \mathbf{v} is the x,y,z velocity of the target relative to the observer. \mathbf{q} is a quaternion transform that relates the orientation of the targets's frame to the observer's frame. The definition of a quaternion is reviewed in Appendix A.1.1. $\boldsymbol{\omega}$ defines the body-fixed rotation rates of the target relative to the inertial observer. The assembled state vector to be estimated is collected as \mathbf{x} .

$$\mathbf{r} = [r_x \quad r_y \quad r_z]^T \quad (4.11)$$

$$\mathbf{v} = [v_x \quad v_y \quad v_z]^T \quad (4.12)$$

$$\mathbf{q} = [q_1 \quad q_2 \quad q_3 \quad q_4]^T \quad (4.13)$$

$$\boldsymbol{\omega} = [\omega_x \quad \omega_y \quad \omega_z]^T \quad (4.14)$$

$$\boxed{\mathbf{x} = [\mathbf{r} \quad \mathbf{v} \quad \mathbf{q} \quad \boldsymbol{\omega}]^T} \quad (4.15)$$

The second order dynamics can be rewritten as a set of first order differential equations. The continuous time, stochastic, nonlinear dynamics are collected as follows. The process noise (\mathbf{W}_v and \mathbf{W}_ω) enter as acceleration inputs on $\dot{\mathbf{v}}$ and $\dot{\boldsymbol{\omega}}$. Although the forces and torques from the spacecraft thrusters are included in the continuous time dynamics, they are dropped to 0 in the scope of this project (no feed-forward estimation). Thus, we describe the free floating nonlinear dynamics of a tumbling spacecraft as [30] [31]

$$\dot{\mathbf{r}} = \mathbf{v} \quad (4.16)$$

$$\dot{\mathbf{v}} = \frac{1}{m}(\mathbf{W}_v + \mathbf{F}_T) \quad (4.17)$$

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} = \frac{1}{2} \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} \otimes \mathbf{q} \quad (4.18)$$

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \mathbf{W}_\omega + \mathbf{F}_R) \quad (4.19)$$

Where we have defined \mathbf{J} as the moment of inertia tensor along the geometric axes of the SPHERES. The definition of the inertia tensor is covered in Appendix A.1.2.

$$\mathbf{J} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \quad (4.20)$$

For convenience, we have introduced the outer-product matrix for quaternion kinematics as

$$\mathbf{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & \omega_1 \\ \omega_3 & 0 & -\omega_1 & \omega_2 \\ -\omega_2 & \omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \quad (4.21)$$

We note that this definition is the conjugate of the common form. This is on account of the convention (specific to this thesis) that the quaternion kinematics propagate the quaternion *from* the target frame *to* the inertially fixed observer's frame. This is the measurement returned by the Exterior Orientation problem (Equation 4.2). In addition, the cross product matrix $[\boldsymbol{\omega} \times]$ condenses the notation for subsequent state space representations.

$$[\boldsymbol{\omega} \times] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \quad (4.22)$$

4.3.1 Process Noise Modeling

The process noise enters the dynamics in the linear and angular rotational acceleration equations. \mathbf{W}_v and \mathbf{W}_ω are the disturbance accelerations that are applied to both free floating spacecraft. Since the state vector being modeled is a relative state, the process noise can be grouped onto the target spacecraft in the scope of relative estimation.

The process noise has been modeled as Gaussian white noise. For an orbiting spacecraft it can include micro-forces such as solar pressure, gravity gradients and atmospheric drag. However, in the instantiation of this estimation problem (i.e. SPHERES inside of the International Space Station) there are air drafts from circulating fans that provide the largest magnitude disturbances. To make the results the most interesting, this is what has been modeled.

$$E[\mathbf{W}_v(\tau_1)\mathbf{W}_v(\tau_2)^T] = \mathbf{W}_{c1}\delta(\tau_1 - \tau_2) \quad (4.23)$$

$$E[\mathbf{W}_\omega(\tau_1)\mathbf{W}_\omega(\tau_2)^T] = \mathbf{W}_{c2}\delta(\tau_1 - \tau_2) \quad (4.24)$$

$$E[\mathbf{W}_v(\tau_1)\mathbf{W}_\omega(\tau_2)^T] = \mathbf{0}_{3 \times 3} \quad (4.25)$$

4.4 Multiplicative Extended Kalman Filter

A Multiplicative Extended Kalman Filter (MEKF) has been implemented to estimate the full state dynamics of the spin stabilized satellite. Since measurements are only position and attitude, the information from the dynamic equations can be used to obtain a smoothed estimate of the full state (including velocities) that filters out measurement and process noise. The MEKF is reviewed herein.

The continuous, nonlinear dynamics (as defined in Section 4.3) can be expressed in general as a stochastic nonlinear differential equation and nonlinear measurement equation with the following notation.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) + \mathbf{B}_w \mathbf{w} \quad (4.26)$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) + \mathbf{v} \quad (4.27)$$

4.4.1 Re-parameterize

The discrete-time Extended Kalman Filter (EKF) has been introduced to account for nonlinearities in the dynamic model. The nonlinear dynamics (expressed as \mathbf{f}) can be linearized at every time step around the current best estimate. In this way, the same optimality principles employed for the traditional Kalman filter can be used for nonlinear systems to drive the estimation error to zero.

However, the EKF is not mathematically tuned to handle quaternion dynamics. Instead, an MEKF has been introduced in the literature to account for the fact that unit quaternions must maintain unit magnitude. Even small error quaternions must preserve unity (and should not be driven to zero). This motivates the re-parameterization of the error quaternion as a three-element set of Modified Rodrigues Parameters (MRP), defined exactly as follows.

$$\boldsymbol{\sigma} = \frac{4}{1 + q_4} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (4.28)$$

$$\dot{\boldsymbol{\sigma}} = \frac{d}{dt}(\boldsymbol{\sigma}) = \frac{4}{(1 + q_4)^2} \begin{bmatrix} \dot{q}_1(1 + q_4) - \dot{q}_4 q_1 \\ \dot{q}_2(1 + q_4) - \dot{q}_4 q_2 \\ \dot{q}_3(1 + q_4) - \dot{q}_4 q_3 \end{bmatrix} \quad (4.29)$$

In this way, the zero vector ($\boldsymbol{\sigma} = \mathbf{0}$) represents no error (the goal). The Modified Rodrigues Parameterization is valid for angle errors less than a full rotation. This is adequate since the error quaternion is reset to 0 after every measurement update.

The MRPs can be calculated from each measurement by using the quaternion product of the measurement ($\bar{\mathbf{q}}_k$) and the complex conjugate of the current state estimate ($\mathbf{q}_{ref_{k-1}}^*$), and applying Equation 4.29.

$$\bar{\boldsymbol{\sigma}}_{k-1} = \bar{\mathbf{q}}_k \otimes \mathbf{q}_{ref_{k-1}}^* \quad (4.30)$$

Also, at the end of each iteration, once the MRPs have been filtered, the current best estimate of the target's quaternion can be recovered by taking the quaternion product of the error quaternion and the previous estimate.

$$\mathbf{q}_k = \delta \mathbf{q}(\boldsymbol{\sigma}_k) \otimes \mathbf{q}_{ref_{k-1}} \quad (4.31)$$

4.4.2 Linearize

According to the MEKF, at each time step, the dynamics must be linearized around the current best estimate and subsequently discretized in order to apply the Kalman Filter

equations.

$$\mathbf{A}_k = \left. \frac{\partial \mathbf{f}(\mathbf{a})}{\partial \mathbf{x}} \right|_{\mathbf{a}=\hat{\mathbf{x}}_{k-1}} = \left[\begin{array}{ccc} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots \\ & & \ddots \end{array} \right]_{\mathbf{x}=\hat{\mathbf{x}}(t)} \quad (4.32)$$

$$\mathbf{C}_k = \left. \frac{\partial \mathbf{h}(\mathbf{a})}{\partial \mathbf{x}} \right|_{\mathbf{a}=\hat{\mathbf{x}}_{k-1}} \quad (4.33)$$

In the scope of this problem, the translational dynamics are trivially linear. However, the rotational dynamics (both the kinematics of the MRPs and Euler's equation) are nonlinear and have been linearized analytically. The kinematics of the MRPs can be written as a function of the angular velocities as follows according to [32] [33].

$$\dot{\boldsymbol{\sigma}} = \frac{1}{2} \left[\mathbf{I}_{3 \times 3} \left(\frac{1 - \boldsymbol{\sigma}^T \boldsymbol{\sigma}}{2} \right) + [\boldsymbol{\sigma} \times] + \boldsymbol{\sigma} \boldsymbol{\sigma}^T \right] \boldsymbol{\omega} \quad (4.34)$$

By inspection, it is clear that the linearized version of this can be reduced to

$$\dot{\boldsymbol{\sigma}} \approx \frac{1}{2} [\boldsymbol{\sigma} \times] \boldsymbol{\omega} + \frac{1}{4} \mathbf{I}_{3 \times 3} \boldsymbol{\omega} \quad (4.35)$$

Linearizing Euler's equation can also be done with some effort. First, it is easiest to expand the vector form of Euler's rotational dynamics into explicit equations. Refer to Appendix A.1.3 for details of the Jacobian. The linearized continuous A matrix evaluated at the best estimate is called \mathbf{A}_ω .

$$\dot{\boldsymbol{\omega}} = \mathbf{J}^{-1} \begin{bmatrix} \omega_2(I_{zz}\omega_3 - I_{xy}\omega_1 - I_{yz}\omega_2) - \omega_3(I_{yy}\omega_2 - I_{xy}\omega_1 - I_{yz}\omega_3) \\ \omega_3(I_{xx}\omega_1 - I_{xy}\omega_2 - I_{xz}\omega_3) - \omega_1(I_{zz}\omega_3 - I_{xz}\omega_1 - I_{yz}\omega_2) \\ \omega_1(I_{yy}\omega_2 - I_{xy}\omega_1 - I_{yz}\omega_3) - \omega_2(I_{xx}\omega_1 - I_{xy}\omega_2 - I_{xz}\omega_3) \end{bmatrix} \quad (4.36)$$

$$\dot{\boldsymbol{\omega}} \approx \mathbf{A}_\omega \boldsymbol{\omega} \quad (4.37)$$

The linearized continuous dynamics can be assembled as

$$\dot{\mathbf{x}} = \mathbf{A}_k \mathbf{x} + \mathbf{B}_w \mathbf{w} \quad (4.38)$$

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{\boldsymbol{\sigma}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \frac{1}{2}[\boldsymbol{\omega} \times] & \frac{1}{4}\mathbf{I}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{A}_\omega \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \\ \boldsymbol{\sigma} \\ \boldsymbol{\omega} \end{bmatrix} \quad (4.39)$$

$$+ \begin{bmatrix} \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \frac{1}{m}\mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{J}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{w}_v \\ \mathbf{w}_\omega \end{bmatrix} \quad (4.40)$$

To apply the discrete MEKF equations, we must discretize the dynamics to obtain \mathbf{A}_d and \mathbf{W}_d .

$$\mathbf{x}_k = e^{\mathbf{A}\Delta t} \mathbf{x}_{k-1} + \int_0^{\Delta t} e^{\mathbf{A}\tau} \mathbf{B}_w \mathbf{w} d\tau \quad (4.41)$$

$$\mathbf{x}_k = \mathbf{A}_d \mathbf{x}_{k-1} + \mathbf{w}_k \quad (4.42)$$

This can be achieved numerically using the Hamiltonian matrix, \mathbf{S} at each time step.

$$\mathbf{S} = \begin{bmatrix} -\mathbf{A}_k & \mathbf{B}_w \mathbf{W}_c \mathbf{B}_w \\ \mathbf{0}_{12 \times 12} & \mathbf{A}_k^T \end{bmatrix} \quad (4.43)$$

$$\mathbf{Z} = e^{\mathbf{S}\Delta t} = \begin{bmatrix} \mathbf{Z}_{11} & \mathbf{Z}_{12} \\ \mathbf{0}_{12 \times 12} & \mathbf{Z}_{22} \end{bmatrix} \quad (4.44)$$

$$\mathbf{A}_d = \mathbf{Z}_{22}^T \quad (4.45)$$

$$\mathbf{W}_d = \mathbf{Z}_{22}^T \mathbf{Z}_{12} \quad (4.46)$$

4.4.3 Propagate

During each iteration, the state must be propagated from the previous estimate. Euler integration of the nonlinear differential equation provides the state, and the

propagated covariance can also be found.

$$\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + \Delta t \cdot \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}) \quad (4.47)$$

$$\mathbf{Q}_{k|k-1} = \mathbf{A}_d \mathbf{Q}_{k-1|k-1} \mathbf{A}_d^T + \mathbf{W}_d \quad (4.48)$$

Again, we note the key assumptions about the driving noise \mathbf{W}_d , modeled as white.

$$E[\mathbf{w}_k] = 0 \quad \forall k \quad (4.49)$$

$$E[\mathbf{w}_{k_1} \mathbf{w}_{k_2}^T] = W_{k_1} \Delta(k_1 - k_2) \quad (4.50)$$

Where

$$\Delta(k) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \quad (4.51)$$

4.4.4 Measurement Update

The measurement update equations can use the position and quaternion solutions from the exterior orientation problem. The measurement noise modeling is derived in Section 4.2.6.

$$\mathbf{y}_k = \begin{bmatrix} \bar{\mathbf{r}}_k \\ \bar{\boldsymbol{\sigma}}_k \end{bmatrix} = \mathbf{C} \mathbf{x}_k + \mathbf{v}_k \quad (4.52)$$

$$= \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{0}_{3 \times 3} & \mathbf{I}_{3 \times 3} & \mathbf{0}_{3 \times 3} \end{bmatrix} \begin{bmatrix} \mathbf{r}_k \\ \mathbf{v}_k \\ \boldsymbol{\sigma}_k \\ \boldsymbol{\omega}_k \end{bmatrix} + \begin{bmatrix} \mathbf{v}_r \\ \mathbf{v}_\sigma \end{bmatrix} \quad (4.53)$$

With this, we are now able to compute successively the Kalman gain, \mathbf{L}_k and the updated state and covariance estimates.

$$\mathbf{L}_k = \mathbf{Q}_{k|k-1} \mathbf{C}_d^T [\mathbf{C}_d \mathbf{Q}_{k|k-1} \mathbf{C}_d^T \mathbf{R}_k]^{-1} \quad (4.54)$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{L}_k (\mathbf{y}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1})) \quad (4.55)$$

$$\mathbf{Q}_{k|k} = (\mathbf{I} - \mathbf{L}_k \mathbf{C}_d) \mathbf{Q}_{k|k-1} \quad (4.56)$$

However, in implementing the above equations, numerical stability issues may occasionally occur when the covariance diverged due to large condition numbers. Thus, using the fact that the covariance is always positive symmetric definite, we can implement a two step correction consisting of a numerically robust version of the covariance update equation. This has been show to be less sensitive to arithmetic truncation, especially when R is small.

$$\mathbf{Q}_{k|k} = (\mathbf{I} - \mathbf{L}_k \mathbf{C}_k) \mathbf{Q}_{k|k-1} (\mathbf{I} - \mathbf{L}_k \mathbf{C}_k)^T + \mathbf{L}_k \mathbf{R}_k \mathbf{L}_k^T \quad (4.57)$$

$$\mathbf{Q} = \frac{1}{2}(\mathbf{Q} + \mathbf{Q}^T) \quad (4.58)$$

4.5 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) provides an alternative filtering technique compared with the MEKF. While the MEKF ignores the nonlinearities of the model, the UKF propagates a set of sample points through the nonlinear model, thereby obtaining a better characterization of the mean and covariance. My implementation of this filter is achieved using the same dynamics and MRPs presented in the previous section. The generalized UKF method is reviewed herein.

At each iteration, first, a set of $2n$ sigma points is generated about the current

best estimate.

$$\mathbf{x}_{k-1}^i = \hat{\mathbf{x}}_{k-1|k-1} \pm \tilde{\mathbf{x}}^i \quad (4.59)$$

$$\tilde{\mathbf{x}}^i = \sqrt{n\mathbf{Q}_{k-1|k-1}_i}, \quad i = 1 \dots n \quad (4.60)$$

Where $\sqrt{n\mathbf{Q}_i}$ is the i^{th} row of the square root matrix. Each sigma point is subsequently propagated through the discrete, nonlinear dynamic equations.

$$\hat{\mathbf{x}}_k^i = \mathbf{f}(\hat{\mathbf{x}}_{k-1}^i, \mathbf{u}_{k-1}, t_{k-1}) \quad (4.61)$$

The propagated sigma points $\hat{\mathbf{x}}_k^i$ can be used to obtain a better approximation of the propagated mean and covariance.

$$\hat{\mathbf{x}}_{k|k-1} = \frac{1}{2n} \sum_{i=1}^{2n} \hat{\mathbf{x}}_k^i \quad (4.62)$$

$$\mathbf{Q}_{k|k-1} = \frac{1}{2n} \sum_{i=1}^{2n} (\hat{\mathbf{x}}_k^i - \hat{\mathbf{x}}_{k|k-1})(\hat{\mathbf{x}}_k^i - \hat{\mathbf{x}}_{k|k-1})^T + \mathbf{w}_{k-1} \quad (4.63)$$

New sigma points can be generated to obtain a more accurate measurement prediction. In the scope of this project, the measurement update equation is linear and this step is trivially unnecessary.

$$\mathbf{x}_{k|k-1}^i = \hat{\mathbf{x}}_{k|k-1} \pm \tilde{\mathbf{x}}^i \quad (4.64)$$

$$\tilde{\mathbf{x}}^i = \sqrt{n\mathbf{Q}_{k|k-1}_i}, \quad i = 1 \dots n \quad (4.65)$$

$$\hat{\mathbf{y}}_k^i = \mathbf{h}(\hat{\mathbf{x}}_k^i, t_k) = \mathbf{C}\hat{\mathbf{x}}_k^i \quad (4.66)$$

$$\hat{\mathbf{y}}_k = \frac{1}{2n} \sum_{i=1}^{2n} \hat{\mathbf{y}}_k^i \quad (4.67)$$

The estimated covariance and cross-covariance can be obtained using the sample

variance equations.

$$\mathbf{Q}_{yy} = \frac{1}{2n} \sum_{i=1}^{2n} (\mathbf{y}_k^i - \mathbf{y}_k)(\mathbf{y}_k^i - \mathbf{y}_k)^T + \mathbf{R}_k \quad (4.68)$$

$$\mathbf{Q}_{xy} = \frac{1}{2n} \sum_{i=1}^{2n} (\mathbf{x}_k^i - \mathbf{x}_{k|k-1})(\mathbf{y}_k^i - \mathbf{y}_k)^T \quad (4.69)$$

Finally, we have the Kalman gain and update equations as follows.

$$\mathbf{L}_k = \mathbf{Q}_{xy} \mathbf{Q}_{yy}^{-1} \quad (4.70)$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{L}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k) \quad (4.71)$$

$$\mathbf{Q}_{k|k} = \mathbf{Q}_{k|k-1} - \mathbf{Q}_{xy} \mathbf{Q}_{yy}^{-1} \mathbf{Q}_{xy}^T \quad (4.72)$$

$$= \mathbf{Q}_{k|k-1} - \mathbf{L}_k \mathbf{Q}_{yy} \mathbf{L}_k^T \quad (4.73)$$

4.6 Validation in Simulation

The scenario studied is a spin stabilized case about the x-axis of the target SPHERES satellite. The initial conditions ensure that the SPHERES is aligned with the fiducials pointed roughly towards the camera and the products of inertia of the SPHERES ensure that the nonlinear dynamics are interesting.

$$\mathbf{r}_0 = [1 \quad 0 \quad 0]^T m \quad (4.74)$$

$$\mathbf{v}_0 = [-0.1 \quad 0 \quad 0]^T m/s \quad (4.75)$$

$$\mathbf{q}_0 = [0 \quad 0 \quad 1 \quad 0]^T \quad (4.76)$$

$$\boldsymbol{\omega}_0 = [0.1 \quad -0.02 \quad -0.0001]^T rad/s \quad (4.77)$$

$$\mathbf{x}_0 = [\mathbf{r}_0 \quad \mathbf{v}_0 \quad \mathbf{q}_0 \quad \boldsymbol{\omega}_0]^T \quad (4.78)$$

In the scenario considered, we have propagated for 20 seconds with a time step of 0.01. Longer times were also studied and the results were comparable. The noise

covariances, \mathbf{R} and \mathbf{W}_c , were also varied. In the presented solution, I have set

$$\mathbf{R} = \begin{bmatrix} 0.05\eta^2\mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \eta^2\mathbf{I}_{3\times 3} \end{bmatrix} \quad (4.79)$$

$$\mathbf{R}_k = \mathbf{R} \quad (4.80)$$

$$\mathbf{W}_c = 10^{-4} \begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & 0.002\mathbf{I}_{3\times 3} \end{bmatrix} \quad (4.81)$$

$$\mathbf{W}_k \approx \mathbf{W}_c\Delta t \quad (4.82)$$

Due to the conflicting units between states, some states have smaller or larger variances. The matrices were selected based on expected disturbances. We also note that \mathbf{R} is a function of η^2 , an confidence output of the exterior orientation problem and generally small (≈ 0.005).

