Reconfiguration Methods for On-orbit Servicing, Assembly, and Operations with Application to Space Telescopes

by

Swati Mohan

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

Masters of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007
[June 2007]

© Massachusetts Institute of Technology 2007. All rights reserved.

Author .....
Department of Aeronautics and Astronautics
May 25, 2007

Certified by .....
David W. Miller
Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Jaime Peraire
Chairman, Committee on Graduate Students

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUL 11 2007
LIBRARIES

ARCHIVES

Reconfiguration Methods for On-orbit Servicing, Assembly, and Operations with Application to Space Telescopes

by

Swati Mohan

Submitted to the Department of Aeronautics and Astronautics
on May 25, 2007, in partial fulfillment of the
requirements for the degree of
Masters of Science in Aeronautics and Astronautics

Abstract

Reconfiguration is an important characteristic in furthering on-orbit servicing, assembly, and operations. Previous work has focused on large assemblers manipulating small payloads, where the dynamics of the assembler is not significantly changed. This work seeks to identify the impact of reconfiguration on maneuver performance. Reconfiguration is considered in two categories: implementation and application. Implementation of reconfiguration consisted of developing a method for defining and updating a configuration, implementation on the SPHERES testbed, and execution of tests (in simulation and on the International Space Station) to assess the control performance improvement after reconfiguration.

Four applications were considered in this work, two hardware applications and two systems applications modeled through simulation. The objective of the SWARM application was to demonstrate autonomous assembly capability through docking and undocking maneuvers. The objective of the SIFFT application was to demonstrate formation reconfiguration capability, through the expansion and rotation of an equilateral triangle of three satellites.

The objective of the systems applications was to determine the impact of reconfiguration in a larger mission context. One application, Mass Property Update, considered how the choice of method for obtaining the mass property information impacts operations. The other application, Modularity Analysis, considered how the implementation of modularity is driven by the mission objectives.

Overall, this work has served to demonstrate the control impact of reconfiguration though implementation on the SPHERES testbed. This implementation was used on two hardware applications to determine the performance of reconfiguration for assembly and formation reconfiguration missions. Also, the impact of reconfiguration has been studied in the broader systems context. The choice of method of mass property update was demonstrated to have an impact on operations, in terms of reliability and mass. Finally, the method incorporation of modularity for purposes of on-orbit servicing and assembly was demonstrated to be driven by mission design parameters.

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

Acknowledgments

The author would like to thank the significant contributors to this work. Funding for this work has come from the following sources:

- NASA Jet Propulsion Laboratory Research and Development Fund
- NASA Harriet G. Jenkins Pre-doctoral Fellowship program
- NASA Small Business Innovation Research grant for SWARM
- NASA Goddard Space Flight Center for SIFFT
- NASA Autonomous Assembly and Reconfiguration program

Thank you also to the NASA Marshall Space Flight Center for providing the flat floor resources, on which much of the testing for the thesis was accomplished.

Thank you to Professor David W. Miller for his support, advice and direction throughout the S.M program. Thanks to the entire SPHERES team for a great team environment, and helping to pull off really cool experiments on the ISS. Thanks to the SSL girls for creating a social environment that made it fun to come into work over nights and weekends. Finally, a special thanks to my family who have been a constant source of motivation and encouragement over the years.

Contents

List of Figures

List of Tables

Nomenclature

Mathematical Variables

x, y, z	Position
v_x, v_y, v_z	Velocity
Δt	Time increment
$\Theta_x, \Theta_y, \Theta_z$	Euler Angles (Attitude)
$\omega_x, \omega_y, \omega_z$	Angular rates
F_x, F_y, F_z	Force
T_x, T_y, T_z	Torque
m	Mass
I	Inertia
α	Angular acceleration
a	Linear acceleration
ζ	Damping ratio
ω_n	Bandwidth
τ	Time constant
u	Control input