4.6.1 Representative Convergence

Here we present representative convergence of the filter for all 13 states. It can be seen in Figure 4-20 that the noise is relatively white except for the Type I errors which are the obvious state outliers. The large turquoise bullet represents the initial guess of the state, which quickly converges to the truth state.

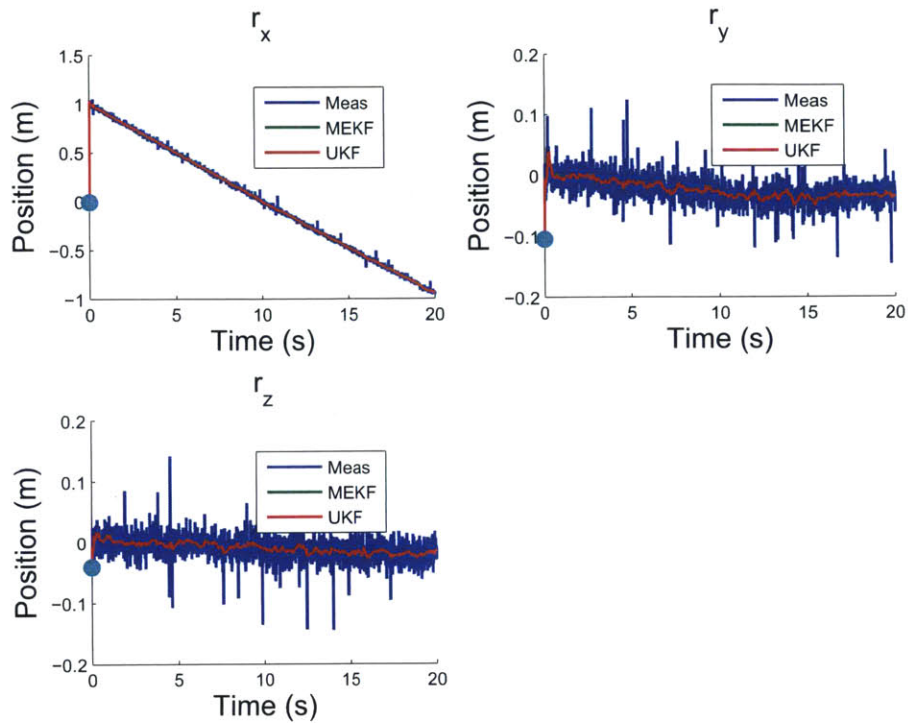


Figure 4-20: Representative tracking of position states with full noise modeling.

For the position and velocities, subject to process noise, the dynamics are linear and it is obvious that the MEKF and UKF behave similarly and very well. Their differences in pure linear estimation really can only be discerned through Montecarlo simulations.

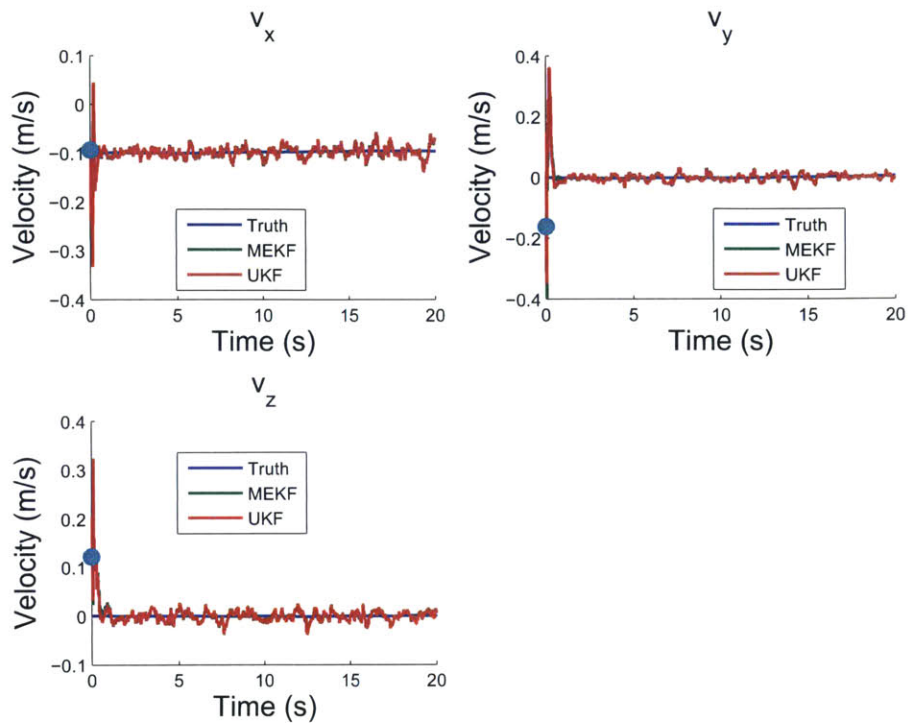


Figure 4-21: Representative tracking of velocity states with full noise modeling.

The merits of the Unscented Transform manifest themselves in the estimation of the nonlinear rotational dynamics with non-Gaussian noise. In Figure 4-22, it can be seen that the magnitude of the noise varies periodically. Since this plot is busy and small, the error vector is plotted in Figure 4-24.

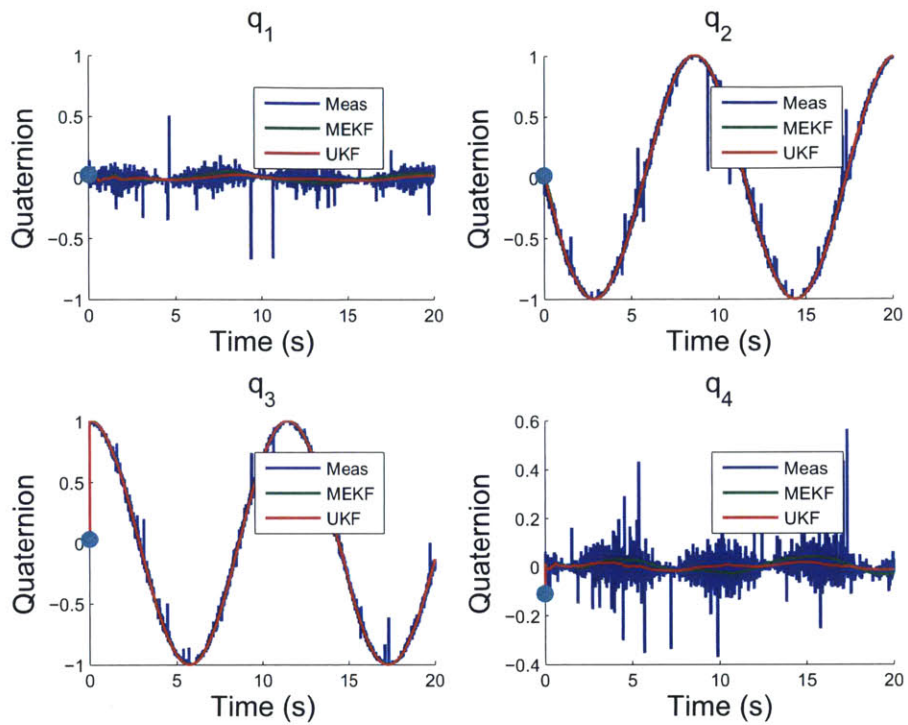


Figure 4-22: Representative tracking of quaternion states with full noise modeling.

It can be seen that the MEKF suffers from higher frequency, larger magnitude oscillations in the ω estimate compared to the UKF. Since it is spin-stabilized about the x-axis, the ω_x rate is non-zero at 1.1 rad/s.

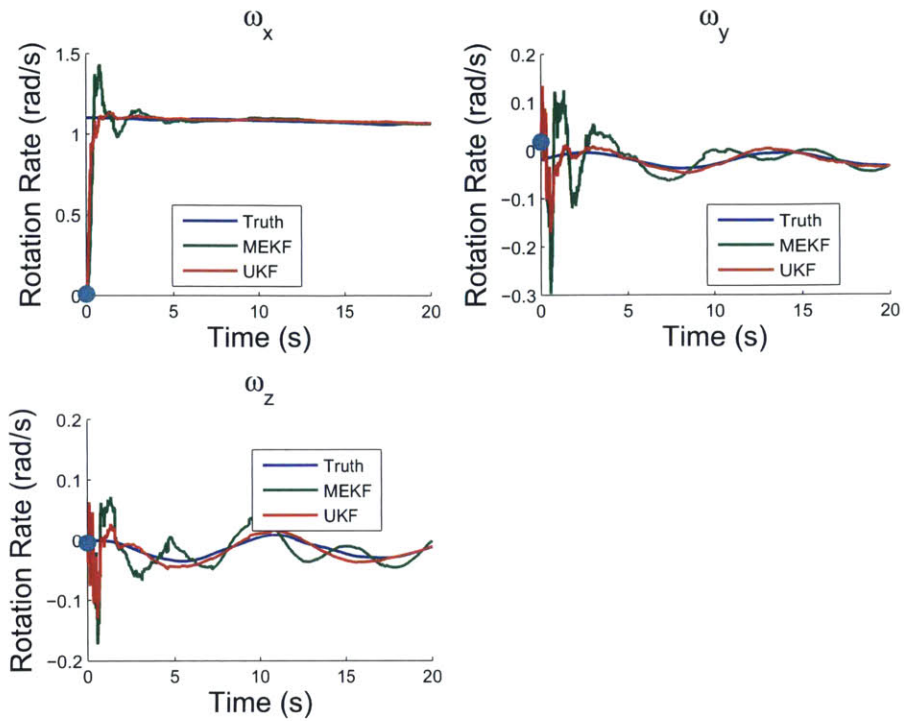


Figure 4-23: Representative tracking of angular velocity states with full noise modeling.

4.6.2 Representative Errors

In this section, we plot the estimation errors relative to the known truth states. The UKF performs better across the board in the nonlinear states. It is noted that the UKF does suffer from higher frequency errors in the initial few steps compared with the MEKF but this is eventually smoothed out for the better.

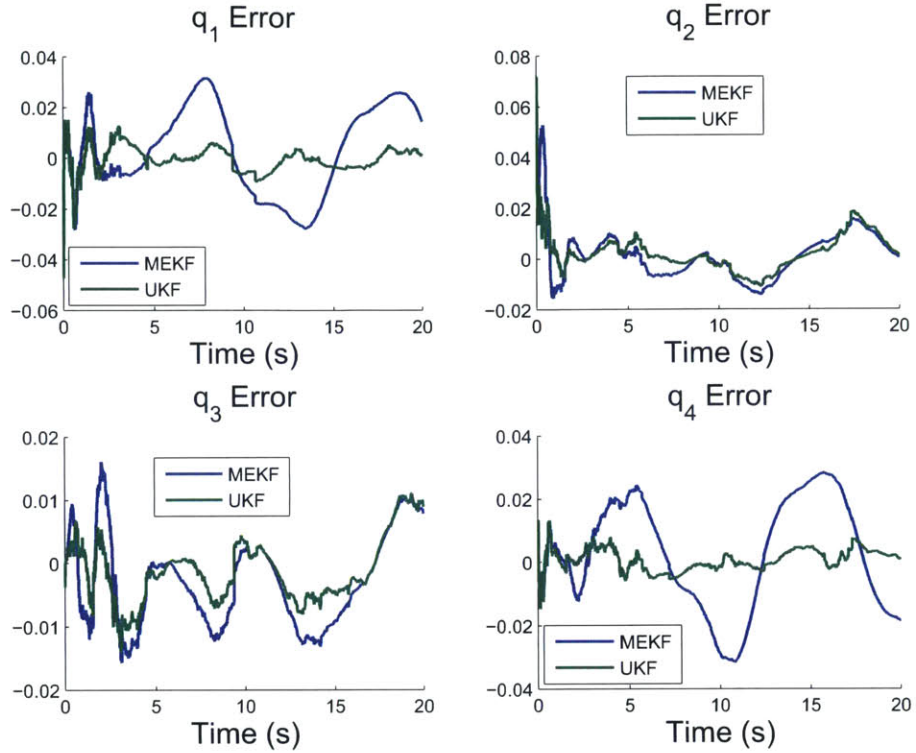


Figure 4-24: Representative quaternion error with full noise modeling.

The effects of the Type I errors get smoothed and integrated into the angular rate estimates. It can be seen that the estimates suffer latch ups (sharp discontinuities) when multiple Type I errors are observed successively (seen in Figure 4-22). However, it is clear that the UKF recovers much better than the MEKF from Type I errors.

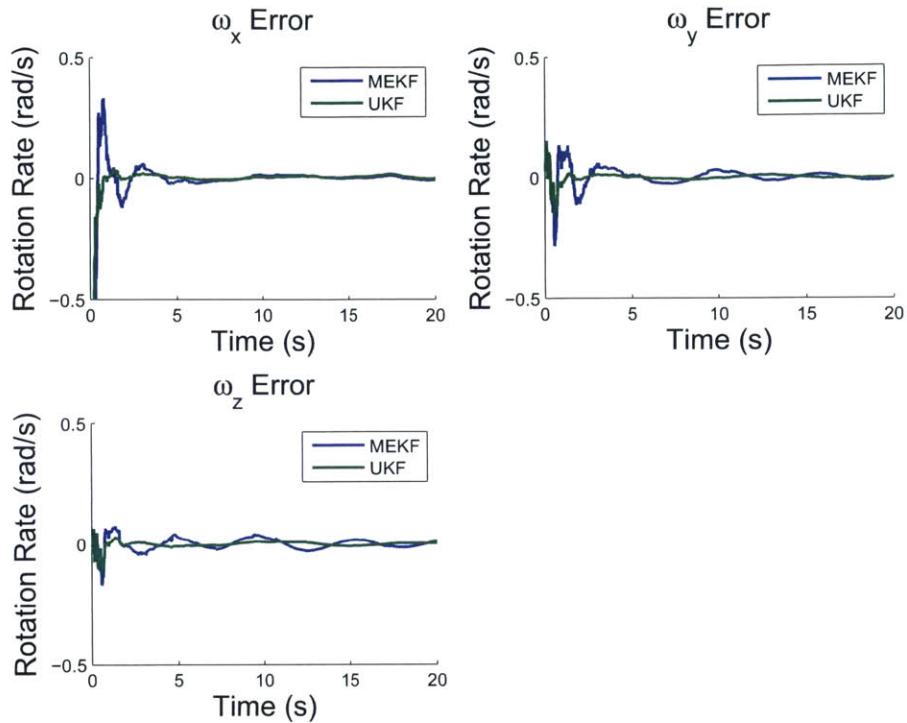


Figure 4-25: Representative angular velocity error with full noise modeling.

4.6.3 Modeling Variations

As an intellectual exercise, we have also modeled the system with varying degrees of noise to better understand the robustness of the controllers. First, we consider a simplified white noise case where the noise on the quaternion measurement is directly white (does not undergo the nonlinear transform described previously). In addition there are no Type I errors. In this case the MEKF and UKF perform nearly identically.

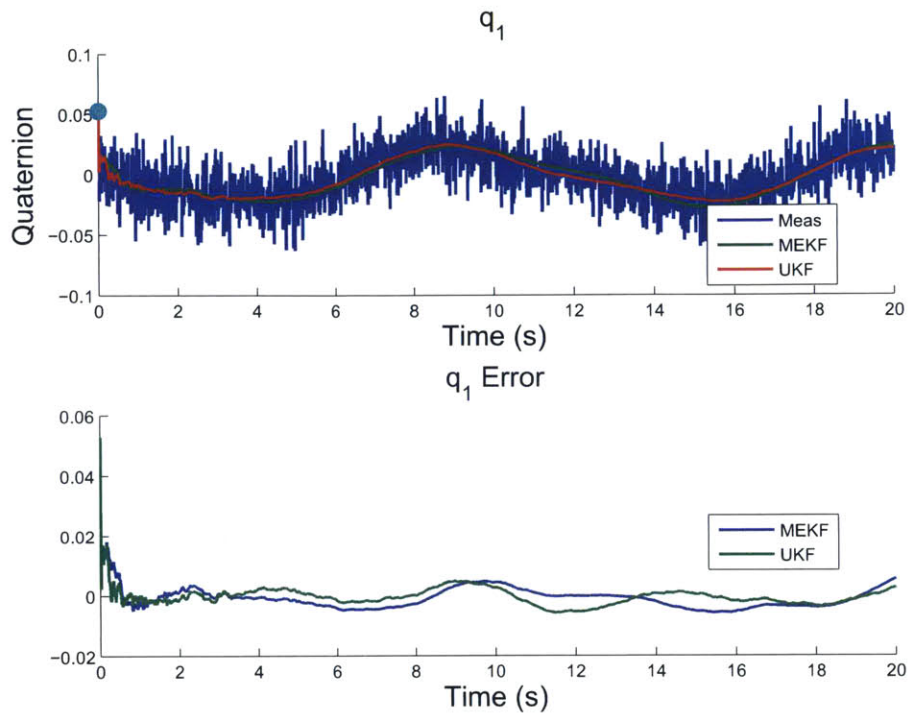


Figure 4-26: Representative angular velocity error with full noise modeling.

In a second noise variation investigation, we change the noise covariances within the filters by a factor of two relative to the true W and R . In a real world filter, the process noise and measurement noise intensities will not be known exactly. In fact, they may be different by a factor of 2 or more. We ran the MEKF and UKF filters and both demonstrated robustness to variation.

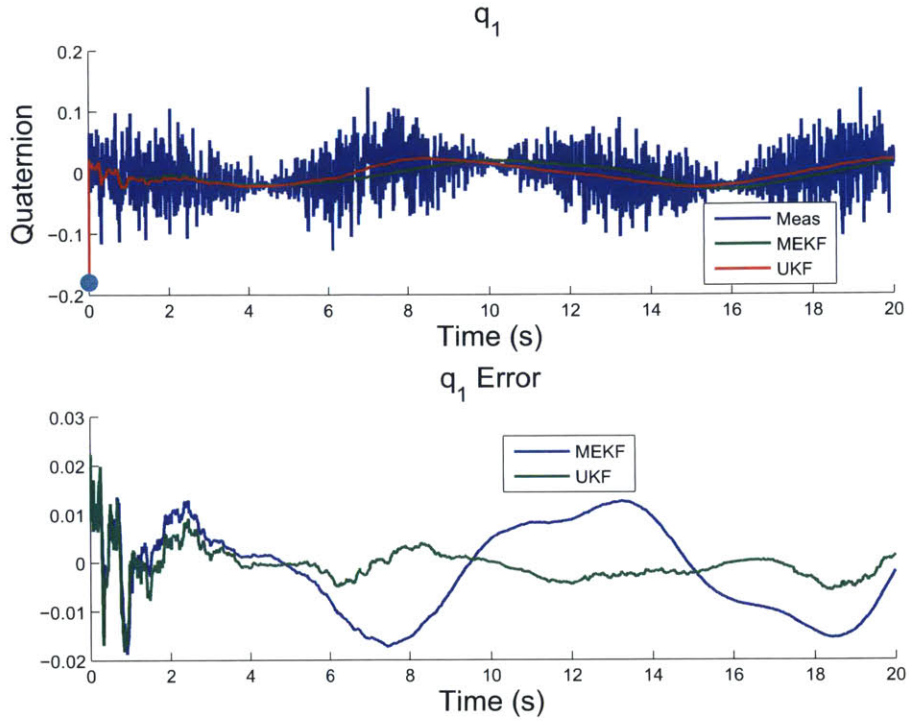


Figure 4-27: Representative angular velocity error with full noise modeling.

4.6.4 Montecarlo Simulation

Since a single filter run is based on stochastic noise, it is natural to perform a Montecarlo simulation in order to judge and compare the expected filter performance. To this end, we have simulated 100 scenarios with identical noise intensities. Using the results from each run, we can compute a metric called the Mean Squared Error (MSE). This can be calculated by summing over all time for each run for each filter. Since the system is subject to nonlinear dynamics and non-gaussian noise, the MEKF is no longer a non-biased estimate of the state, which manifests itself in the derivation of the MSE.

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{\mathbf{x}}_i - \mathbf{x}_i)^2 \quad (4.83)$$

$$MSE = Var(\hat{\mathbf{x}}) + (Bias(\hat{\mathbf{x}}, \mathbf{x}))^2 \quad (4.84)$$

However, the MSE is not the best performance parameter that can be used because its magnitude is a function of the base units of measurement. Thus, we introduce the normalized root mean squared deviation as follows.

$$NRMSD(\hat{\mathbf{x}}_i) = \frac{\sqrt{MSE(\hat{\mathbf{x}}_i)}}{\mathbf{x}_{max_i} - \mathbf{x}_{min_i}} \quad (4.85)$$

The NRMSD can be calculated for each state and averaged over all montecarlo runs. From the NRMSD (Figure 4-28), we can make conclusions about the performance of each filter.

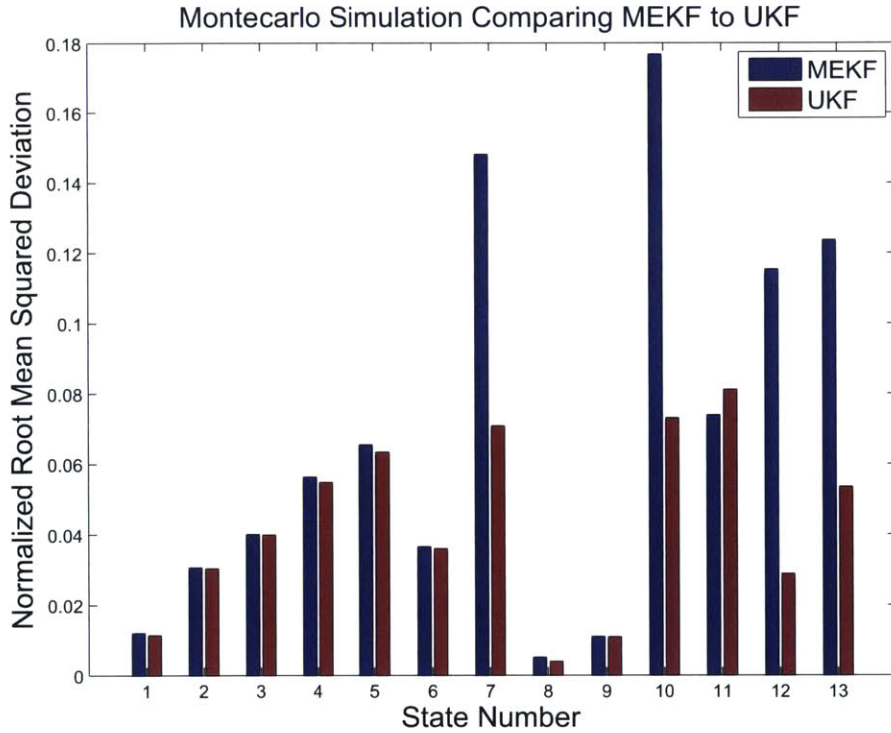


Figure 4-28: The non-dimensional performance of the MEKF and UKF filters (lower is better)

It is obvious that for the linear translational dynamics, the performance of the MEKF relative to the UKF is nearly identical. However, in the presence of nonlinearities and non-Gaussian noise, it becomes obvious that the UKF has significant advantages, especially in two of the quaternion states (q_1 and q_4) and two of the of the angular velocity rates (w_y and w_z which are nonzero precisely due to nonlinearities).

4.7 Validation in Three Degrees of Freedom

This section validates the performance of the filters, showcasing their performance in real-time on representative hardware in the SPHERES lab. To achieve this, we have used the Matlab Coder toolbox to auto-code the Matlab function into C++ files. The SPHERES VERTIGO processor runs C++ so it was simple enough to integrate the proposed MEKF and UKF filters within the SPHERES software framework. As shown in the Figure 4-29, the hardware demonstration involved floating the SPHERES on a 3 DoF air carriage and collecting state measurements during a flyby maneuver. Since on the ground we are restricted to 3 DoF, a spin stabilized scenario was infeasible to implement.

4.7.1 Ground Test Objectives

The ground testing campaign was performed with two principle objectives. This provided the basis for six degree-of-freedom testing on NASA's reduced gravity aircraft (Section 4.8) and future ISS testing.

- A. Demonstrate fiducial tracking with non-zero target velocity
- B. Confirm that the nonlinear UKF shows improved performance compared to the MEKF using actual measurements

4.7.2 Experimental Results

The camera was capable of acquiring a lock on the target spacecraft (Figure 4-29) during the pass. From this, the full state data was able to be estimated by both filters. Although we are missing a known "truth" state, we are able to verify through the video that the direction and order of magnitude of the position and velocity states was correctly being estimated. A more complete analysis should make a thorough comparison with the SPHERES estimator. The difference between MEKF and UKF is plotted in Figure 4-30 and it can be seen that they tend to converge over time (for the linear translational dynamics).



Figure 4-29: [Animation] Real-Time iducial Tracking (Target Is Locked)

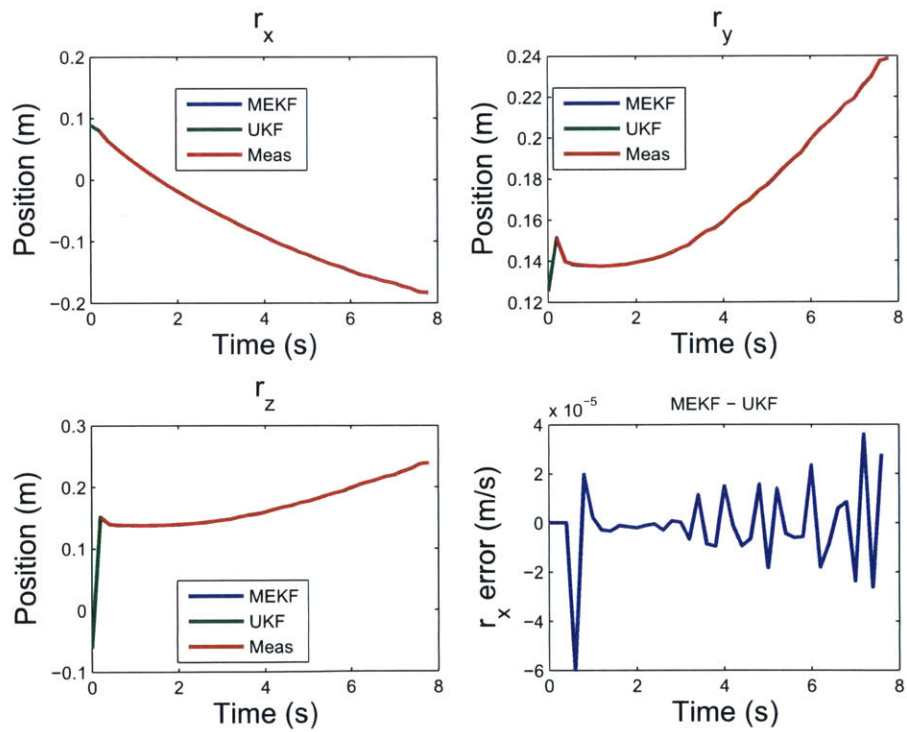


Figure 4-30: Quaternion estimate of the state in the SPHERES hardware demo

Even by visual inspection of the video capture, it is clear that the rotation rates are small and do not vary by more than 10% throughout the run (there are no non-conservative external forces or torques acting on the spacecraft). However, the MEKF estimator exhibits high estimate oscillations in comparison to the UKF until converging to a more reasonable solution.

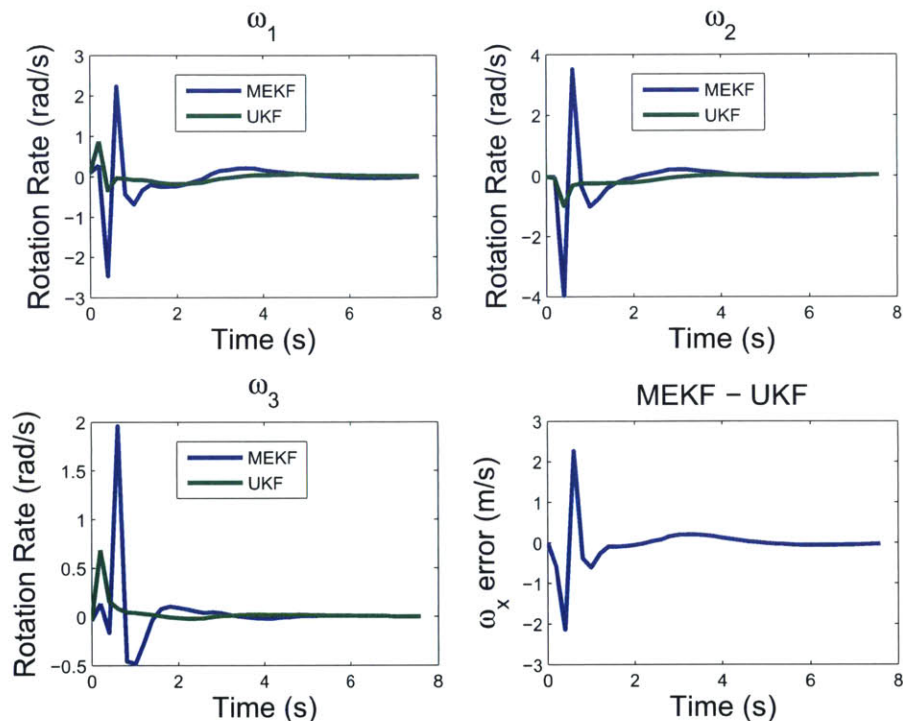


Figure 4-31: Angular rate estimate of the state in the SPHERES hardware demo

Finally, it is noted that the disturbances experienced in this hardware demo are on the same order of magnitude expected on the ISS, if not a bit larger than expected. The air carriages are not perfect isolators and the glass table is never perfectly balanced, which introduces process noises that we would not expect in zero gravity.

4.8 Validation in Six Degrees of Freedom

To increase confidence in the ISS test session performance of the UDP relative sensor, a parabolic flight campaign was necessary to bridge the gap between 3 DoF and 6 DoF sensing. While the sensor was tested successfully in 2D ground testing, only 3 of the 6 degrees of freedom were exercised.

4.8.1 RGA Objectives

Two primary objectives for the zero gravity flight were formulated to increase the technology readiness level of the UDP relative sensor.

- A. Fiducial identification of a free-floating target in a dynamically representative environment
- B. Fiducial identification of the target in a NASA lighting environment and at varying distances

The high precision relative tracking is necessary to ensure a high probability of successful docking. Free floating objects introduce dynamics into the system that cannot be replicated on the ground. Although our algorithm was tested repeatedly at MIT, its expected zero-g performance was unknown until after successful RGA testing. Moreover, the lighting conditions on the aircraft are rather similar to those aboard the ISS (overhead, low power, confined space) which increases our confidence in our camera calibration settings.

4.8.2 RGA Results

The key data and primary discoveries in the area of UDP relative sensing can be summarized in these three main points.

- The UDP system design was validated, namely camera location and operation, fiducial size and color

- A video data was set collected for tuning the fiducial identification and tracking algorithm post-flight
- Knowledge of the maximum and minimum range measurements for SPHERES as free floaters was obtained

Reduced gravity testing has provided the team with a large data set of free floating targets and representative relative velocities. Using the video captured during the zero gravity campaign, the team has been able to iteratively improve our tracking and filtering algorithm to improve robustness under representative disturbances.

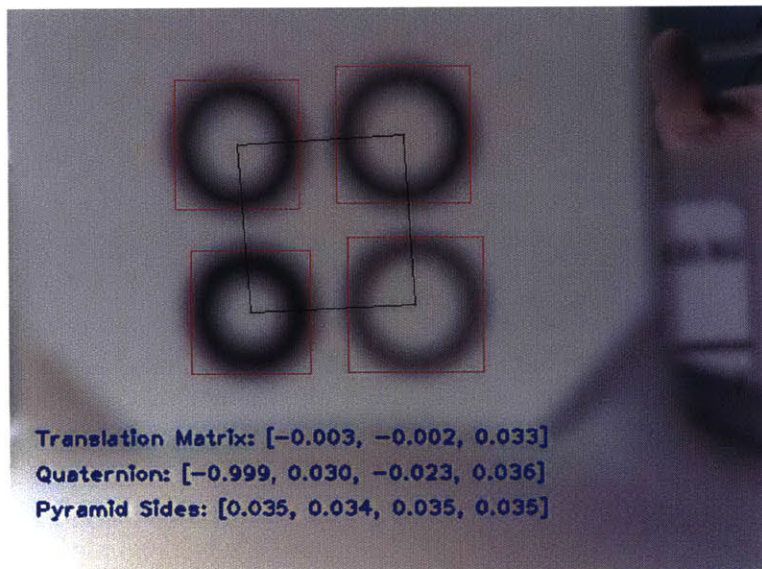


Figure 4-32: [Animation] Real-Time Fiducial Tracking During Undocking (Target Is Locked)

A successful tracking solution will return the best estimate of (1) the position vector to the target fiducials and (2) the relative quaternion between the camera frame and the target frame. Figure 4-32 shows an example of a successful tracking solution on the Reduced Gravity Aircraft during an undocking maneuver. As described in Section 4.2.2, a six step algorithm was used to obtain this tracking estimate. At least four key lessons were learned.

Failure to converge on a state estimate was typically the result of three phenomena. First, if one or more of the concentric circles is either out of the camera's field of view or obscured by some obstructing object/surface, the algorithm will not be able to deliver a solution. This cannot be solved easily in software using the current framework. Fortunately, RGA testing showed us that our current mechanical design is likely acceptable for nominal operations.

Second, no solution will be found if there is sufficient blurred motion in the image. Since the selected camera utilizes a rolling shutter, a certain amount of blur is expected. The blur affects the results principally in the adaptive thresholding step (step 3), which iterates over every pixel and decides to turn it either black or white. Using a large data set of RGA video, we have converged on a range of settings that maximizes the likelihood of finding a solution. Although the details are specific to our application, the results will be utilized and built upon during ISS test sessions.

Third, the occurrence of false positives or Type II false negatives can result in an incorrect solution or no solution at all. The presence of ultrasonic sensors arranged in a perfect square on the face of the UDP presents a challenge when the tracking algorithm is searching for four circles in a square arrangement. Even if one ultrasonic sensor is misinterpreted as a fiducial marker, the entire solution must be thrown out. Using the data sets obtained from RGA, we have eliminated false positives through the use of four cascaded filters (Section 4.2.2).

Finally, through RGA testing we have determined an approximate range for active relative sensing. We were able to detect target fiducials in the range between 2 cm and 1.1 meters. This suggests having a handoff between global and relative navigation systems around approximately 0.7 meters to ensure continuous spacecraft guidance.

4.9 Summary and Conclusions

In this chapter, we have investigated the performance of a spacecraft relative sensor as applied to satellite docking. A fiducial tracking algorithm was implemented and two strategic filtering methods were considered: a Multiplicative Extended Kalman

Filter (MEKF) and a Multiplicative Unscented Kalman Filter (UKF). Specifically, we have developed a tuned and calibrated sensing instrument, applied advanced filtering techniques to estimate the target state in the presence of measurement uncertainties, and validated the entire sensing system in a representative environment. After performing a thorough Montecarlo simulation, we were able to conclude that the UKF performed measurably better when estimating nonlinear states with non-Gaussian noise.

The results from this chapter have direct application on upcoming SPHERES test sessions on the ISS. Future work on this topic may address some of the following fields.

- *Actuator Modeling*: The external forces and torques generated by the satellites have been neglected in this analysis. In the future if this filter is used during satellite docking maneuvers, it would be preferred to feed forward the thruster firing commands in the filter.
- *Free Floating Observer*: The model presented herein assumes a static observer and dynamic target. However, for satellite-satellite docking, both agents are dynamic. It may make sense to separate the dynamics of both bodies for more accurate modelling.
- *Process Noise*: The process noise on the ISS is expected to be marginally different than the process noise for the flat floor demonstration. A thorough characterization of the intensity on-orbit may be necessary in future applications.
- *Sensor Noise*: Similarly, the sensor noise on the ISS is expected to be a function of the lighting conditions and other disturbances. A thorough characterization of the intensity may be necessary in future applications.

The filters may also be improved by implementing an outlier rejection algorithm that can identify and discard Type I errors which provided a certain amount of estimation latch-up. Also, it should be noted that the improvements from the UKF do not

come for ‘free’. The additional sigma points added computation time for propagation and filtering on the order of n . The boresight visual docking methodology presented in this chapter is directly traceable to NASA’s cross-enterprise roadmap documents. These results are intended to be scaled to future manned or unmanned missions to LEO, the Moon, Mars and beyond.

Chapter 5

Actuator: Prototype Design of a Free-Flying Robotic Manipulator

5.1 Overview

This chapter details the preliminary efforts in developing an active, maneuverable appendage to the Synchronized Position Hold and Reorient Experimental Satellites (SPHERES) facility aboard the International Space Station (ISS). A robotic arm for SPHERES can help scientists address many of the challenges of satellite servicing, including time-varying moments of inertia, path dependent actuation, satellite re-purposing, and even locomotion.

Spacecraft with robotic arms have applications both as Assistive Free-Flyers (AFFs) *inside* of a space station and as robotic servicers for satellites *outside* of the space station and in free orbit around Earth. AFFs with robotic arms can aid astronauts in a variety of ways, from performing environmental surveys to distributing inventory. Since astronaut crew time is perhaps the most expensive resource aboard the ISS, adroit AFFs can reduce crew overhead and let the astronauts focus on executing the cutting-edge science for which the ISS was built. Similarly, as described in Chapter 1, manipulators installed on robotic servicers significantly expand the capabilities of a satellite tender.

An external appendage to a spacecraft can add value in a variety of domains.

Such a manipulator should be able to secure itself (or “perch” itself) on a target structure for extended periods of time, and it may be used to assist in spacecraft berthing. This appendage would be of maximum utility if it possessed the ability to point instruments in the full 6 degrees of freedom possible. A fully universal end-effector of a robotic arm would allow the free-flyer to grip a variety of interest points, such as handrails, antennas or other structures. It may also be convenient for the end-effector to be replaceable or swappable depending on the application.

This chapter begins by reviewing the designs of robotic manipulators previously flown and operated in a space (or space-like) environment. Next, Section 5.3 reviews mechanical concepts and prototypes evaluated at MIT for utilization on the SPHERES testbed. Section 5.4 develops simplified control laws for operation of a planar robotic arm via simulation, and Section 5.5 expands on this by showcasing prototype verification results through ground testing. Section 5.6 concludes the chapter by mapping out future work on this project.

5.2 Literary Research

The natural place to start in designing a robotic arm for SPHERES is to first evaluate the current state-of-the-art and previous hardware built in the field of space-based robotic manipulators. This provides guidance and direction in the design cycle for the SPHERES appendage. Since SPHERES is a research testbed for advanced space technologies, an experimental manipulator can be constructed that is of intellectual merit to the space robotics community and adds value to existing or planned space missions. Ten different space arms have been identified and briefly summarized herein. The particular metrics of interest are the geometries, relative sizings, and mechanism designs of each arm. This information ultimately drives the development of the SPHERES Ambulant Robotic Manipulator (ARM) and many of the same attributes are manifest in Section 5.3. Although the robotic landers such as InSight, the Mars Polar Lander, the Mars Surveyor 2001 and the Mars Science Laboratory all employ robotic manipulators, only free-flying arms are evaluated in this study.

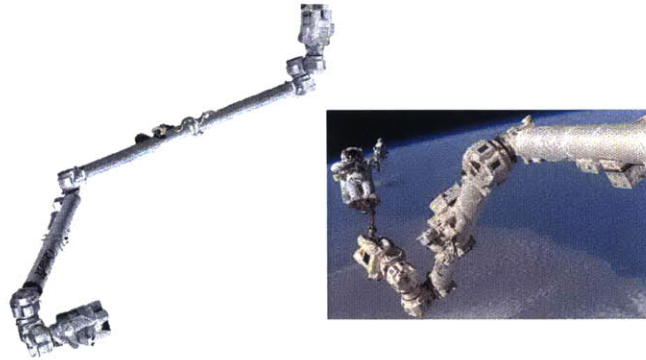


Figure 5-1: Two views of the Space Station Remote Manipulator System (ie Canadarm 2) [8]

The first generation Canadarm is perhaps the most well known space manipulator. The purpose of this arm was to assist in the deployment and retrieval of space hardware from the payload bay of the Space Shuttle orbiter. It was a 50 ft (15.2 m) appendage made from carbon composite material [34] with a similar range of motion to a human arm. It had six degrees of freedom (2 per the shoulder, 1 per the elbow, 3 per the wrist) with a mass of approximately 410.5 kg [35]. The arm housed two cameras (elbow and wrist) and could be operated autonomously or through astronaut control. It always returned with each shuttle mission, which contrasted the second generation Canadarm 2 (aka the International Space Station Mobile Servicing System). The Canadarm 2 moves end-over-end like a slinky or inchworm traveling the entire length of the space station on the Mobile Base System. Each end can provide power, data, and video signals. It adds one degree of freedom in the shoulder joint compared to its predecessor. The Canadarm 2 has a length of 17.6 meters (57.7 ft) and a mass of 1800 kg [36]. It was launched in 2001 and has been operable through 2015 and beyond.

Dextre is the “Canada Hand”, also known as the Special Purpose Dexterous Manipulator (SPDM), as shown in Figure 5-2. The job of Dextre is to perform maintenance work and repairs on the ISS, replacing batteries and cameras external to space station. Dextre is actually operated from the ground, which significantly frees up astronaut time for other space science. Dextre is extremely agile and employs

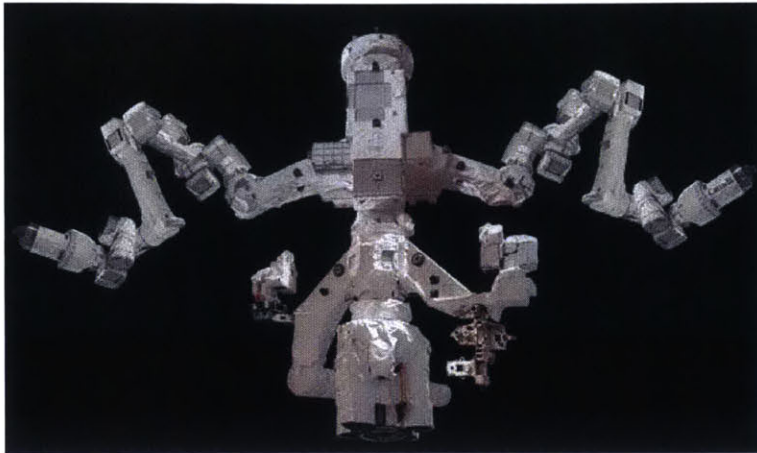


Figure 5-2: The Canadian Built Dextre has been Operational Aboard the ISS Since 2008 [9]

two 3.35 meter arms that can grasp onto the Canadarm (Figure 5-2) [37]. Each manipulator is a seven DoF structure capable of handling 600 kg payloads [38]. Dextre draws an average power of 600 W and has a maximum arm speed of about 2.5 degrees per second [37].