Acronyms

CM	Center of Mass
DOF	Degrees of Freedom
GPS	Global Positioning System
ISS	International Space Station
L1 / L2	Lagrange point 1 / Lagrange point 2
LEO	Low Earth Orbit
MSFC	Marshall Space Flight Center
SPHERES	Synchronized Position Hold Engage Reorient Experimental Satellites
PADS	Position and Attitude Determination System
PD	Proportional - Derivative
PID	Proportional-Integral-Derivative
RFID	Radio Frequency Identification
SIFFT	Synthetic Imaging Formation Flight Testbed
SWARM	Self-Assembling Wireless Autonomous Reconfigurable Modules
UAV	Unmanned areal vehicle
UDP	Universal Docking Port

Chapter 1

Introduction

The invention of the space telescope in the 1980s was an attempt to overcome the disturbance effects of the atmosphere. The Hubble Space Telescope, the first space telescope, was launched in 1990 into low Earth orbit. As the first telescope in space, it has sent dazzling images in the visible light spectrum for almost 20 years (Figure 1-1)[21]. Nowadays, there are several telescopes that span the spectrum. Some examples are Spitzer Space Telescope (Infrared)[3], Chandra X-Ray Telescope [9], and Solar and Heliospheric Observatory (SOHO) (Ultra-Violet) [22]. These telescopes also exist in different locations and orbits, not just LEO. Current telescope orbits include highly elliptical Earth orbits (Chandra), Earth trailing heliocentric orbits (Spitzer), LEO (Hubble), and Earth-Sun L1 (SOHO). All telescopes have different requirements based on their particular orbit the type of telescope. For example, infrared telescopes are

Figure 1-1: Hubble Images of Cat's Eye Nebulae and Eagle Nebulae [14]

required to have cryogenic cooling systems. However, all space telescopes (all satellites in general) are subjected to the same limitations due to launch restrictions.

1.1 Motivation

Telescopes are the tools to see beyond the front door of our planet and solar system. Telescopes, such as Hubble, Spitzer, and Chandra, have contributed to leaps in the knowledge and understanding of the universe. However, the telescopes of today are not sufficiently capable to answer the important questions. Are we alone in the universe? Are there Earth-like planets? The desire to look deeper into space, farther back in time, is creating an interest in pushing the boundaries of space telescope design. However, the practical issues of launching and operating a telescope in space lead to considerable disadvantages. Space telescopes, similar to any satellite, tend to be developed as point-designs, meant to optimize their particular tasks and missions. Launch costs are so high, and launch opportunities so rare, that it is only natural to try to include as much functionality as possible into each telescope. While components may be re-used from one telescope to the next, there is usually a very high degree of customization, leading to inevitable increases in testing and verification costs, along with unavoidable decreases in reliability. The customization has several drawbacks, such as substantial up-front design cost, lack of flight heritage, lack the cost and learning curve savings of large production runs, and difficulty of on-orbit repair, replacement, and upgrade of components. Also, space telescopes are required to fit within the confines of the launch vehicle payload faring, essentially limiting the maximum diameter of a monolithic mirror to be 5 m or less [31].

Implementing modular design and on-orbit servicing can greatly help to minimize or eliminate these issues. All telescopes, regardless of mission objectives, have a common set of sub-system functions. These include, but are not limited to, communication, attitude control, avionics, and power. System integration is often plagued by sub-systems not meeting their delivery deadline and by difficulty getting access to a particular sub-system when problems are encountered during integrated test. Several

benefits may arise by modularizing the sub-systems, standardizing their interfaces, and minimizing the number of utilities that require physical connection between modules in order to validate functionality. These include easier access to modules during system test, a more complete level of validation that is performed at the module level, and the ability to have spares so that a malfunctioning module can be extracted and replaced. Also, nonrecurring design cost may be reduced through the re-use of modules, designs, and integration processes. When faced with the design of a new telescope, instead of customizing the design at all levels, the design can focus on the number and orientation of existing modules as well as the design of select custom modules and thereby reduce design costs. Module designs and validation processes can be re-used as new technologies become available and older modules become obsolete. Module design can build upon some heritage from other industries, but much of the design must be tailored specifically for the space telescope. The mirrors (science payload) comprise such a large fraction of the physical mass of the spacecraft; hence, it is not sufficient to delegate them simply into a payload module, as is done in a nominal spacecraft module breakdown.