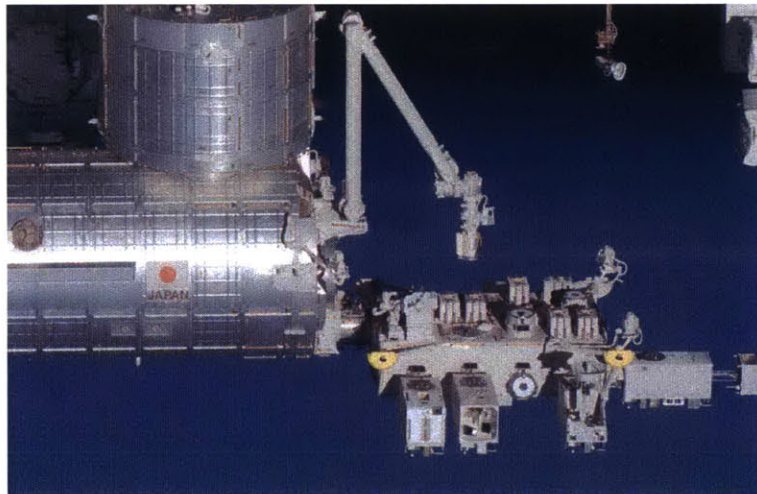


Figure 5-3: The Japanese Experiment Module (JEM) Remote Manipulator System (RMS) [10]

The Japanese Experiment Module (JEM) Remote Manipulator System (RMS) is a 10 meter robot arm that services the ISS Exposed Facility (EF) and helps move equipment to the Experiment Logistics Module (ELM), as seen in Figure 5-3. This

arm was first operated in 2008 and possesses a six DoF range of motion [39]. However, the JEM Small Fine is a 2 meter extension that attaches to the end effector of the main arm. The JEM RMS was desied to use the same grapple fixtuers as the Canadarm 2. The JEM RMS has a maximum arm actuation speed of 2.5 degrees per second, can achieve less than 5 cm end-tip accuracy and can handle payloads up to 7000 kg in mass [40].

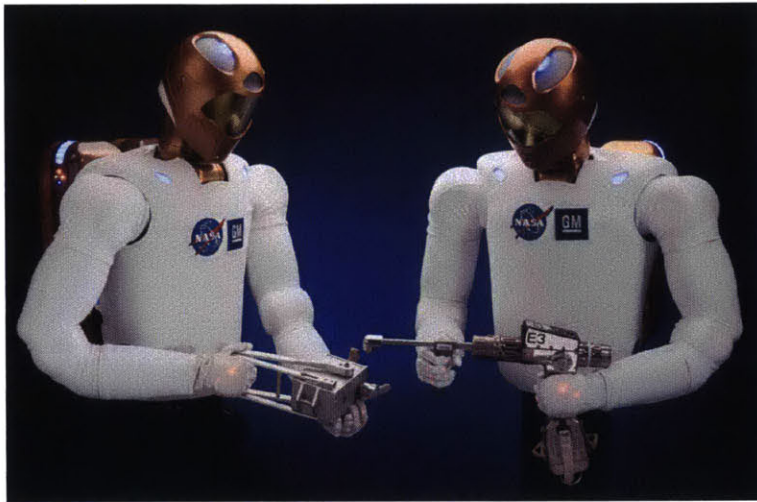


Figure 5-4: The Robonaut Arms Are Capable of Grasping Tools [11]

Robonaut is a humanoid robot that is active inside of the ISS with the astronauts (Figure 5-4). The machine resembles the upper torso of a person and has been designed to work side-by-side with the astronauts on a variety of tasks. Robonaut 2 was launched in 2011 with a mass of 149 kg and can freely handle packages up to 9 kg in mass [41]. The robot is usually controlled via teleoperation from the ground. The arms have a stretch length of 0.81 meters, 42 degrees of freedom (including fingers), and can move up to 7 linear feet per second. In total, robonaut has cost upwards of 2.7 millions dollars, which does not include development or testing expenses [42].

The Engineering Test Satellite VII (aka Kiku-7) was part of a technology demonstration mission performed by JAXA during the 1980s (Figure 5-5). The over-arching mission objectives were to conduct unmanned orbital operation and servicing tasks, both automatically and as a remotely piloted agent. A robotic arm 2 meters in length weighing 160 kg was installed and provided 6 DoF maneuvering capability [43]. The

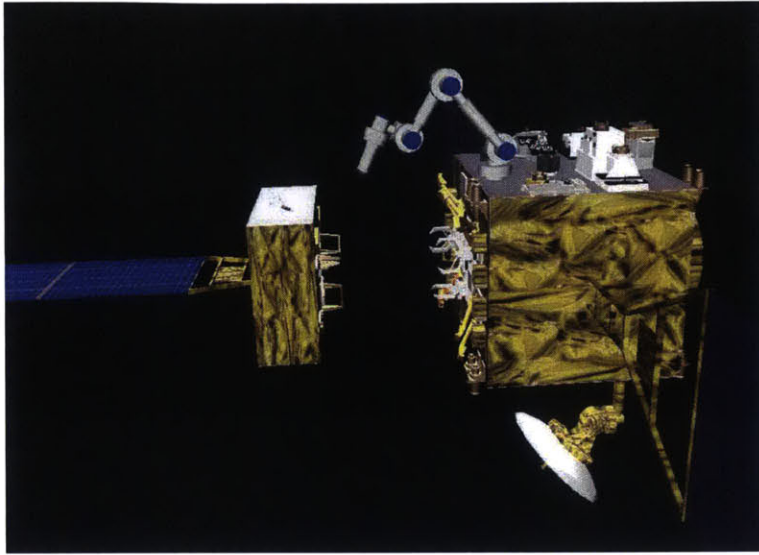


Figure 5-5: The Engineering Test Satellite VII Robotic Arm [12]

manipulator was capable of less than 2mm positioning accuracy at the end tip of the satellite while providing more than 40 N of force [43]. The actuators were driven by DC brushless motors and a harmonic drive gear train. In the end, this arm demonstrated the first successful release, tracking, and capture of a target satellite without help from the ground, something the United States has yet to demonstrate as of 2015.

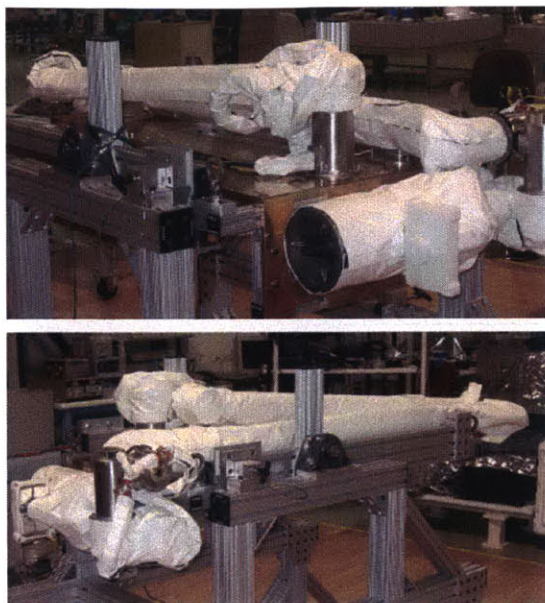


Figure 5-6: The Orbital Express OEDMS Robotic Arm During Final Assembly [13]

The Orbital Express Demonstration System (OEDS) was an on-orbit satellite servicing mission that took place from March to July 2007. A robotic arm called OEDMS (Figure 5-6) was deployed to “autonomously capture a fully unconstrained free-flying client satellite, autonomously transfer a functional battery between two spacecraft, and autonomously transfer a functional computer” [44]. This arm was a six degree-of-freedom limb drawing 131 W during nominal operation [44]. The arm performed a global video survey early in the mission and also was used to reorient the target satellites for a sensor suite checkout [44]. The rates and tip speeds were carefully controlled during autonomous operations.

The European Robotic Arm (ERA) is a planned robotic servicing system that is intended to service and re-assemble the Russian segment of the ISS (Figure 5-7). It is unique in that it is a symmetric, 7 DoF system with two end-effectors that can act as a hand or a base depending on the orientation. This arm is 11.3 meters in length, with an up-mass of 630 kg and a payload capacity of 8000 kg [45]. As a walking arm, it has the ability to reach remote locations on the ISS and will use this ability to inspect exterior surfaces with an infrared camera. It has a maximum tip speed of 10 cm/s and is composed principally from carbon fiber and aluminum [45]. This

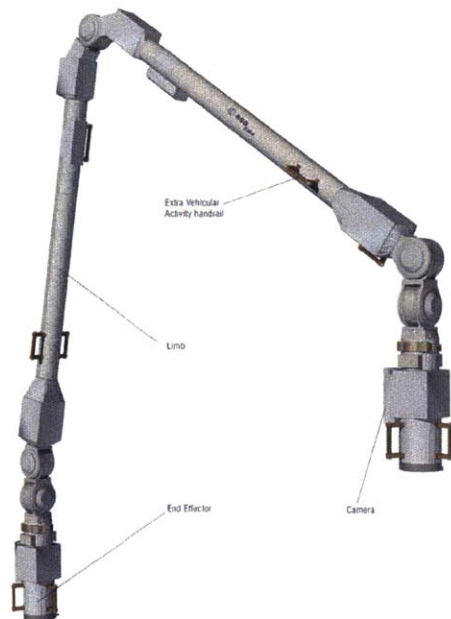


Figure 5-7: The European Robotic Arm (ERA) [14]

appendage has cost ESA over 250 million Euros [46].

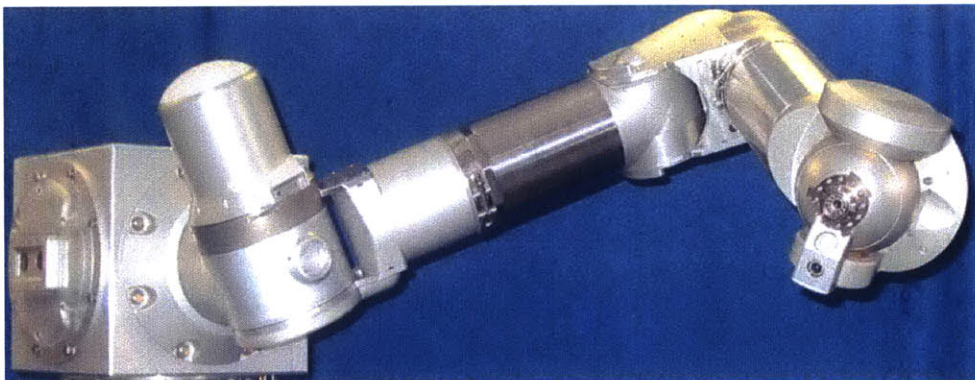


Figure 5-8: The Ranger Dexterous Manipulator [15]

The Ranger Neutral Buoyancy Vehicle (RNBV) was part of an underwater robotics testbed at the University of Maryland that was used to advance robotic manipulation in neutral buoyancy. The Ranger manipulator (Figure 5-8) is an 8 DoF system with a 1.35 meter reach and weighing 77 kg [47]. This arm was designed to be operated underwater and have roughly the same grip force (311 N) and reach capability as an astronaut in a space suit. Ranger was capable of both teleoperation and autonomous

control. Significant effort was invested to achieve positioning accuracy of 2.3 cm at high speeds (1 m/s) [15]. Although Ranger was rated for a vacuum and neutral buoyancy, it never directly transitioned to a flight project.



Figure 5-9: The Dynamic Manipulation Flight Experiment (DYMAFLEX) [16]

The University of Maryland's Dynamic Manipulation Flight Experiment (DYMAFLEX) is the follow-on project to Ranger, leveraging much of the technology experience but applying it to a nanosatellite bus. DYMAFLEX (Figure 5-9) is intended to investigate the coupled dynamics and associated control mitigation strategies for a free-flyer in performing satellite servicing activities [16]. This mission statement is analogous to that of the SPHERES Ambulant Robotic Manipulator (ARM) which is the subject of this chapter. The DYMAFLEX manipulator is 4 DoF with an actuator length of 65.4 cm [16] and a maximum tip velocity of 2.0 m/s. The arm draws 13.5 W of steady state power [16].

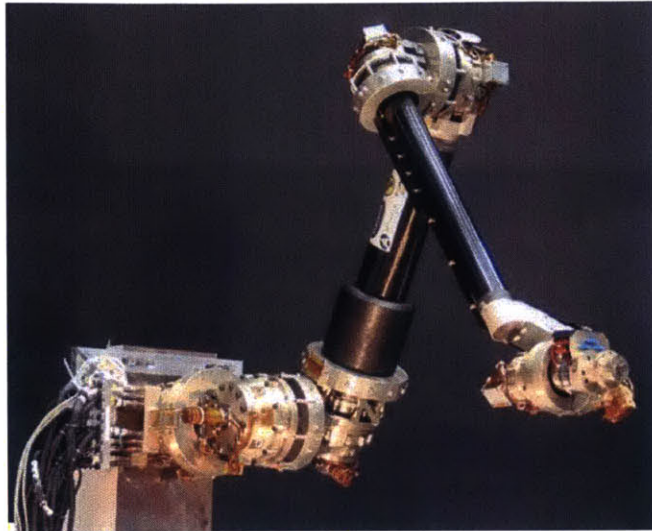


Figure 5-10: The Front-end Robotic Enabling Near-term Demonstration (FREND) [17]

The Front-end Robotics Enabling Near-term Demonstration (FREND) is a DARPA-sponsored concept that is designed to demonstrate autonomous grappling of a resilient space object (RSO) as part of the DARPA RSGS mission. Many flight-traceable components and algorithms can be found on the FREND arm including servo control, impedance control, inverse kinematics, machine vision and guidance laws [48]. The FREND arm (Figure 5-10) is a 7 DoF manipulator with a reach length of 2.4 meters and a mass of 78 kg [17]. Most notably, the arm has a tip positioning resolution of less than 2 mm and a tracking accuracy (of a moving object) of less than 1 cm. The end effector consists of a tool drive that can actuate a range of swappable grippers.

5.3 Hardware Design

This chapter showcases a series of advanced concept designs for a robotic manipulator for the SPHERES satellites. This robotic appendage may have a variety of uses: prodding unknown objects; pushing and pulling cargo; retrieving and returning tools, equipment or other assets; assisting in the deployment of expandable structures; harvesting parts from retired satellites; servicing depleted or damaged satellites. Servicing itself is a broad term and but generally consists of actions necessary to

extend the operational life of a spacecraft by replacing propellant and batteries or refurbishing solar panels and avionics. With these applications in mind, a robotic arm for SPHERES was iteratively developed beginning with a LEGO Mindstorms prototype and evolving into a 4 DoF design leveraging 3D-printing for complex geometries.

5.3.1 Lego Arm Overview

LEGO Mindstorms was selected as the initial platform for developing a robotic arm for the SPHERES for a number of reasons. First, and most significantly, there already exist LEGO Mindstorm materials on the International Space Station. The LEGO NXT Intelligent Brick has been operating aboard the ISS for years, most recently demonstrating a spinning gyrobot for educational purposes [49]. One possible direction for developing a robotic arm for SPHERES would be to construct the arm in-situ, by providing the astronaut with detailed build instructions. This would reduce the required up-mass to the ISS to a simple interface board that would act as a relay to the SPHERES expansion port. Since the LEGO Mindstorm motors, plastics and electronics have already passed the requisite NASA Safety requirements (such as EMI and flammability), ground testing and development time would be significantly reduced. The proposed solution is presented in Figure 5-11.

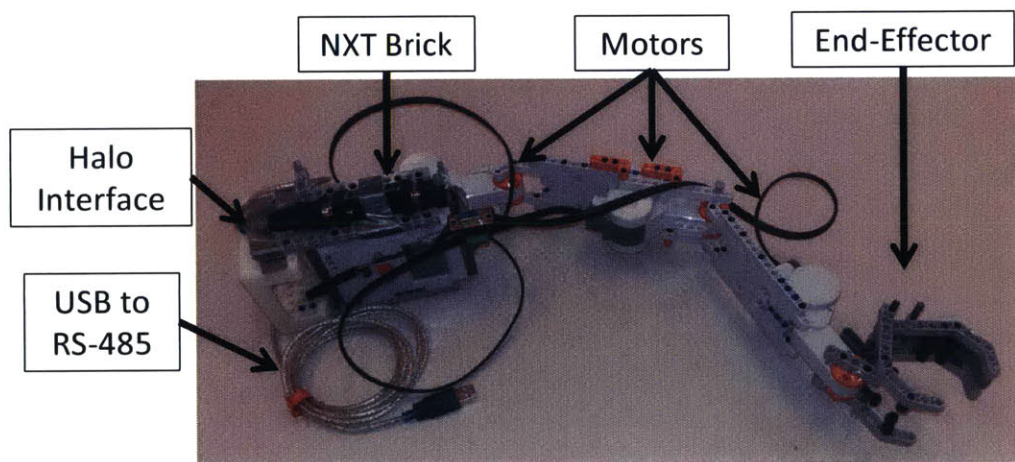


Figure 5-11: An Assembly Overview of the SPHERES Lego Robotic Arm

The second advantage to using Lego Mindstorms for SPHERES is that it supports quick fabrication and testing of physical assemblies. This means that the design could be rapidly iterated upon by the astronaut if a new solution to a task is required. This would be a paradigm shift from the traditional space payloads that are unable to adapt to changing mission objectives.

The Lego Minstorms design is powered by 6 AA batteries, the same energy source used by the SPHERES satellites themselves. The NXT brick can be programmed in cross-platform language called RobotC, which facilitates development beyond the graphical programming language bundled with the retail product. More specifically, it enables efficient tuning of the feedback control software for the motor actuators. Each motor includes a rotary encoder which enables precision arm control. The mechanical unit was tested experimentally with a SPHERES Halo assembly on the MIT glass table facility, demonstrating the feasibility of the robotic system for space servicing applications. The Lego NXT was programmed to interface with the Halo Guest Scientist platform (Section 2.4), making it the sixth payload to do so.

5.3.2 SPHERES Ambulant Robotic Manipulator (ARM)

This section details the efforts in designing and constructing a custom robotic arm for SPHERES, termed the Ambulant Robotic Manipulator (ARM). Thus far, three generations of designs have converged on the article shown in Figure 5-12. This manipulator has leveraged lessons learned from the Lego prototype and previous ISS payloads build by the MIT Space Systems Laboratory. The SPHERES ARM has been designed from the ground up to be lean in both mass and complexity. The five principle design drivers are

- **Functionality:** A drawback to the Lego Mindstorms prototype is that the NXT brick constrains the degrees of freedom to 3 based on the number of available digital output ports. A custom arm built by MIT provides the freedom to explore the entire design space of out-of-plane manipulation, as well as to choose the optimal actuators for each joint. In this way, the arm can be optimized to

satisfy the set of mission objectives required by the payload. A secondary objective is for the robotic arm to be able to lift itself in 1-g. Although not necessary for space ops, this is desired for 1-g testing in the SPHERES ground facility.