As future NASA missions grow in complexity, the ability to autonomously repair and service spacecraft becomes critical. In low Earth orbit, the modular design of the Hubble Space Telescope has allowed the observatory to first be repaired and then updated as technology advanced. Hubble has demonstrated the usefulness, robustness, and operation of a modular approach to spacecraft design. For planned L2-based observatories such as TPF-C [17], TPF-I [17], Planet Imager [24], MAXIM [7], and Stellar Imager [8], the advantages and usefulness of a modular approach increase dramatically. For example, if one of the TPF-I spacecraft fails, the mission objective will not be accomplished, to directly detect and characterize planets. With a modular design, small servicing spacecraft can be sent to L2 to replace defunct components (i.e., modules). As another example, a spacecraft's reaction control system can be refueled to extend a mission's life. This is currently being demonstrated by DARPA's Orbital Express. Easy access, standard interfaces, and readily available spares form the basis of successful servicing. The benefit of on-orbit servicing is dependent upon the

modular design of space telescopes. This work explores implementing reconfiguration (the change in system properties to enable a desired functionality) for the purpose of enabling on-orbit servicing and assembly of space telescopes. Implementation of reconfiguration allows for detailed study of control performance during servicing and assembly scenarios. Control performance is then used to assess the impact of modularity on overall system performance.

1.2 Background

Current work has studied the impact of modularity in general. There have been studies on the benefits of modular architectures, both from a technical perspective and an economical perspective. Though there has been work done on identifying how to modularize a generic satellite [25], little work has been attempted on modularizing a space telescope. Hubble was a modular satellite, but it was a point modular design. This means the design of Hubble can not easily be used to extrapolate as a baseline to modularize other space telescopes [15]. Also, the control impact of reconfiguration in space systems has not been studied extensively. There has been some work on identifying the impact of reconfiguration for aircraft, of recovery time when an engine fails [10]. The lack of analysis on reconfiguration is indicative of the minimal implementation that has occurred in space. Though there have been many dockings of the Progress spacecraft to the ISS, the mass of the Progress spacecraft (7000 kg) versus the mass of the International Space Station (213,800 kg) is so small (3%) that the Progress spacecraft has very little influence on the dynamics of the ISS. The Space Shuttle (99,318 kg), however, is approximately a 50% mass increase to the ISS [1]. Reconfiguration does occur when the Shuttle docks to the ISS. It exists in the form of updating the notch-filter controllers for the new dynamics.

Three aspects are considered in this paper: on-orbit servicing, assembly, and reconfiguration for formation maneuvers. The first successful on-orbit servicing was demonstrated by DARPA's Orbital Express Vehicle. The ASTRO servicing spacecraft (700 kg) successfully transferred 31.97 lbm of hydrazine to the NextSat satellite (224

kg). The relative mass difference between the ASTRO and NextSat (32%) make this a significant control achievement [16]. The application of this technology to space telescopes would greatly help to increase their capability in space. Other work has been done on progressing on-orbit servicing technology. Studies have been done to establish methods for analyzing the cost effectiveness of servicing under uncertaininty [26] [32]. Preliminary designs for servicing vehicles have been published. Some are plug-and-play systems [25], while others use Smart Velcro [20]. Other work has been done on the cost and logistics of servicing [19] and on the flexibility enabled by servicing [30] [18].