- **Traceability:** Since the SPHERES is ultimately a testbed for emerging space technologies, the SPHERES-ARM must provide flight traceable returns that can be applied to future space missions needing robotic arms. Thus, it is preferable to make the robotic arm as “interesting” as possible for research purposes in order to validate a wide a range of algorithms as possible. For this reason, out of plane motion provided by at least one more degree of freedom is desired from the SPHERES-ARM.
- **Complexity:** The Lego Mindstorms brick adds a third processor to the free-floating system, adding to the SPHERES Digital Signal Processor (DSP) and the VERTIGO Avionics Stack’s PICO-ITX. With each processor comes overhead which reduces the science output of the system. This motivated a custom arm solution directly controllable by the Halo. For this reason, a custom arm solution is desired to be directly integratable into the existing hardware testbed.
- **Compatability:** The mechanical and electrical interfaces of the SPHERES-ARM have been designed to be mountable to and controllably by the Halo.
- **ISS Safety:** All ISS payloads must conform to the array of safe requirements stipulated in the NASA Safety and Integration Reuglations for ISS Payloads Operations. Although the current third generation arm does not meet all safety objectives, future design iterations should meet the following
 - *Standard Safety Hazards:* Stowage Structural Failure, Sharp Edges/Corners/Protrusions, Shatterable Materials, Flammable Materials, Materials Offgassing, Non-ionizing Radiation (EMI), Electromagnetic Compliance (EMC), Touch Temperature, Electrical Power Distribution, Rotating Equipment, Mating/Demating Powered Connectors, and Contingency Return and Rapid Safing

- *Testing for:* EMI, Touch Temperature, Acoustics
- *Analysis for:* Thermal, Structural (launch) Loads, Crew Induced Loads (worst case kickloads), Sharp Edges, Pinch Points, Holes

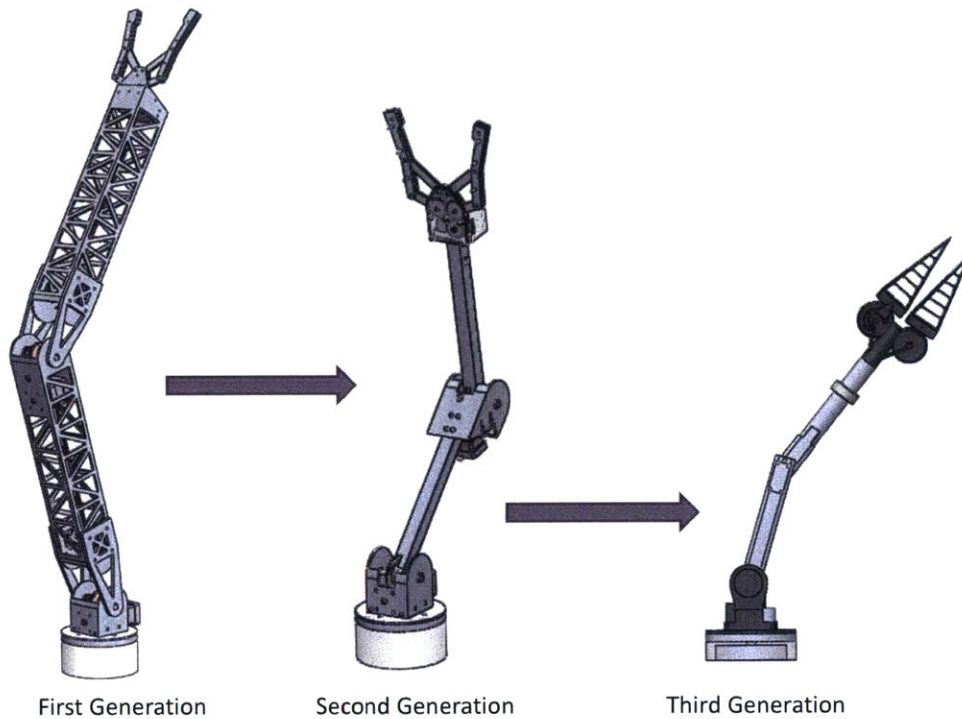


Figure 5-12: Evolution of the SPHERES-ARM Over Three Generations of Designs

The robotic arm depicted in Figure 5-13 is one of the concepts first evaluated in this study. This design is a 4 degree-of-freedom (DoF) manipulator providing 3D out-of-plane capabilities. The final lengths of the upper arm and forearm can be varied during testing to determine the optimal size for use inside the ISS. A turntable at the base of the arm provides out of plane rotation using a continuous-rotation servo. A shoulder mechanism sits atop the rotational base to actuate the rest of the arm. The shoulder employs a worm-gear mechanism to provide the necessary step-up torque for arm manipulation. The worm-gear also prevents disturbance torques on the arm from back-driving the motor. If additional manipulation is required, an elbow joint consisting of a belted stepper motor that drives a worm-gear (in much the same way as the shoulder) can supply additional flexibility as needed.

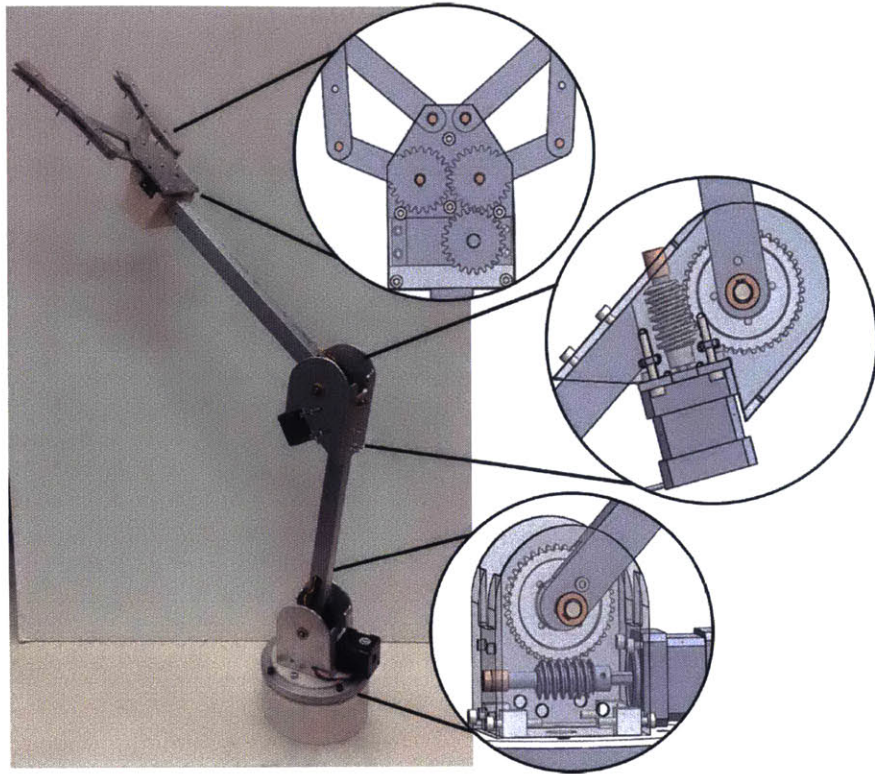


Figure 5-13: A Mechanical and Electrical Prototypes of the Second Generation SPHERES-ARM

During the design process, multiple motor actuators were considered for the SPHERES ARM. The Generation 2 model includes both stepper motors and servo motors. Stepper motors are capable of continuous rotation with precise position control - a feedback system is not required. This is useful because the worm drive mechanism in the base and shoulder have a large gear step-up ratio, meaning the shoulder and elbow motors should be capable of continuous motion. Because steppers rely on magnetics for rotation, they are also extremely low EMI which is best for meeting ISS integration requirements. However, stepper motors consume high levels of current relative to their output torque. Moreover, their large volume is less than desirable for this application. This motivated the move to a pure servo solution in the Generation 3 model.

R/C servos are an integrated package that includes a DC motor, gearbox, and control circuitry with feedback. Compared to stepper motors, servos provide much

higher speed (up to 2000 peak RPM) and torque, and are also more power efficient (in the 80-90% range). The control signal is pulse width modulated (PWM), which means that the angle position can be accurately commanded. Moreover, multi-turn servos are available for applications (such as ours) which may need additional windings. Continuous rotation servos are also available (or can be modified) but position control is lost without an external sensing source. For the Generation 3 ARM, a continuous rotation servo was selected for the turntable base, a six-turn servo was selected for the shoulder joint which is sufficient for the 4:1 step-up gear ratio of the shoulder, and a 360 degree servo was selected for the elbow mechanism. Back of the envelope calculations showed that the necessary torque could be provided to actuate the full range of motion of the arm in 1-g.

Using the literature review from Section 5.2, a variety of mechanisms were studied to see what would be most appropriate for the SPHERES ARM. Chief among design considerations were planetary gears, worm drives and spur gear winches. Planetary gears offer smooth operation, high gear ratios, and transfer efficiencies up to 65% [50]. The gear layout resembles a sun with revolving planets, hence the name. They are known to be one of the most compact gear trains which makes sense for the SPHERES ARM application. However, they were not utilized in the third generation ARM design due to high part count and high number of potential failure points. A worm drive, or worm gear mechanism, was selected for the second generation's shoulder and elbow joints despite the low torque transfer efficiencies of around 40% due to high frictional losses [50]. A worm gear provides a 90 degree change in direction of rotation and provides a large gear ratio. Also, the high friction with the helical screw prevents backdriving. These gears have operational heritage in the SPHERES UDP (Chapter 3). However, the third generation design relies principally on spur gears in the base and shoulder joints. Since the third generation arm is largely 3D printed and lightweight, only a 4:1 spur gear step-up is required for the shoulder joint and turn table. The elbow motor is expected to be strong enough on its own. Spur gears are very reliable and have the highest frictional transfer efficiency reaching 75%.

With these design selections in mind, the Generation 3 robotic arm has been developed to meet a range of operational objectives. First, it can fold completely on itself, which makes stowage easier for launch and on the ISS. Second, it supports a swappable end-effector which is desired to study multiple science scenarios, including grasping of mechanical objects or tagging RFID chips aboard the ISS for inventory management. Figure 5-14 shows the intended design.

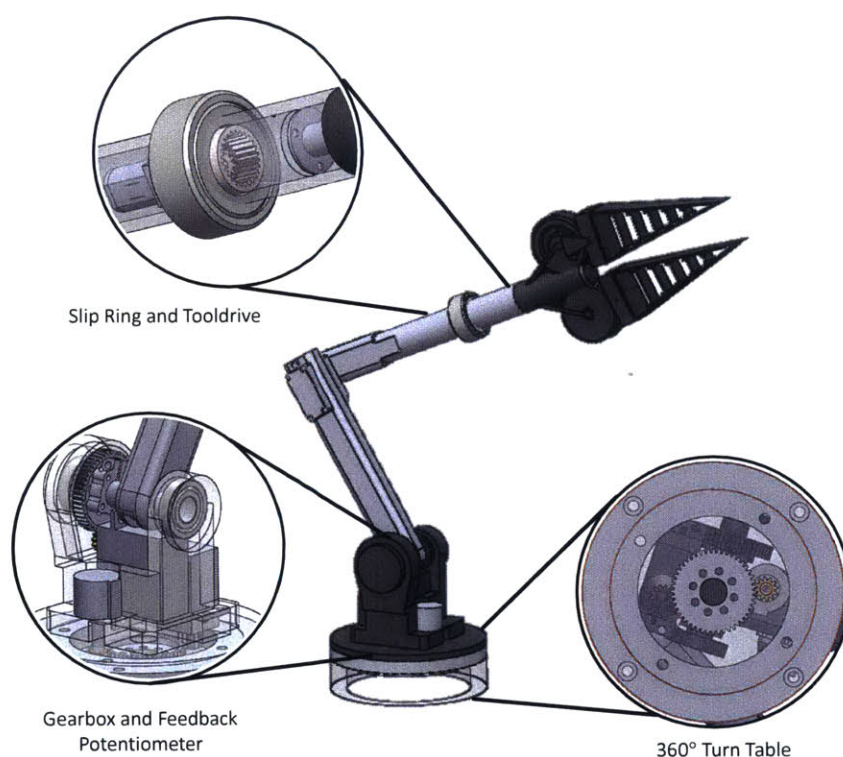


Figure 5-14: The Design of the Third Generation SPHERES-ARM Prior to Manufacturing

End-Effector Design

A traditional end-effector provides the grasping force needed for target capture. Sample capture points include crew handles or other free floating targets like the SPHERES which can represent a retired or damaged satellite in need of servicing. The marman ring is a common adapter used on satellites to mate to the upper stage of a rocket. Thus it makes sense for the end-effector to be capable of grasping the

edge of a ring-like object. Other creative end-effectors have been considered, ranging from a universal end-effector like a jamming granular gripper for grasping irregularly shaped objects, to flexible grippers that incorporate gecko or electrostatic adhesion.

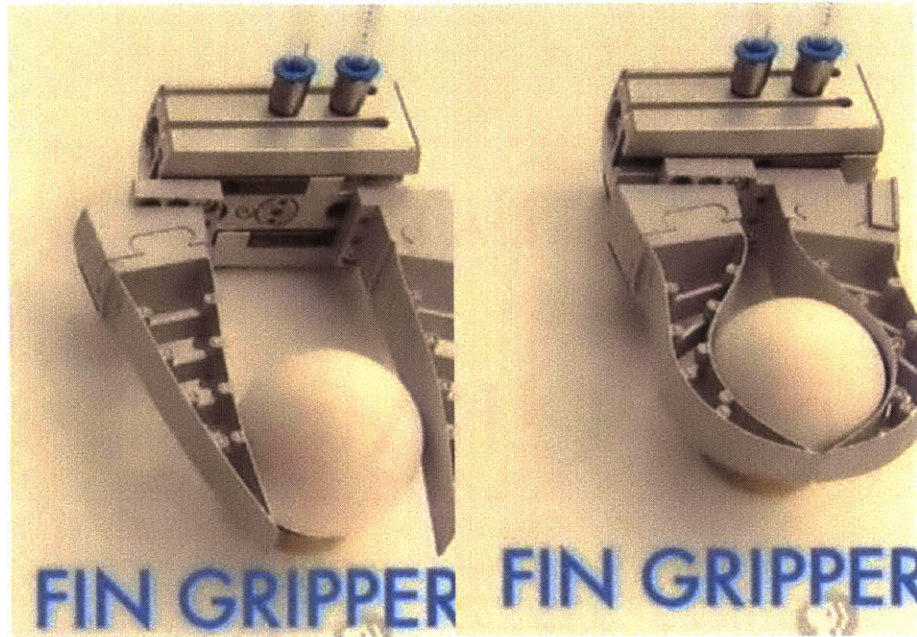


Figure 5-15: The Fin Ray Gripper Can Apply Equal Pressure Gently But Firmly to an Egg [18]

The end-effector for the SPHERES-ARM has been designed to be replaceable by the crew, which makes future expansion possible. There is a multi-wire slip ring present in this design, which can support both rotary tool drives and electronic payloads such as RFID scanners or cameras. The first prototype design proposed is a fin ray gripper, inspired by the flexibility of both fishtails and elephant trunks (Figure 5-16). This bio-engineered design is expected to be effective in grasping a marman ring or a target SPHERES in order to tug damaged satellites into new orbits or orientations.

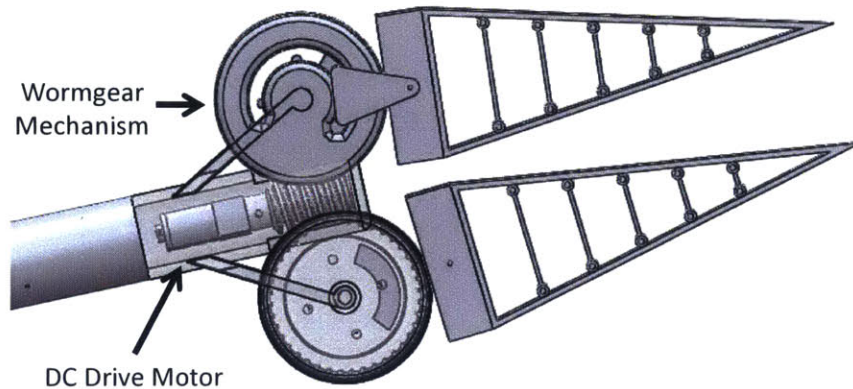


Figure 5-16: The Proposed Fin Ray Gripper Design for the SPHERES ARM

Expected Applications

The SPHERES-ARM is intended to be mounted directly on a SPHERES Halo for incorporation into the Halo Guest Scientist robotics platform. Since the Halo exoskeleton supports up to 6 peripherals simultaneously, it would be reasonable to operate at least two arms from a single Halo. In this way, the second (or third) arm can provide a reactive force to stabilize the satellite during servicing operations.

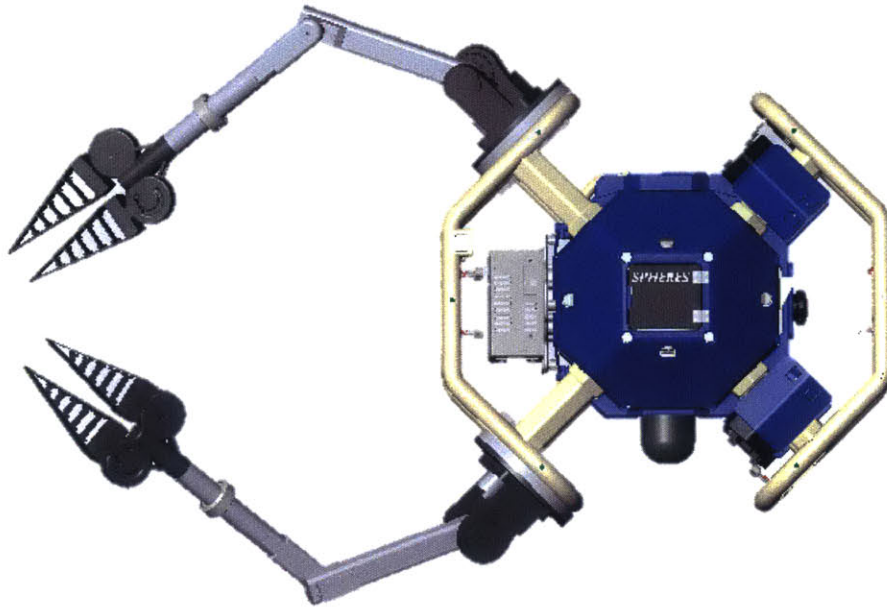


Figure 5-17: Two SPHERES-ARMs Mounted on the Halo for Satellite Servicing Activities

5.4 Arm Control and Simulation

This section presents preliminary work on the control and dynamics of a free-floating robotic arm. Both the planar and restricted out-of-plane inverse kinematics problems and solutions are presented. Also, the results from an exploratory dynamics simulation are described. In the past, the coupled dynamics between the satellite bus and robotic arm have been ignored and treated as negligible disturbances. However, the large mass ratio between SPHERES and its manipulator implies that this controls problem must be addressed in order to accurately command and control a free-flying arm for the satellite. Solutions to these unrestricted free-body problems are non-trivial and still an area of active research.

5.4.1 Inverse Kinematics

The kinematics of a robotic system describes the arrangement of each component in 3D space. In the context of the SPHERES robotic arm, the robotic system can be represented as a collection of linkages with specific masses, inertias, lengths and geometries. Forward kinematics uses the angular rotation of a joint to compute the configuration of the system; typically the location of the end effector is the variable of interest. Inverse kinematics reverses this problem by solving for the required joint rotations given a desired end effector location. Unfortunately, the robotic kinematics is a surjective problem. That is, the forward kinematics can be easily computed to obtain a single solution, but the results of the inverse kinematics may provide multiple (or infinite) solutions. This introduces complexities in the problem formulation, especially for complicated geometries.

Planar Problem Formulation

For the SPHERES robotic arm, a simplified model of straight-bar linkages is created, and the satellite body-frame is considered fixed in this sub-section. In two dimensions, the most obvious approach of solving the inverse kinematics problem is to use a sequence of conjoined circles. With a two-joint arm, a single circle is drawn with a radius of the length of the first arm segment, centered at the base of the arm, (x_1, y_1) . Next, another circle can be formed with a radius the length of the second arm segment, centered at the desired end effector location. Where these circles intersect is the location that the “elbow” joint needs to be in order to reach the desired end goal. It is clear that in this scenario, there are two equally valid solutions (Figure 5-18). The executed solution should depend on the initial condition of the arm and any additional external constraints.

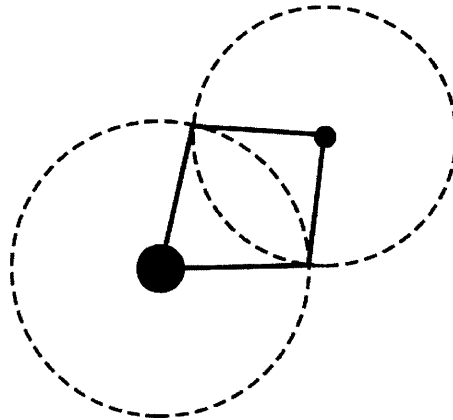


Figure 5-18: Conjoined Circle Method (CCM) for Inverse Kinematics

The same method can be used for greater degrees of freedom and additional arm segments. There will be more solutions, however, (usually an infinite number of solutions) due to the nature of the two dimensional space. Figure 5-19, showing one possible approach, illustrates this concept. A limited number of circles is shown in the figure for clarity, but it is noted that there are in fact an infinite amount of potentially intersecting circles.

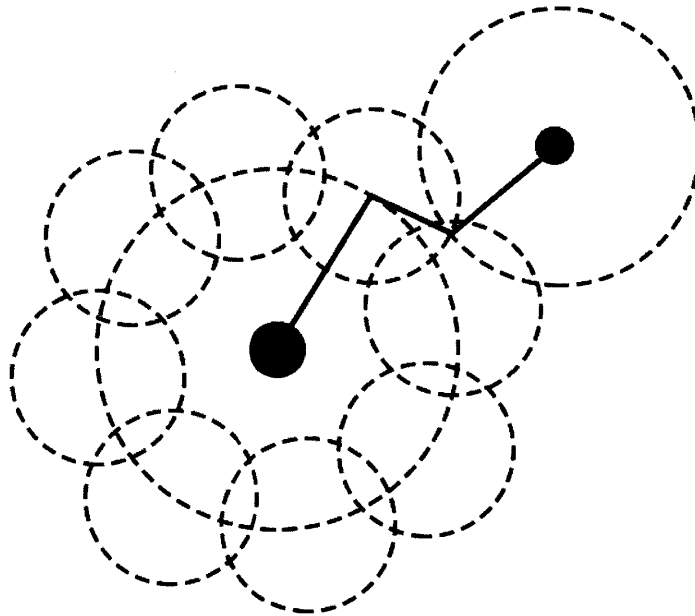


Figure 5-19: Conjoined Circle Method for 3 DoF

Mathematically, the Conjoined Circle Method (CCM) can be solved using a nonlinear solver, such as *fsolve* in Matlab. For a two-joint arm, the unknown variables are the x,y locations of the elbow joint (x_2, y_2) . Once this is known, simple trigonometry can be used to solve for the angle of the first and second motors, which is the control input for the hardware.

Increasing the number of arm segments increases the number of free variables. Collectively, this becomes a two point boundary value problem, with the constraints being lengths of the rigid linkages and the boundaries being the fixed locations of the base and end-effector.

$$(x_2 - x_1)^2 + (y_2 - y_1)^2 = L_1^2 \quad (5.1)$$

$$(x_3 - x_2)^2 + (y_3 - y_2)^2 = L_2^2 \quad (5.2)$$

$$\vdots \quad (5.3)$$

$$(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2 = L_n^2 \quad (5.4)$$

The base of each linkage is co-located on the rim of the circle of the previous linkage. This cycle continues until the n^{th} segment centered at (x_n, y_n) reaches the desired end effector location at (x_{n+1}, y_{n+1}) . There will also be additional constraints to consider, which helps prune the solution space. First, there are contact constraints of the physical arm which prevent bodies from intersecting. This limits the range of motion for each motor. Second, it is logical to minimize the travel path of the arm from its initial condition. Finally, each solution changes the center of mass and inertial properties of the robotic system, which may or may not be acceptable based on capabilities of the satellite's control system. To obtain the updated inertia properties, the parallel axis theorem can be applied to each linkage based on the forward kinematics. All of this has been applied to an MIT-internal Matlab tool shown graphically in Figure 5-20. The algorithms could be autocoded to C for application to the embedded SPHERES environment.

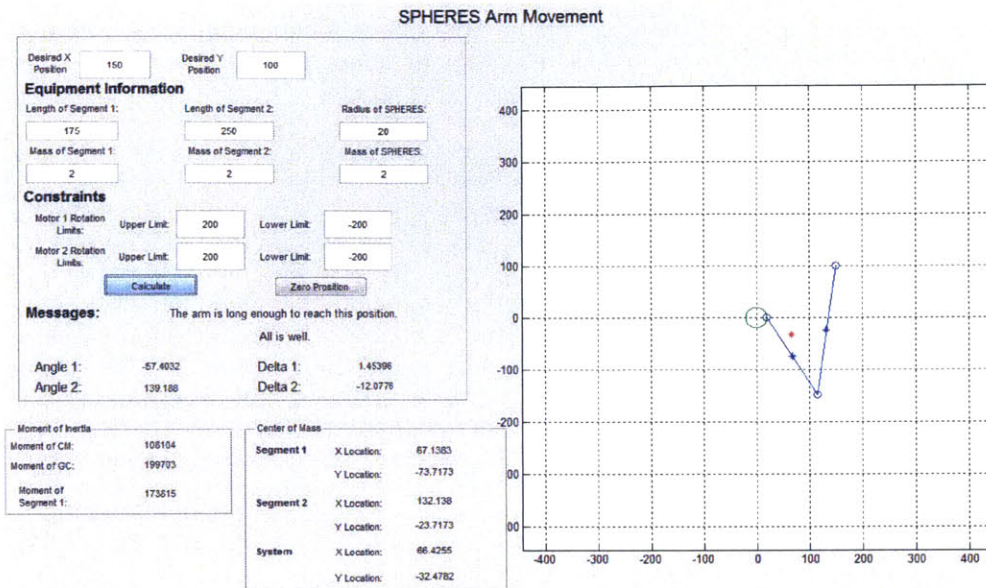


Figure 5-20: Conjoined Circle Method for 3 DoF

Restricted 3D Problem Formulation

The arm control problem can be extended to three-dimensional space when at least one motor has a degree of out-of-plane actuation. If each joint was a full three DoF ball joint, a conjoined spherical method could be used in a similar manner to Section 5.4.1. In this section, we consider a restricted 3D arm based on the Ambulant Robotic Manipulator presented in Section 5.3.2. In this design, all of the arm segments lie in the same plane, but the plane can rotate. In the end, the desired end-effector location must lie in the plane defined by the robotic arm. Mathematically, we seek to ensure that $(P - D) \cdot N = 0$ is satisfied, where P is a point on the plane of the arm, D is the desired location, and N is the vector normal to the plane of the arm.

The problem can be reduced to finding the angle θ , representing the rotation angle of the base, so that the desired destination point $X_D \hat{i} + Y_D \hat{j} + Z_D \hat{k}$ can be reached using the planar solution previously presented. To further simplify the problem, especially in the presence of free-dynamic motion (Section 5.4.2), each motor will be actuated sequentially rather than simultaneously.

Consider a robotic arm initialized in the xy -plane. The y -axis of the arm is defined

to be the rotation axis about which the arm and corresponding plane will be revolved. If the end-effector position has been initialized to $(x_i, y_i, 0)$, then a rotation matrix can be applied to find the new end-effector location as $x_i \cos \theta \hat{i} + y_i \hat{j} - x_i \sin \theta \hat{k}$.

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 0 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow x_i \cos \theta \hat{i} + y_i \hat{j} - x_i \sin \theta \hat{k} \quad (5.5)$$

Next the revolved normal vector to the arm's plane can be found using an equivalent rotation transformation. Since the arm was initialized in the xy -plane, the initial normal was chosen as $(0, 0, 1)$. After some calculations, the rotated normal vector can be represented as $\sin \theta \hat{i} + \cos \theta \hat{k}$.

$$\begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \sin \theta \hat{i} + \cos \theta \hat{k} \quad (5.6)$$

Finally, the constraint equation $(P - D) \cdot N = 0$ can be solved for some value θ .

$$0 = (P - D) \cdot N \quad (5.7)$$

$$= ((\cos \theta \hat{i} + \hat{j} - \sin \theta \hat{k}) - (X_D \hat{i} + Y_D \hat{j} + Z_D \hat{k})) \cdot (\sin \theta \hat{i} + \cos \theta \hat{k}) \quad (5.8)$$

$$= \sin \theta (\cos \theta - X_D) - \cos \theta (\sin \theta + Z_D) \quad (5.9)$$

$$= \sin \theta \cos \theta - \sin \theta X_D - \cos \theta \sin \theta - \cos \theta Z_D \quad (5.10)$$

$$= \sin \theta X_D + \cos \theta Z_D \quad (5.11)$$

Thus, θ is found to be

$$\begin{aligned} \tan \theta &= -\frac{Z_D}{X_D} \\ \theta &= \tan^{-1} \left(-\frac{Z_D}{X_D} \right) \end{aligned} \quad (5.12)$$

This result states how much the base must rotate so that the desired end point lies in the plane of the arm. From there, the problem can be treated as planar, using the conjoined circle method described previously. The complete algorithm has been developed and applied in simulation, using an in-house Simulink model that integrates rotation rates to reproduce motion.

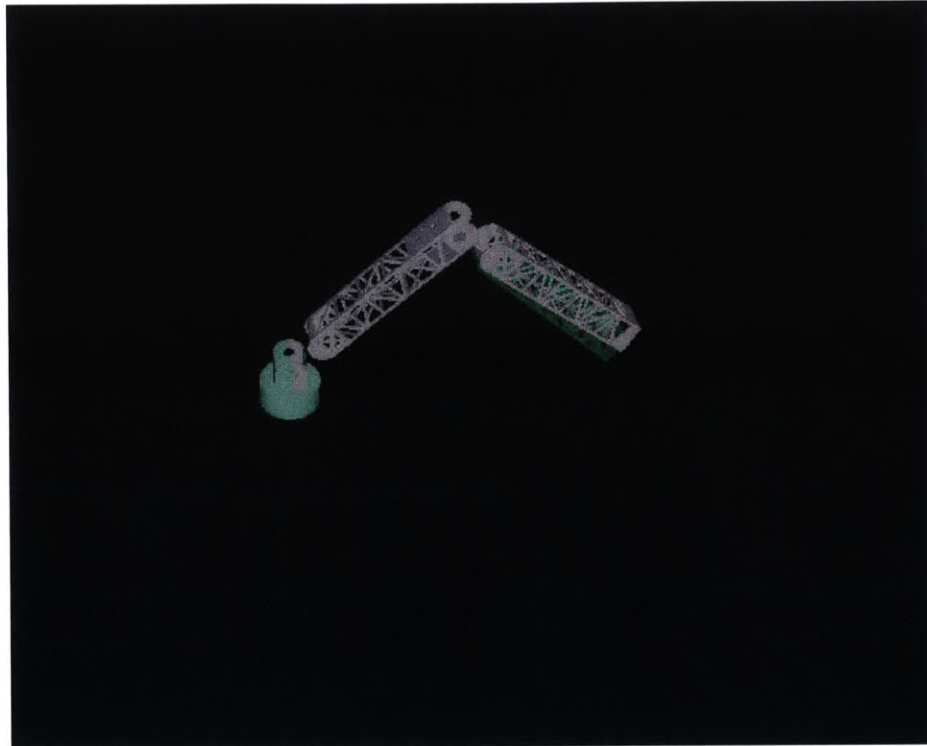


Figure 5-21: [Animation] A Visualization of the Inverse Kinematics Solver for A Fixed-Base, 3D Robotic Arm

5.4.2 Dynamic Modeling

The previous section outlined a well known approach to the kinematics of a fixed robotic manipulator. This section introduces the coupled dynamics between the base and the arm which make the free-floating robotic manipulator problem so interesting. First, we distinguish “free flying” from “free floating”: “floating” indicates that there are no external forces acting on the agent, while “flying” suggests there are thruster

firings (or other external inputs) simultaneous to appendage actuation. For simplicity, only the floating agent is considered in this section, but it is noted that the effect of external forces on a spacecraft is well known and can be super-imposed on the floating solution.

In order to predict the dynamics of the system, a relationship relating multi-jointed bodies can be derived from Newton's 2nd Law of Rotation ($\tau = I\alpha$) and 3rd Law of Reaction. Assuming that the bodies begin at rest, the torque equations can be integrated in order to see how a two-segment system will move the pivot point.

$$\frac{\theta_1}{I_2} = \frac{\theta_2}{I_1} \quad (5.13)$$

Where I_1 and I_2 are the moments of inertia of each of the respective segments, and θ_1 and θ_2 are the respective displacement angles. Thus if one displacement angle is known (such as the commanded rotation of the arm) the reaction angle can be computed. Figure 5-22 shows this graphically.

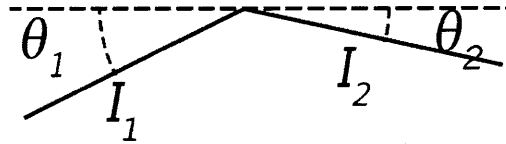


Figure 5-22: The Proposed Angle Displacement Method for A Two-Segment Body

Another property of space manipulators is that the center of mass of the system does not change. To also satisfy this condition, the center of mass in the body frame should be calculated before and after the angular displacement of the manipulator. A “pseudo-movement vector” can then be calculated and applied in the global frame to deduce the updated location of the spacecraft system.

A graphic simulation was created in C++ using the SFML graphics library in order to visualize the proposed displacement angle method and predict arm dynamics. The appendage is extended to three segments (including the SPHERES) to most accurately represent the planar Lego Robotic Arm. Given a desired end effector

location, a brute force search method was implemented in this simulation to deduce the necessary motor commands. Note that the traditional inverse kinematics solution cannot be applied directly. Using the SFML graphics engine, this dynamics can be displayed visually by drawing basic shapes each representing the arm segments and the SPHERES. The results are shown in Figure 5-23.

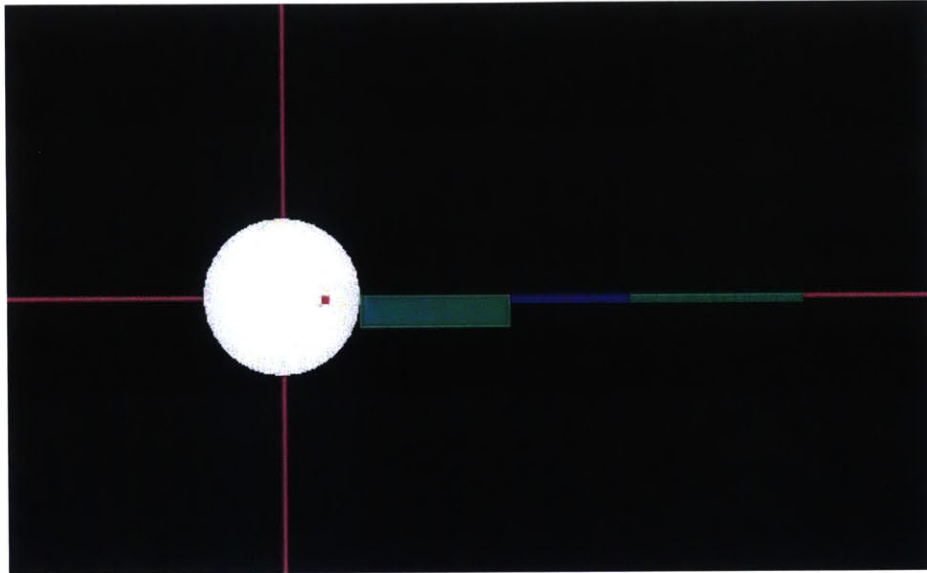


Figure 5-23: [Animation] Dynamic Simulation of a Free-Floating Satellite and Planar Manipulator

One key observation of interest is that the final end-effector location is path dependent [51]. More specifically, given a set of motor commands, the “hand” can end at a different position in space depending on the order and timing of motor execution. Intuitively, this can be deduced by considering Equation 5.13. For a given joint angle, larger reaction angles can be achieved when the inertia is smaller (arm is retracted) and vice versa. This could theoretically be utilized as a propellant-less pointing device because the spacecraft orientation can change based only on manipulator actuation.

In order to showcase the complexities of a full out-of-plane manipulator, a motion study was created in SolidWorks using one robotic arm concept (Figure 5-24). The mass of this appendage is large relative to the SPHERES Halo assembly, which

accentuates the coupled, nonlinear dynamics.

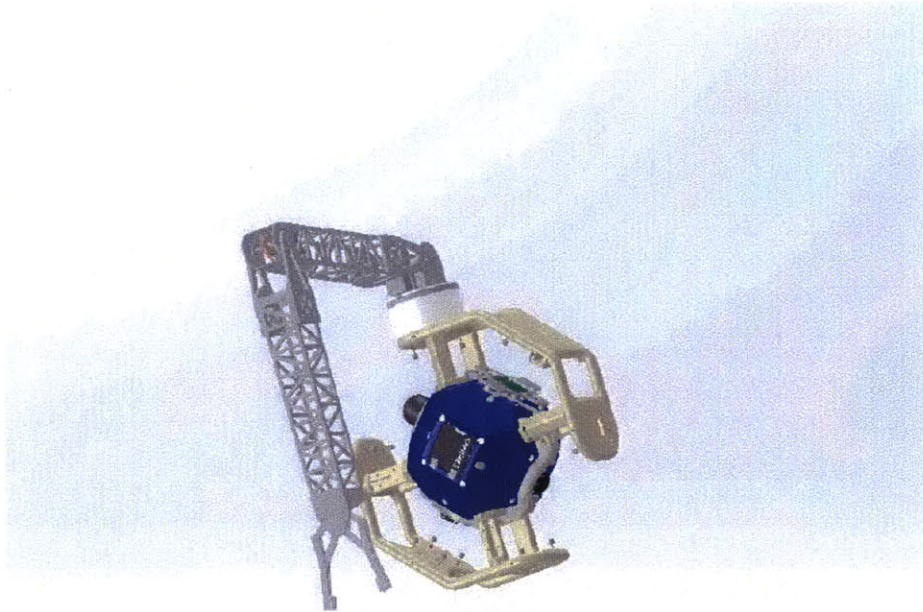


Figure 5-24: [Animation] Dynamic Motion Study of a Robotic Arm and a SPHERES Halo

5.5 Verification in Three Degrees-of-Freedom

A verification test was conducted on the MIT Glass Table Facility which allows for three degree of freedom movement of the SPHERES satellites. The purpose of the test was to first demonstrate that the prototype arm is controllable and operable within the SPHERES Halo Guest Scientist framework. This test also verified the open loop control gains in the 1-g environment. The second objective of the 3 DoF test was to validate the dynamic simulation of a free-flying satellite (Section 5.4.2). Finally, the test was expected to provide a data set of inertial measurements for a free-floater with a moveable appendage in a relevant test environment.

Figure 5-25 shows the dynamic motion in action. The arm executed a sequence

of six repeated commands, paralleling the same reference commands as shown in simulation (Figure 5-23). These commands were sent every four seconds, and the arm provided feedback at intervals of about 0.105 seconds, or about 38 steps every four seconds. Although an open loop command sequence was generated, this platform can be used in conjunction with an active satellite to grasp objects such as a handle on a target SPHERES.

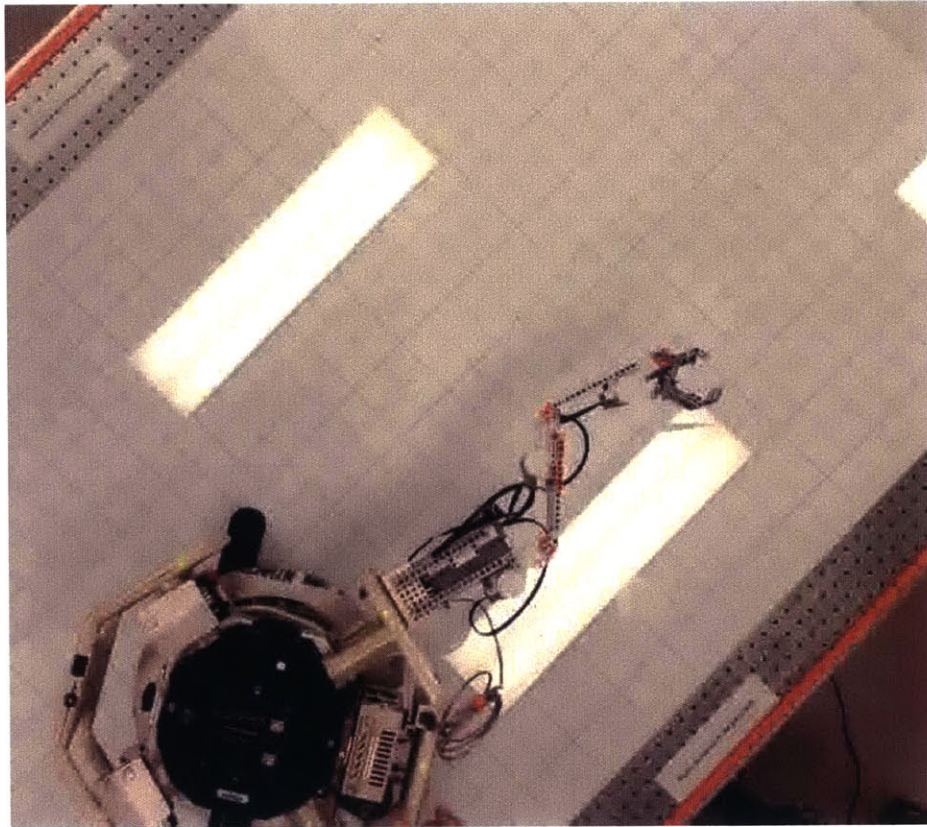


Figure 5-25: [Animation] Hardware Demonstration of the Lego Robotic Arm Executing the Same Open Loop Controls as Figure 5-23.

Figure 5-26 shows that the arm appropriately tracked the reference pattern using the internal motor encoders for feedback. This demonstrated the stable communication between the Lego Mindstorms Brick and the VERTIGO Avionics Stack. The communication line is needed to convert the signal from USB into the RS-485 communication protocol

understood by the NXT, and back. Moreover, this showed that the dynamic simulation should be extended to include the effects of the air carriage and table leveling since the drifting is non-negligible for a passive satellite. However, the coupled dynamics between the SPHERES and arm can be observed. This reaction torque can be seen to change the thrust vector directions and arm base location in the global frame, which motivates the need for further study to account for these changing system dynamics.

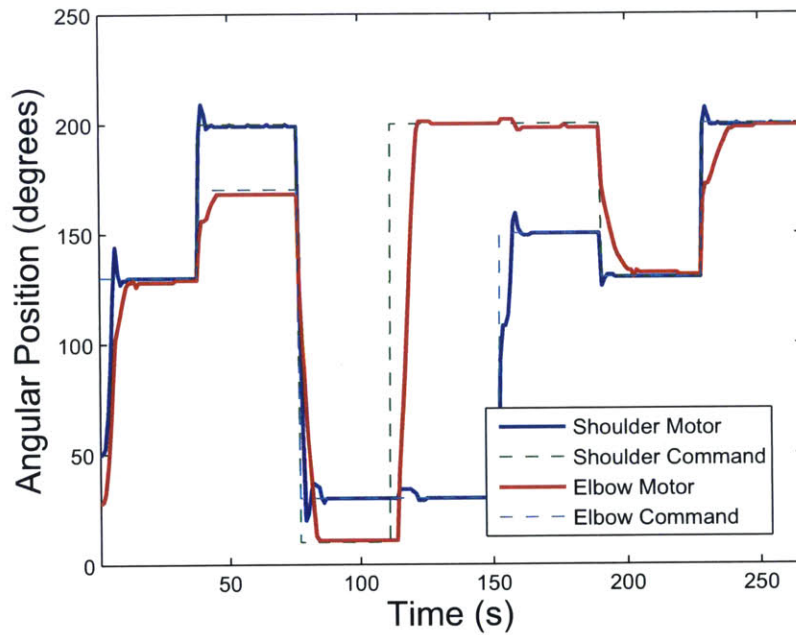


Figure 5-26: Results from the Lego Arm Hardware Demonstration Including Actuator Commands and Rotational Feedback

5.6 Future Work

Moving forward, the SPHERES ARM should continue development in order to push the state-of-the-art in the field of free-flying robotic manipulators. First, the third generation arm should be fabricated and tested to provide an experimental platform for analyzing novel simulation and control algorithms for satellite servicing applications. Next, a unified dynamic simulation should be constructed that can combine the inverse kinematics with the free-floating simulations described in this chapter. Ideally,

it would support complex, out-of-plane movement while interfacing with the SPHERES simulation to merge thruster firings and state controllers. Finally, an appropriate avionics bus must be developed, including the motor controllers, to interface effectively between the Halo expansion port and the three servos plus end-effector of the arm.