On-orbit assembly technology is very similar to on-orbit servicing technology. Preliminary architectures have been developed, most emphasizing a modular architecture. Concepts for future missions include on-orbit assembly as a enabling technology. However, to date, the only on-orbit assembly that has occurred has been manned, instead of automated.

Much work has been done for the formation reconfiguration. Methods for determining the optimal formation reconfiguration have been determined [33]. Also, there has been work done on trajectory planning to account for constraints such as obstacle avoidance. Though minimal hardware demonstration has occurred, the theory has been well developed. Other work is being conducted on developing autonomy algorithms, especially in a real-time environment. Real-time calculation of control and path planning has been demonstrated on a set of UAVs at MIT [5].

A lot of work has been done addressing various aspects of servicing and/or assembly, such as path planning, docking, or modularity. However, not much work has been done to fit these pieces together. Reconfiguration enables seamless transition of maneuvers during the mission by incorporating these various aspects. It is important to develop a method for implementing reconfiguration, such that it effectively incorporates the key system properties.

1.3 Reconfiguration Methodology

Reconfiguration is defined in this work as the **method of changing system properties to enable a specific functionality**. Reconfiguration can be applied to a broad range of systems and functions. In this study, the specific system that is considered is the space telescope. Research on reconfiguration in this thesis falls under two categories: implementation and application.

1.3.1 Implementation

The implementation category refers to the implementation of reconfiguration on a satellite. In this work, reconfiguration was implemented on the SPHERES satellite testbed. The implementation accounts for the changes in the physical properties of the SPHERES when they dock with other systems (such as the mass, inertia, and center of mass). The objective of implementation is to determine the properties of the system after it undergoes a change from one configuration to another (ex. docking to a payload) and be able to effectively control to the same level of accuracy as before reconfiguration. This work consists of modifications to the control algorithms on SPHERES, as well as developing a method for determining and calculating configurations.

Definitions

There are a few terms that are mentioned throughout the text that have specific connotations within this thesis.

- *Elements* are smallest modules of the systems. *Element* functionality is encapsulated, such that only the interfaces are specified. *Elements* cannot be broken down further. For example, a SPHERES satellite and a docking port are each *elements*, but a the combination of a SPHERES satellite on an air carriage is not an *element*.

- *Configuration* refers to the particular way *elements* are arranged in the system.

The example of a SPHERES satellite on an air carriage is a *configuration*. The *elements* in the system are the SPHERES satellite and the air carriage, and they are connected by the -Z face of the satellite being attached to the +Z face of the air carriage.

- *Hard* interfaces are specified as those that are connected by a physical means: mechanical, fluid, electrical, etc. The implementation of reconfiguration on SPHERES assumes hard interfaces, primarily through dockings.

- *Soft* interfaces are connections that are easily breakable, such as communication, control, and sensing. Soft interfaces are more prevalent in interferometer or sparse aperture telescopes. *Soft* interfaces are of primary interest in the formation geometry reconfiguration application.

1.3.2 Application

The implementation of reconfiguration is demonstrated on four different applications. These applications relate to the the three aspects of on-orbit servicing, assembly, and formation geometry reconfiguration. The four applications are grouped in two categories: hardware and simulation. Hardware applications use the implemented reconfiguration on SPHERES to demonstrate a capability. The simulation applications develop models for analyzing the systems impact of reconfiguration on overall mission objectives. Systems impacts include modularity level, time to assemble, mass, and fuel usage.

1.4 Research Objectives

The overall objective of this work is to determine a method for implementing reconfiguration on SPHERES and assess the performance on various applications. The primary objectives are given below.

- Implement mass property reconfiguration on SPHERES satellites

- Identify how the method of obtaining mass property information affects operations
- Identify control impact of reconfiguration on SPHERES satellite
- Define a baseline modular telescope for a space telescope
- Assess impact of modularity on servicing and assembly missions
- Hardware demonstration of formation geometry reconfiguration

1.5 Outline

The thesis focuses on three subjects.