Chapter 6

Conclusion

In summary, this thesis has documented the delivery of a set of novel sensors and actuators for the SPHERES test facility on the International Space Station. The purpose of this research has been to advance the field of space robotics in a way that supports in-space docking and servicing of satellites. SPHERES as a testbed already provides real world impulsive thrusting and metrology accuracies in a dynamically authentic space environment. This research extends the capabilities of the SPHERES facility by enabling docking and reconfiguration of the satellites on-orbit, and delivering a prototype appendage for future grappling operations. The preliminary results presented here, plus data from future orbital test sessions, are intended to advance the state-of-the-art in satellite cooperation while providing risk reduction capabilities to early precursor servicing missions.

The thesis began by motivating the concept of satellite cooperation, both scientifically and economically, for particular classes of missions. The first contribution of this research was the introduction of a peripheral agnostic software architecture. This framework allows SPHERES to support a broader range of space technology payloads, including docking ports, control moment gyroscopes, lidars, and thermal cameras. The SPHERES Halo hardware was reviewed and the associated Halo Guest Scientist Program was established. The Halo Guest Scientist Program (GSP) is an extension of the existing SPHERES GSP. This Halo GSP was briefly compared with related robotics platforms, and the Halo GSP's niche in the robotics community was identified.

Guest scientists are invited to use the platform to conduct academic research, validate advanced control or autonomy algorithms, or develop new payloads for use in this testbed.

Next, a flight-qualified docking mechanism for the SPHERES was presented. This payload leverages previous work done at the MIT Space Systems Laboratory in order to develop a mass-efficient, low-risk, universal docking interface. With the SPHERES new ability to dock and undock in six degrees-of-freedom, different satellite assembly schemes and reconfigurable control algorithms may be investigated on-orbit. The chapter also documented a series of validation experiments that accelerated the technology readiness level of the docking port hardware. In both three and six degrees-of-freedom, the SPHERES-UDP assembly has been shown to be controllable and dock-able. Data gathered during a microgravity flight campaign have assisted with determining the mass characteristics and expected flight performance of the upgraded satellite system.

The thesis continued by evaluating the performance of a high precision relative sensor, integrated within the docking port actuator. The measurements provided by this system were shown to be sufficient to enable a high probability of docking between SPHERES, even in a zero gravity environment. Moreover, these measurements were integrated into an Unscented Kalman Filter, which provides a nonlinear estimate of the target satellite state. It was shown that this technique outperforms a traditional Extended Kalman Filter in highly dynamic scenarios, which encourages more academically interesting docking schemes to be explored.

Finally, various robotic arm concepts for SPHERES were investigated, simulated, prototyped and tested. Lego Mindstorms was employed for its a rapidly reconstructible interface, before the second generation SPHERES Ambulant Robotic Manipulator (ARM) was produced from COTS hardware. The physical prototypes provided a baseline architecture for dynamic simulation and control refinement. The solution to the restricted 3D inverse kinematics problem was presented, while a dynamic simulator for a planar manipulator was verified through testing on the SPHERES flat floor test facility.

In conclusion, this thesis has contributed original engineering solutions to the emerging field of satellite servicing, which itself challenges the operating strategies of the pre-existing satellite industrial complex. Future SPHERES scientists will use the developed tools to investigate the guidance, navigation and control of spacecraft across multiple levels of satellite cooperation. Areas of suggested future work by the SPHERES scientists include

- Achieving a successful and repeatable SPHERES docking on-orbit, especially in the presence of rotations, nutations, and drift
- Managing the fault detection, isolation and recovery logic during proximity operations
- Developing a hybrid estimator that combines relative sensing measurements with global metrology for a more reliable and precise state estimate
- Combining the controller for the ambulant robotic manipulator with a unified dynamic model of SPHERES

Designers of future servicing or assembly missions should use this thesis as a resource, and are invited to advance the Technology Readiness Level of their hardware or software through the Halo Guest Scientist Program. Teams of scientists at NASA or DARPA, working on projects as diverse as Phoenix or RSGS, can use the facility created at MIT to greatly improve the capabilities of their mission. Preliminary testing in microgravity on the ISS can be conducted as a technology pathfinder both in hardware and software. SPHERES and its associated payloads like the docking port now provide an avenue for accelerating modern algorithms related to the docking and servicing of satellites in space.

Bibliography

- [1] Frank J. Cepollina. Renewing the Partnership between Astronomy and the Human Space Flight Program : A National Strategy. 2008.
- [2] David Barnhart and T T O Program Manager. Phoenix : Initial Technical Elements and Interfaces (Satlet / PODs) Three basic concepts combine to create the Phoenix architecture Tele-presence control of Commercial Hosted Payloads service. pages 1–27, 2012.
- [3] NASA Smart SPHERES Tested Successfully on International Space Station, 2011.
- [4] DARPA's Phoenix Project To Recycle U.S. Satellites Using Satlets And The Zero Propellant Maneuver, 2012.
- [5] Bryan Patrick Mccarthy, Alvar Saenz-otero, and David W Miller. Flight Hardware Development for a Space-based Robotic Assembly and Servicing Testbed. (May), 2014.
- [6] Lorenzo Olivieri. *Development and Characterization of Standardized Docking System for Small Spacecraft*. PhD thesis, University of Padawa, 2015.
- [7] Brent Twedde. Computer vision based navigation for spacecraft proximity operations. *Massachusetts Institute of Technology*, (February), 2010.
- [8] Candarm Photograph, 2015. Accessed on: 2015-05-05. http://www.fansshare.com/community/uploads34/6784/candarm_vw/.

- [9] Dextre Photograph. Accessed on: 2015-05-05. <http://singularityhub.com/wp-content/uploads/2010/07/dextre-robot-in-space.jpg>.
- [10] JEM RMS Photograph. Accessed on: 2015-05-05 http://en.wikipedia.org/wiki/File:Japanese_Experiment_Module_exterior_-_cropped.jpg.
- [11] Robonaut 2, the Next Generation Dexterous Robot, 2010. Accessed on: 2015-05-05. http://www.nasa.gov/mission_pages/station/multimedia/robonaut_photos.html.
- [12] Academic Experiments Carried Out on ETS-VII Japanese Flying Space Robot, 2001. Accessed on 2015-05-05. <http://www.astro.mech.tohoku.ac.jp/~yoshida/ETS-VII/>.
- [13] Frank J. Cepollina and Jill McGuire. Viewpoint: Building a National Capability for On-Orbit Servicing.
- [14] European Robotic Arm, 2015. European Space Agency. Accessed on: 2015-05-05. http://www.esa.int/Our_Activities/Human_Spaceflight/.
- [15] Ranger Dexterous Manipulator Specifications. Technical report, University of Maryland Space Systems Laboratory.
- [16] DYMAFLEX University NanoSat Program (NS-7) Dynamic Manipulation Flight Experiment. University of Maryland Space Systems Laboratory. Accessed on: 2015-05-05. <http://www.ssl.umd.edu/html/nanosat.html>.
- [17] Bernard Kelm and J.A. Angielski. FRENDO: Pushing the Envelope of Space Robotics. *Space Research and Satellite Technology*, 2008.
- [18] David Pogue. PBS Making Stuff Wilder. NOVA. <https://www.youtube.com/watch?v=Q1MBIaNuLa8>.
- [19] Brook R. Sullivan, David Barnhart, Lisa Hill, Paul Oppenheimer, Bryan Benedict, Gerrit van Ommering, Laurie Chappell, John Ratti, and Peter Will.

- DARPA Phoenix Payload Orbital Delivery (POD) System: FedEx to GEO. *Proceedings of the AIAA SPACE Conference*, pages 1–14, 2013.
- [20] Christopher Michael Jewison, David Miller, and Alvar Saenz-otero. Reconfigurable Thruster Selection Algorithms for Aggregative Spacecraft Systems. (June), 2014.
- [21] Robotic Refueling Mission (RRM), 2015. National Aeronautics and Space Administration. Accessed on: 2015-05-05. http://www.nasa.gov/mission_pages/station/research/experiments/778.html.
- [22] David W. Miller, Swati Mohan, and Jason Budinoff. Assembly of a Large Modular Optical Telescope (ALMOST). 02139:70102H–70102H–11, 2008.
- [23] Alvar Saenz-Otero. SPHERES Specifications Sheet. 2006.
- [24] Phoenix Goals, 2012. DARPA Tactical Technology Office. Accessed on: 2015-05-05. http://www.darpa.mil/Our_Work/TTO/Programs/Phoenix/.
- [25] Doug Messier. DARPA Proposes \$127 Million in Space Spending, 2015. Parabolic Arc. Accessed on: 2015-05-05. <http://www.parabolicarc.com/2015/02/23/darpa-proposes-127-million-space-program-spending/>.
- [26] L Rodgers. *Concepts and technology development for the autonomous assembly and reconfiguration of modular space systems*. Master’s thesis, Massachusetts Institute of Technology, 2005.
- [27] Lance B Gatrell, William a Hoff, and Cheryl W Sklair. Robust image features: concentric contrasting circles and their image extraction, 1992.
- [28] Martin a Fischler and Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with. *Communications of the ACM*, 24:381–395, 1981.
- [29] Ramesh Jain, Rangachar Kasturi, and Bg Schunck. Machine vision. page 549, 1995.

- [30] Emil Fresk and George Nikolakopoulos. Full Quaternion Based Attitude Control for a Quadrotor. *European Control Conference (ECC)*, 2013.
- [31] Sebastien Xamb Descamps. Euler and the Dynamics of Rigid Bodies. *Quaderns D'Historia de l'Enginyeria*, Volum IX, 2008.
- [32] F. Landis Markley. Attitude Error Representations for Kalman Filtering. *AIAA Journal of Guidance, Control, and Dynamics*, 26(2), 2002.
- [33] Rush D Robinett Hanspeter Schaub and John L Junkins. Adaptive External Torque Estimation By Means of Tracking A Lyapunov Function. 1996.
- [34] The Shuttle Remote Manipulator System – The Canadarm. MacDonald Dettwiler Space and Advanced Robotics. http://www.ieee.ca/millennium/canadarm/canadarm_technical.html.
- [35] Canadarm 2 Moving Payloads in Space. Canadian Space Agency. Accessed on: 2015-05-05. http://resources.yesican-science.ca/trek/canadarm2/history/comparison_student.html.
- [36] Canadarm and Canadarm2 - Comparative table, 2002. Canadian Space Agency. Accessed on: 2015-05-05. <http://www.asc-csa.gc.ca/eng/iss/canadarm2/c1-c2.asp>.
- [37] Dextre, 2008. National Aeronautics and Space Administration. Accessed on: 2015-05-05. http://www.nasa.gov/mission_pages/station/structure/elements/dextre.
- [38] Chris Bergin. Dextre, the Canadian robot, waits begins operational service. NASA Spaceflight. <http://www.nasaspaceflight.com/2010/07/dextre-the-canadian-robot-begins-operational-service/>.
- [39] Japanese Experiment Module Remote Manipulator System, 2008. Japan Aerospace Exploration Agency. Accessed on: 2015-05-05. <http://iss.jaxa.jp/en/kibo/about/kibo/rms/>.

- [40] Jaime Marshik. STS-124/1J FD 08 Execute Package. *JEDI (Joint Execute package Development and Integration)*, 2008.
- [41] Robonaut 2 NASA Facts. National Aeronautics and Space Administration. Accessed on: 2015-05-05. http://www.nasa.gov/sites/default/files/files/Robonaut2_508.pdf.
- [42] Robonaut R2. NASA Johnson Space Center. <http://robonaut.jsc.nasa.gov/ISS/>.
- [43] ETS-VII (Engineering Test Satellite VII) / Kiku-7, 2002. Sharing Earth Observation Resources. Accessed on: 2015-05-05. <https://directory.eoportal.org/web/eoportal/satellite-missions/e/ets-vii>.
- [44] Andrew Ogilvie. Autonomous Satellite Servicing Using the Orbital Express Demonstration Manipulator System. *DARPA*, 2008.
- [45] H.J. Cruijssen. The European Robotic Arm: A High-Performance Mechanism Finally on its way to Space. *ESMATS*, 2014.
- [46] SpaceNews Editor. ESA To Hear Proposal To Finance Life-Extending Space Tug. *SpaceNews*, 2004.
- [47] Enrico Sabelli. Categorizing Admittance Control Parameters for the Ranger 8-DoF Tele-Operated Space Manipulator.
- [48] Glen Henshaw and Bernard Kelm. DARPA Phoenix Overview and Risk Reduction Plans, 2014.
- [49] LEGO MINDSTORMS Education NXT Gyrobot, 2014. Lego Education. <https://education.lego.com/en-us/lesi/afterschool/lego-space/nxt-gyrobot>.
- [50] Types of Winches: Planetary Gear, Worm Gear, and Spur Gear Title. Promark. Accessed on: 2015-05-05. <http://blog.promarkoffroad.com/2009/10/23/planetary-worm-spur-gear-electric-winch/>.

- [51] Wenfu Xu, Cheng Li, Bin Liang, Yu Liu, and Yangsheng Xu. The Cartesian Path Planning of Free- Floating Space Robot using Particle Swarm Optimization. *International Journal of Advanced Robotic Systems*, 5(3):1, 2008.
- [52] Duncan Lee Miller and David W Miller. Development of Resource-Constrained Sensors and Actuators for In-Space Satellite Docking and Servicing. (June), 2015.
- [53] (DARPA) Defense Advanced Research Projects Agency and (TTO) Tactical Technology Office. Robotic On-Orbit Servicing Capability with Commercial Transition. Technical report, 2011.
- [54] P.S. Schenker. A composite manipulator utilizing rotary piezoelectric motors: new robotic technologies for Mars in-situ planetary science. *Jet Propulsion Laboratory*, 1998.
- [55] Brent Edward Tweddle and David W Miller. *Computer Vision-Based Localization and Mapping of an Unknown, Uncooperative and Spinning Target for Spacecraft Proximity Operations*. PhD thesis, Massachusetts Institute of Technology, 2013.
- [56] David Charles Sternberg, Prof David Miller, and Alvar Saenz-otero. *Development of an Incremental and Iterative Risk Reduction Facility for Robotic Servicing and Assembly Missions*. Masters thesis, Massachusetts Institute of Technology, 2014.
- [57] The Robotic Arm (RA) and Robotic Arm Camera (RAC). NASA Jet Propulsion Lab. Accessed on: 2015-05-05. <http://mars.jpl.nasa.gov/msp98/mvacs/instruments/ra.html>.
- [58] James Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. *Stanford University*, 2006.
- [59] Zach Bailey. Determining the Value of On-Orbit Telescope Servicing. 2010.

- [60] Dustin Berkovitz and David W Miller. *System Characterization and Online Mass Property Identification of the SPHERES Formation Flight Testbed*. PhD thesis, Massachusetts Institute of Technology, 2007.
- [61] Matthew Knutson, David Miller, and Sungyung Lim. *Fast Star Tracker Centroid Algorithm for High Performance CubeSat with Air Bearing Validation*. Number June. 2012.
- [62] Chris Jewison and Duncan Miller. RGA IMU Data Analysis. Technical report, Massachusetts Institute of Technology Space Systems Laboratory, 2015.
- [63] Simon Nolet and David W Miller. Development of a Guidance , Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite. *Control*, (June), 2007.
- [64] Gregory Eslinger. *Dynamic Programming Applied to Electromagnetic Satellite Actuation*. PhD thesis, Massachusetts Institute of Technology, 2013.
- [65] Alvar Saenz-otero and David W Miller. *Design Principles for the Development of Space Technology Maturation Laboratories Aboard the International Space Station*. PhD thesis, Massachusetts Institute of Technology, 2005.

Appendix A

Additional Mathematical Definitions

A.1 Additional Mathematical Definitions

A.1.1 Quaternion

A unit quaternion is a four element representation of the attitude of an object. It consists of vector and scalar elements that are related to the Euler axis and Euler angle of rotation as follows. Also as a unit quaternion, it obeys the unit length constraint.

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_v \\ q_4 \end{bmatrix} = \begin{bmatrix} \mathbf{e} \sin \phi/2 \\ \cos \phi/2 \end{bmatrix} \quad (\text{A.1})$$

$$|\mathbf{q}|^2 = |\mathbf{q}_v|^2 + q_4^2 = 1 \quad (\text{A.2})$$

Quaternion multiplication is a noncommutative operation that can be defined as either a matrix-vector product or compacted in vector notation. We have defined \mathbf{v}_a and \mathbf{v}_b as the vector parts of the quaternions \mathbf{q}_a and \mathbf{q}_b with q_4 as the scalar element

of each.

$$\mathbf{q}_a \otimes \mathbf{q}_b = \begin{bmatrix} q_{a4} & -q_{a3} & q_{a2} & q_{a1} \\ q_{a3} & q_{a4} & -q_{a1} & q_{a2} \\ -q_{a2} & q_{a1} & q_{a4} & q_{a3} \\ -q_{a1} & -q_{a2} & -q_{a3} & q_{a4} \end{bmatrix} \begin{bmatrix} q_{b1} \\ q_{b2} \\ q_{b3} \\ q_{b4} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_a \times \mathbf{v}_b + q_{4a}\mathbf{v}_b + q_{4b}\mathbf{v}_a \\ q_{4a}q_{4b} - \mathbf{v}_a \bullet \mathbf{v}_b \end{bmatrix} \quad (\text{A.3})$$

We also introduce the transform from 1-2-3 Euler angles to quaternions which is needed for measurement noise generation. The solution is provided in [58].

$$\mathbf{q}_{123}(\phi, \theta, \psi) = \begin{bmatrix} c_{\phi/2}c_{\theta/2}c_{\psi/2} + s_{\phi/2}s_{\theta/2}s_{\psi/2} \\ -c_{\phi/2}s_{\theta/2}s_{\psi/2} + s_{\phi/2}c_{\theta/2}c_{\psi/2} \\ c_{\phi/2}c_{\phi/2}s_{\theta/2} + s_{\phi/2}c_{\theta/2}s_{\psi/2} \\ c_{\phi/2}c_{\theta/2}s_{\psi/2} - s_{\phi/2}s_{\theta/2}c_{\psi/2} \end{bmatrix} \quad (\text{A.4})$$

A.1.2 Inertia Definition

The inertia tensor collects the mass moments of inertia of a rigid body in matrix form. The moment of inertia I_{xx} , I_{yy} , I_{zz} measure the resistance to rotational accelerations along the x,y,z axes. The products of inertia, the off-diagonal elements of the inertia tensor, are a measure of the induced acceleration around the second axis provided an acceleration input in the first axis. The inertia tensor is a symmetric matrix.

$$(I_{xx})_O = \int_m (y^2 + z^2)dm \quad (I_{yy})_O = \int_m (x^2 + z^2)dm \quad (I_{zz})_O = \int_m (x^2 + y^2)dm$$

$$(I_{xy})_O = (I_{yx})_O = \int_m (xy)dm \quad (I_{xz})_O = (I_{zx})_O = \int_m (xz)dm \quad (I_{yz})_O = (I_{zy})_O = \int_m (yz)dm$$

A.1.3 Full Linearization of the Euler Dynamics

Proceeding with the linearization about $\hat{\omega}$ of Euler's rotational dynamics, we can find the Jacobian, evaluate it at the current best estimate and multiply by the inverse of the inertia tensor to solve for the rotational acceleration. We have verified that this produces the same results as first multiplying by the inverted inertia and then taking

the Jacobian. In this way, the linearized \mathbf{A}_ω matrix is obtained.

$$\dot{\omega} \approx \mathbf{J}^{-1} \begin{bmatrix} I_{xy}\omega_3 - I_{xz}\omega_2 & I_{zz}\omega_3 - I_{yy}\omega_3 - 2I_{yz}\omega_2 - I_{xz}\omega_1 & I_{zz}\omega_2 + 2I_{yz}\omega_3 - I_{yy}\omega_2 + I_{xy}\omega_1 \\ I_{xx}\omega_3 + 2I_{xz}\omega_1 + I_{yz}\omega_2 - I_{zz}\omega_3 & I_{yz}\omega_1 - I_{xy}\omega_3 & I_{xx}\omega_1 - 2I_{xz}\omega_3 - I_{xy}\omega_2 - I_{zz}\omega_1 \\ I_{yy}\omega_2 - 2I_{xy}\omega_1 - I_{xx}\omega_2 - I_{yz}\omega_3 & I_{yy}\omega_1 + 2I_{xy}\omega_2 - I_{xx}\omega_1 + I_{xz}\omega_3 & I_{xz}\omega_2 - I_{yz}\omega_1 \end{bmatrix}_{\omega=\dot{\omega}} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \quad (\text{A.5})$$

$$\dot{\omega} \approx \mathbf{A}_\omega \omega \quad (\text{A.6})$$

Appendix B

Source Code

This appendix contains source code used to simulate the performance of the nonlinear estimator for the docking port sensor. A refined version of these algorithms was converted to C using the Matlab Real-Time Workshop for implementation on the hardware.