- Chapter 2 briefly describes the SPHERES testbed. It discusses the specific components that were used in implementing reconfiguration, such as the mixer and thruster geometry.
- Chapter 3 describes reconfiguration implementation. This chapter steps through the method of implementing reconfiguration on SPHERES, as well as test results from simulation and ISS that assess the control impact of reconfiguration.
- Chapter 4 describes the applications on which the implementation of reconfiguration was tested. This section details four applications, consisting of two hardware demonstrations and two simulation studies:
  - Demonstration of assembly (SWARM)
  - Demonstration of formation reconfiguration (SIFFT)
  - Impact of mass property update on operations
  - Impact of modularity on robotic servicing
- Chapter 5 provides a conclusion, final analysis, and future work section.

Chapter 2

SPHERES Testbed Overview

2.1 Testbed Overview

SPHERES (Synchronized Position Hold, Engage, Reorient Experimental Satellites) is a formation flight testbed designed to provide a fault-tolerant environment for the development and maturation of control and estimation algorithms for formation flight, docking, autonomy, and reconfiguration. The SPHERES testbed consists of two parts: a ground testbed at MIT and a flight testbed operated by astronauts on the International Space Station (ISS). The flight testbed utilizes the unique microgravity environment of the ISS and creates a laboratory to develop and validate algorithms. SPHERES consists of six self-contained, identical, free-flyer satellites: three on the ground at MIT and three satellites on the ISS (Figure 2-1). Ground satellites are used

Figure 2-1: SPHERES satellite

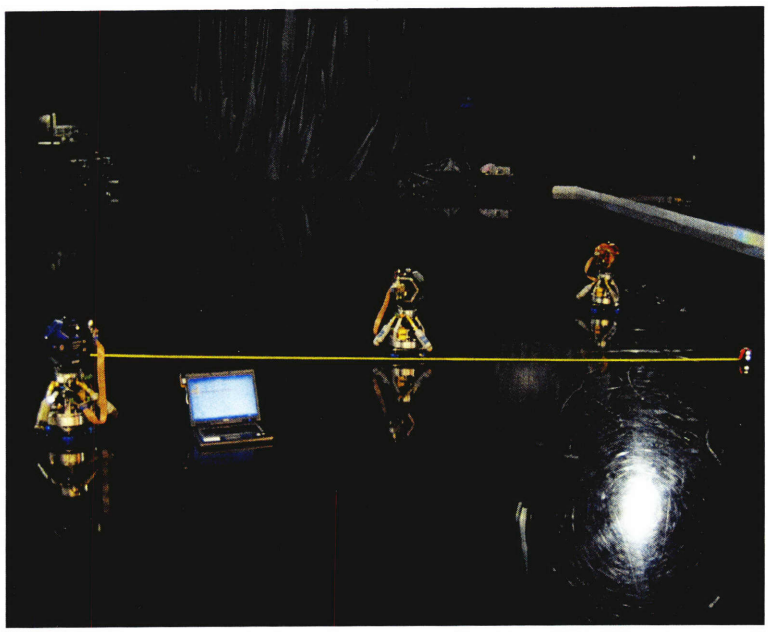


Figure 2-2: SPHERES satellites at MSFC Flight Robotics Laboratory. Max separation of satellites in test reaches 3m, shown by the tape measure.

to test algorithms prior to uplink to the ISS. Each SPHERES satellite is a complete spacecraft bus, equipped with sensing, propulsion, computing, and communication capability. Additionally, each satellite has an expansion port, which is used on the ground satellites to augment the functionality of the satellites by attaching external payloads.

The components of the SPHERES testbed are the satellites, a laptop computer that serves as a ground station, and five small beacons that form the Position and Attitude Determination System (PADS). The five beacons create the working area and provide the global reference frame. SPHERES has three main operational environments: simulation, MIT-SSL facility, and the ISS. Ground testing also occurs at NASA's Marshall Space Flight Center Flight Robotics Laboratory (Figure 2-2), especially for testing that requires more space than is provided by the MIT-SSL facility [29] [6]. The following sections describe the aspects of the SPHERES testbed that were directly used in this work.