B.1 Multiplicative Extended Kalman Filter

This file implements the Multiplicative Extended Kalman Filter described in Section 4.4.

```
1 function [x2, Q2] = mekf(x1, Q1, dt, r_meas, q_meas, eta, type2)
2 % this function runs a multiplicative extended kalman filter
3 % x1 is the previous estimate, P1 previous covariance, dt the
4 % discretization step size, r_meas the observed position vector, q_meas
   is
5 % the measured quaternion, eta is the confidence in the measurement
6 % type2 denotes if a type2 error was generated (no measurement)
7
8 % Allocate memory
9 x2 = double(zeros(13,1));
10 Q2 = double(zeros(12,12));
```

```

11 L = zeros(12,6);
12
13 % Relative Parameters
14 d = 0.016; %size of fiducial markers
15 d = 1;
16 m = 4.16; %mass of the SPHERES assembly
17
18 %SPHERES inertia with UDP and VERTIGO
19 Ixx = 302328.2863;
20 Iyy = 640602.2985;
21 Izz = 567071.8412;
22 Ixy = -6600.8319;
23 Ixz = -3662.2998;
24 Iyz = 2745.6257;
25 J = 10^-7*[ Ixx,    -Ixy,    -Ixz;
26             -Ixy,    Iyy,    -Iyz;
27             -Ixz,    -Iyz,    Izz];
28
29 % Intensity of the Process Noise
30 % Wc = 10e-8*[eye(3), zeros(3);
31 %           zeros(3),    0.002*eye(3)];
32 Wc = 10e-4*[eye(3), zeros(3);
33             zeros(3),    0.002*eye(3)];
34
35 % Intensity of the Measurement Noise
36 R = [0.05*eta*eye(3), zeros(3);
37       zeros(3),    1.5*eta.*eye(3)];
38 % R = zeros(6);
39
40 % Measurement Matrix
41 C = [eye(3), zeros(3,9);
42       zeros(3,6), eye(3), zeros(3)];
43
44 % External Forces and Torques
45 F = zeros(3,1);
46 T = zeros(3,1);

```



```

47
48 x = [x1(1:6); [0;0;0]; x1(11:13)];
49 q = x1(7:10);
50
51 % Prediction Step of the MEKF
52 [x_prop, q_prop, Ak, Wk] = f1(x,q,dt,m,J,F,T, Wc); %state predict
53 Qa = Ak*Q1*Ak' + Wk;
54 % Qa = Ak*P1*Ak';
55 Qa = 0.5*(Qa+Qa'); %enforce symmetry
56
57 % If type2, no measurement recieved
58 if type2
59     Q2 = Qa;
60
61     da = x_prop(7:9);
62     dq2 = [8*da; 16-da'*da] ./ (16+da'*da);
63     dq2 = dq2 / sqrt(dq2'*dq2);
64     %     q2 = qmult1([-1 -1 -1 1]'.*dq2,q); % EDITed and change
65     %     q2 = qmult1(q,dq2);
66     q2 = qmult1(dq2,q); %old
67
68     x2 = [x_prop(1:6); q_prop; x_prop(10:12)];
69     return
70 end
71
72
73 % %Update Step of MEKF
74 % Convert measurement to modified rodriguez parameters
75 dq = qmult1(q_meas, [-1; -1; -1; 1].*q); %previous
76
77 ap_meas = 4*dq(1:3)/(1+dq(4));
78 y_meas = [r_meas; ap_meas];
79 innov = (y_meas - C*x_prop);
80
81 % Calculate Gain Matrix
82 L = Qa*C'*invmat1(C*Qa*C'+R);

```

```

83
84 xp = x_prop + L*innov;
85 % Alternative Version of Covariance Update
86 Q2 = (eye(12)-L*C)*Qa*(eye(12)-L*C)' + L*R*L';
87 % Q2 = (eye(12) - L*C)*Qa;
88 Q2 = 0.5*(Q2'+Q2); % make covariance symmetric
89
90 % Reset quaternion
91 da = xp(7:9);
92 dq2 = [8*da; 16-da'*da] ./ (16+da'*da);
93 dq2 = dq2 / sqrt(dq2'*dq2);
94 q2 = qmult1(dq2,q);
95
96 x2(1:6) = xp(1:6);
97 x2(7:10) = q2;
98 x2(11:13) = xp(10:12);
99 end
100
101
102 function [x2, q2, Ak, Wk] = f1(x,q,dt,m,J,F,T,Wc)
103 % f1 is the nonlinear propagation function
104 % x is the 12 state, q quaternion, dt descritized time step, m mass, J
105 % inertia, F/T forces/torques, Wc is the continous process noise
106
107 x2 = zeros(12,1);
108 q2 = zeros(4,1);
109
110 r = x(1:3);
111 v = x(4:6);
112 a = x(7:9); % should be zero
113 w = x(10:12);
114
115 iJ = inv(J);
116 Ixx = J(1,1);
117 Ixy = -J(1,2);
118 Ixz = -J(1,3);

```

```

119 Iyx = -J(2,1);
120 Iyy = J(2,2);
121 Iyz = -J(2,3);
122 Izz = J(3,3);
123 w1 = w(1);
124 w2 = w(2);
125 w3 = w(3);
126
127 % Cross Product Matrix
128 w_cross = [ 0      -w3    w2;
129             w3     0      -w1;
130            -w2     w1     0];
131
132 % Effective Quaternion for Quaternion Product (from target to observer)
133 O = [ 0,      -w3,     w2,     w1;
134       w3,     0,      -w1,     w2;
135       -w2,     w1,     0,      w3;
136       -w1,    -w2,    -w3,     0];
137
138 % Propagate nonlinear estimate
139 x2(1:3) = x(1:3) + dt*x(4:6);
140 x2(4:6) = x(4:6);
141
142 q2 = expm(0.5*O.*dt)*q;
143 % dq = 0.5*O*q; %dq/dt
144 dq = qmult1(q2, [-1; -1; -1; 1].*q);
145 a2 = 4*dq(1:3)/(1+dq(4));
146 x2(7:9) = x(7:9) + a2;
147
148 % Nonlinear angular acceleration
149 wd1 = w3*(Ixy*w1 - Iyy*w2 + Iyz*w3) - w2*(Ixz*w1 + Iyz*w2 - Izz*w3);
150 wd2 = w1*(Ixz*w1 + Iyz*w2 - Izz*w3) - w3*(Ixy*w2 - Ixx*w1 + Ixz*w3);
151 wd3 = w2*(Ixy*w2 - Ixx*w1 + Ixz*w3) - w1*(Ixy*w1 - Iyy*w2 + Iyz*w3);
152 wd = -iJ*[wd1; wd2; wd3];
153 % wd = -inv(J)*cross(w, J*w);
154 x2(10:12) = x(10:12) + dt*wd;

```

```

155 % x2(10:12) = x(10:12); % DEBUG
156
157 % Calculate linearized A matrix
158 %Translational Continous A
159 Atc = [ zeros(3)    eye(3);
160         zeros(3)    zeros(3)];
161
162 %Rotational Continous A
163 %a_dot = [1/2*wcross 1/4]*[a; w]
164 % MRP dynamics
165 A1 = [1/2*w-cross 1/4*eye(3)];
166
167 % Linearized Angular velocity dynamics
168 A2 = [ ...
169       Ixy*w3 - Ixz*w2,    Izz*w3 - Iyy*w3 - 2*Iyz*w2 - Ixz*w1,    Ixy*w1 -
170       Iyy*w2 + 2*Iyz*w3 + Izz*w2;
171       Ixx*w3 + 2*Ixz*w1 + Iyz*w2 - Izz*w3,    Iyz*w1 - Ixy*w3,    Ixx*w1 -
172       Ixy*w2 - 2*Ixz*w3 - Izz*w1;
173       Iyy*w2 - 2*Ixy*w1 - Ixx*w2 - Iyz*w3,    2*Ixy*w2 - Ixx*w1 + Ixz*w3 +
174       Iyy*w1,    Ixz*w2 - Iyz*w1];
175 A2 = -iJ * A2;
176
177 Arc = [ A1;
178         zeros(3,3) A2];
179
180 A = [ Atc    zeros(6,6);
181       zeros(6,6) Arc];
182
183 Bw = [ zeros(3,3)    zeros(3,3);
184       eye(3,3)    zeros(3,3);
185       zeros(3,3)    zeros(3,3);
186       zeros(3,3)    eye(3,3)];
187
188 % Find discrete time A, and W
189 nA = size(A,1);
190 S = [-A Bw*Wc*Bw'; zeros(nA) A'];

```

```

188 CC = expm(S*dt);
189 Ak = CC(nA+1:2*nA, nA+1:2*nA)'; % descritized linearization about k-1
190 Wk = Ak*CC(1:nA, nA+1:2*nA);
191 % Wk = Bw*Wc*Bw'*dt;
192
193 end
194
195 function qout = qmult1(q1, q2)
196 % multiplies 2 quaternions
197
198 qout = zeros(4,1);
199
200 vq1 = q1(1:3);
201 kq1 = q1(4);
202 vq2 = q2(1:3);
203 kq2 = q2(4);
204
205 q1_cross = [0, -q1(3), q1(2);
206             q1(3), 0, -q1(1);
207             -q1(2), q1(1), 0];
208
209 qout(1:3) = kq1.*vq2 + kq2.*vq1 + q1_cross*vq2;
210 qout(4) = kq1*kq2 - vq1'*vq2;
211 qout = qout/norm(qout);
212 end
213
214 function B = invmat1(A)
215 %inverts a symmetric matrix in numerically stable way
216 mat = 0.5*(A+A');
217 [q r] = qr(mat);
218 B = r \ q';
219 end

```

B.2 Unscented Kalman Filter

This file implements the Unscented Kalman Filter described in Section 4.5.

```
1 function [x2, Q2] = ukf( x1, Q1, dt, r_meas, q_meas, eta, type2)
2 % UKF implements an unscented kalman filter
3 % x1 is the previous estimate, P1 previous covariance, dt the
4 % discretization step size, r_meas the observed position vector, q_meas
   is
5 % the measured quaternion, eta is the confidence in the measurement
6 % type2 denotes if a type2 error was generated (no measurement)
7
8 x2 = x1;
9 Q2 = Q1;
10 Qa = Q1;
11 L = 5;
12
13 % Allocate memory
14 x2 = double(zeros(13,1));
15 Q2 = double(zeros(12,12));
16 L = zeros(12,6);
17
18 % Relative Parameters
19 d = 0.016; %size of fiducial markers
20 d = 1;
21 m = 4.16; %mass of SPHERES
22
23 %SPHERES inertia with UDP and VERTIGO
24 Ixx = 302328.2863;
25 Iyy = 640602.2985;
26 Izz = 567071.8412;
27 Ixy = -6600.8319;
28 Ixz = -3662.2998;
29 Iyz = 2745.6257;
30 J = 10^-7*[ Ixx,    -Ixy,    -Ixz;
```

```

31         -Ixy,   Iyy,   -Iyz;
32         -Ixz,   -Iyz,   Izz];
33
34 % Intensity of the Process Noise
35 % Wc = 0.000001*[0.04*eye(3), zeros(3);
36 %         zeros(3),          eye(3)];
37 % Wc = 10e-8*[eye(3), zeros(3);
38 %         zeros(3),          0.002*eye(3)];
39 Wc = 10e-4*[eye(3), zeros(3);
40         zeros(3),          0.002*eye(3)];
41
42 % Intensity of the Measurement Noise
43 % R = 0.000000000000001*[eta*eye(3), zeros(3);
44 %         zeros(3),          eta.*eye(3)];
45 R = [0.05*eta*eye(3), zeros(3);
46         zeros(3),          1.5*eta.*eye(3)];
47 % R = zeros(6);
48
49 % Measurement Matrix
50 C = [eye(3), zeros(3,9);
51         zeros(3,6), eye(3), zeros(3)];
52
53 % Forces and Torques
54 F = zeros(3,1);
55 T = zeros(3,1);
56
57 x = [x1(1:6); [0;0;0]; x1(11:13)];
58 q = x1(7:10);
59
60 % Propagate x to get Wk
61 [x-prop, q-prop, Ak,Wk] = f1(x,q,dt,m,J,F,T, Wc); %state predict
62 %
63 % If type2, no measurement recieved
64 if type2
65     Q2 = Qa;
66

```

```

67     da = x_prop(7:9);
68     dq2 = [8*da; 16-da'*da] ./ (16+da'*da); %%corrected!!!
69     dq2 = dq2 / sqrt(dq2'*dq2);
70     %     q2 = qmult1([-1 -1 -1 1]'.*dq2,q); % EDITed and change
71     %     q2 = qmult1(q,dq2);
72     q2 = qmult1(dq2,q); %old
73
74     x2 = [x_prop(1:6); q_prop; x_prop(10:12)];
75     return
76 end
77
78 Xi = get_sigma(x,Q1);
79
80 % Propagate each sigma point, average together
81 xk = zeros(size(Xi));
82 xa = zeros(12,1);
83 for i = 1:size(Xi,2)
84     [xk(:,i), qa, Ak, ~] = f1(Xi(:,i),q,dt,m,J,F,T, Wc); %state predict
85     xa = xa + xk(:,i);
86 end
87 xa = xa/size(Xi,2);
88
89 % Get propagated Covariance for all sigma points, average together
90 Qa = zeros(size(xa*xa'));
91 for i = 1:size(Xi,2)
92     Qa = Qa + (xk(:,i) - x_prop)*(xk(:,i) - x_prop)';
93 end
94 Qa = Qa/size(Xi,2) + Wk;
95
96 % Get second set of sigma points
97 Xai = get_sigma(xa,Qa);
98 yk = zeros(size(C*Xai(:,1)));
99 yki = zeros(size(yk));
100 for i = 1:size(Xai,2)
101     yki(:,i) = C*Xai(:,i);
102     yk = yk + yki(:,i);

```



```

103 end
104 yk = yk/size(Xai,2);
105
106 % Find Qyy, Qxy
107 Qyy = zeros(size(yk*yk'));
108 Qxy = zeros(size(xa*yk'));
109 for i = 1:size(Xai,2)
110     Qyy = Qyy + (yki(:,i) - yk)*(yki(:,i) - yk)';
111     Qxy = Qxy + (Xai(:,i) - xa)*(yki(:,i) - yk)';
112 end
113 Qyy = Qyy/(size(Xi,2)) + R;
114 Qxy = Qxy/(size(Xi,2));
115
116 % Convert measurement to modified rodriguez parameters
117 dq = qmult1(q_meas, [-1; -1; -1; 1].*q); %previous
118 ap_meas = 4*dq(1:3)/(1+dq(4));
119 y_meas = [r_meas; ap_meas];
120
121 L = Qxy*inv(Qyy);
122 xp = xa + L*(y_meas - yk);
123 Q2 = Qa - L*Qyy*L';
124
125 % Reset quaternion
126 da = xp(7:9);
127 dq2 = [8*da; 16-da'*da] ./ (16+da'*da);
128 dq2 = dq2 / sqrt(dq2'*dq2);
129 q2 = qmult1(dq2,q);
130
131 x2(1:6) = xp(1:6);
132 x2(7:10) = q2;
133 x2(11:13) = xp(10:12);
134 end
135
136 function [Xi] = get_sigma(x, Q)
137 % Generates sigma points around the current estimate,x a 12x1 vector
138 n = size(x,1);

```

```

139 Y = x(:,ones(1,numel(x)));
140 x_l = Y - chol(n*Q)';
141 x_r = Y + chol(n*Q)';
142 Xi = [x_l x_r];
143
144 end
145
146
147 function [x2, q2, Ak, Wk] = f1(x,q,dt,m,J,F,T,Wc)
148 % f1 is the nonlinear propagation function
149 % x is the 12 state, q quaternion, dt descritized time step, m mass, J
150 % inertia, F/T forces/torques, Wc is the continous process noise
151
152 x2 = zeros(12,1);
153 q2 = zeros(4,1);
154
155 r = x(1:3);
156 v = x(4:6);
157 a = x(7:9); % should be zero
158 w = x(10:12);
159
160 iJ = inv(J);
161 Ixx = J(1,1);
162 Ixy = -J(1,2);
163 Ixz = -J(1,3);
164 Iyx = -J(2,1);
165 Iyy = J(2,2);
166 Iyz = -J(2,3);
167 Izz = J(3,3);
168 w1 = w(1);
169 w2 = w(2);
170 w3 = w(3);
171
172 % Cross Product Matrix
173 w_cross = [ 0      -w3    w2;
174            w3     0     -w1;

```

```

175         -w2    w1    0];
176
177 % Effective Quaternion for Quaternion Product
178 O = [ 0,    -w3,    w2,    w1;
179       w3,    0,    -w1,    w2;
180       -w2,    w1,    0,    w3;
181       -w1,    -w2,    -w3,    0];
182
183 % Propagate nonlinear estimate
184 x2(1:3) = x(1:3) + dt*x(4:6);
185 x2(4:6) = x(4:6);
186
187 q2 = expm(0.5*O.*dt)*q;
188 % dq = 0.5*O*q; %dq/dt
189 dq = qmult1(q2, [-1; -1; -1; 1].*q);
190 a2 = 4*dq(1:3)/(1+dq(4));
191 x2(7:9) = x(7:9) + a2;
192
193 % Nonlinear angular acceleration
194 wd1 = w3*(Ixy*w1 - Iyy*w2 + Iyz*w3) - w2*(Ixz*w1 + Iyz*w2 - Izz*w3);
195 wd2 = w1*(Ixz*w1 + Iyz*w2 - Izz*w3) - w3*(Ixy*w2 - Ixx*w1 + Ixz*w3);
196 wd3 = w2*(Ixy*w2 - Ixx*w1 + Ixz*w3) - w1*(Ixy*w1 - Iyy*w2 + Iyz*w3);
197 wd = -iJ*[wd1; wd2; wd3];
198 % wd = -inv(J)*cross(w, J*w);
199 x2(10:12) = x(10:12) + dt*wd;
200 % x2(10:12) = x(10:12); % DEBUG
201
202 % Calculate linearized A matrix
203 %Translational Continous A
204 Atc = [ zeros(3)    eye(3);
205         zeros(3)    zeros(3)];
206
207 %Rotational Continous A
208 %a_dot = [1/2*wcross 1/4]*[a; w]
209 % MRP dynamics
210 A1 = [1/2*w_cross 1/4*eye(3)];

```

```

211
212 % Linearized Angular velocity dynamics
213 A2 = [ ...
214     Ixy*w3 - Ixz*w2,     Izz*w3 - Iyy*w3 - 2*Iyz*w2 - Ixz*w1,     Ixy*w1 -
           Iyy*w2 + 2*Iyz*w3 + Izz*w2;
215     Ixx*w3 + 2*Ixz*w1 + Iyz*w2 - Izz*w3,     Iyz*w1 - Ixy*w3,     Ixx*w1 -
           Ixy*w2 - 2*Ixz*w3 - Izz*w1;
216     Iyy*w2 - 2*Ixy*w1 - Ixx*w2 - Iyz*w3,     2*Ixy*w2 - Ixx*w1 + Ixz*w3 +
           Iyy*w1,     Ixz*w2 - Iyz*w1];
217 A2 = -iJ * A2;
218
219 Arc = [ A1;
220         zeros(3,3) A2];
221
222 A = [ Atc     zeros(6,6);
223       zeros(6,6) Arc];
224
225 Bw = [ zeros(3,3)     zeros(3,3);
226       eye(3,3)      zeros(3,3);
227       zeros(3,3)     zeros(3,3);
228       zeros(3,3)     eye(3,3)];
229
230 % Find discrete time A, and W
231 % nA = size(A,1);
232 % S = [-A Bw*Wc*Bw'; zeros(nA) A'];
233 % CC = expm(S*dt);
234 % Ak = CC(nA+1:2*nA, nA+1:2*nA)'; % discretized linearization about k-1
235 % Wk = Ak*CC(1:nA, nA+1:2*nA);
236 Ak = zeros(size(A)); % Ak is not used
237 Wk = Bw*Wc*Bw'*dt;
238
239 end
240
241
242 function qout = qmult1(q1, q2)
243 % multiplies 2 quaternions

```

```

244
245 qout = zeros(4,1);
246
247 vq1 = q1(1:3);
248 kq1 = q1(4);
249 vq2 = q2(1:3);
250 kq2 = q2(4);
251
252 q1_cross = [0, -q1(3), q1(2);
253             q1(3), 0, -q1(1);
254             -q1(2), q1(1), 0];
255
256 qout(1:3) = kq1.*vq2 + kq2.*vq1 + q1_cross*vq2;
257 qout(4) = kq1*kq2 - vq1'*vq2;
258 qout = qout/norm(qout);
259 end
260
261 function B = invmat1(A)
262 %inverts a symmetric matrix in numerically stable way
263 mat = 0.5*(A+A');
264 [q r] = qr(mat);
265 B = r \ q';
266 end

```