2.2 Hardware

Detailed information about the SPHERES satellites can be found in References [11], [28], and [4]. Only elements specifically used in the work will be called out. The key elements for this work are communication, sensors, expansion port, and propulsion.

2.2.1 Communication

SPHERES uses two communication channels. The SPHERES-to-SPHERES (STS) channel is used for inter-satellite communication, enabling cooperative and coordinated maneuvering between satellites during tests. The SPHERES-to-Laptop (STL) channel is used to transmit data and telemetry to the laptop station. Each channel is on a independent radio frequency. More detail can be found in Reference [28].

2.2.2 Sensors

The SPHERES position and attitude determination system (PADS) consists of inertial sensors and ultrasound beacons and receivers. Inertial sensors include three gyroscopes and three accelerometers, providing three-axis inertial measurements. The ultrasound system consists of 24 ultrasound receivers and one beacon on each satellite. There are five external wall-mountable beacons. Estimation is based on sequenced time-of-flight measurements from the beacons to the receivers to determine a state. PADS provides real-time position, velocity, attitude, and angular rate information to each SPHERES satellite up to five Hz (Figure 2-3). More detail can be found in Reference [11].

2.2.3 Expansion Port

The SPHERES expansion port provides the interface to augment the current functionality of the SPHERES. The expansion port is a 100 pin connector that enables the expansion port payload to interface with the satellite's main processor. Thus, the payload is able to acquire and transmit information back and forth through the

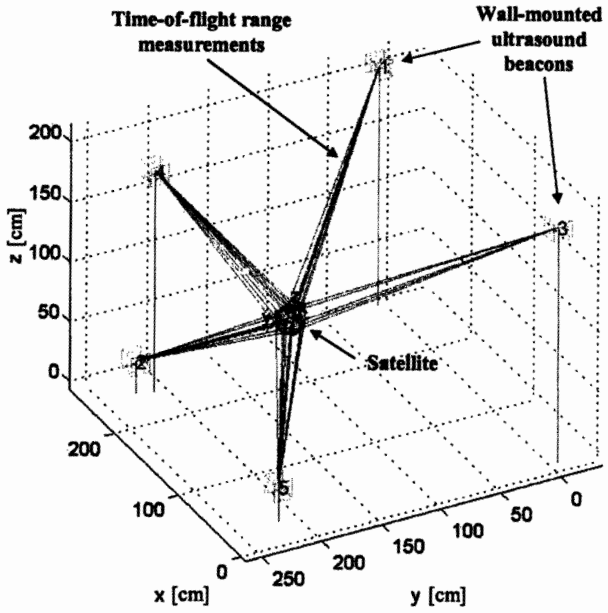


Figure 2-3: Geometry of PADS. Beacon locations define operational volume. Time of flight ranging used from at least four beacons to get full state determination.

interface. This interface is used extensively on the ground with a number of payloads that have been designed: tether mechanism, optical pointing payload, and universal docking port. The particular payload that is used heavily in this work is the Universal Docking Port (UDP) (Figure 2-4). The UDP is a genderless docking port with embedded beacons and receivers. Each UDP has a metrology ring, with three beacons and three receivers. These sensors are used in addition to the SPHERES sensors in a estimator that provides the state of the satellite relative to other UDPs.

The mechanical design of the UDP is a pin-hole mechanism. The pin enters the hole and trips a photosensor. The photosensor initiates the two counter-rotating cams that close to grip the pin. The UDP also has an electromagnet to aid in smooth docking and undocking. More detail can be found in Reference [27].

2.2.4 Propulsion

Each satellite has the ability to maneuver in six degrees of freedom (DOF) propelled by a cold-gas thruster system fueled by CO_2. The satellite has a single propellant tank

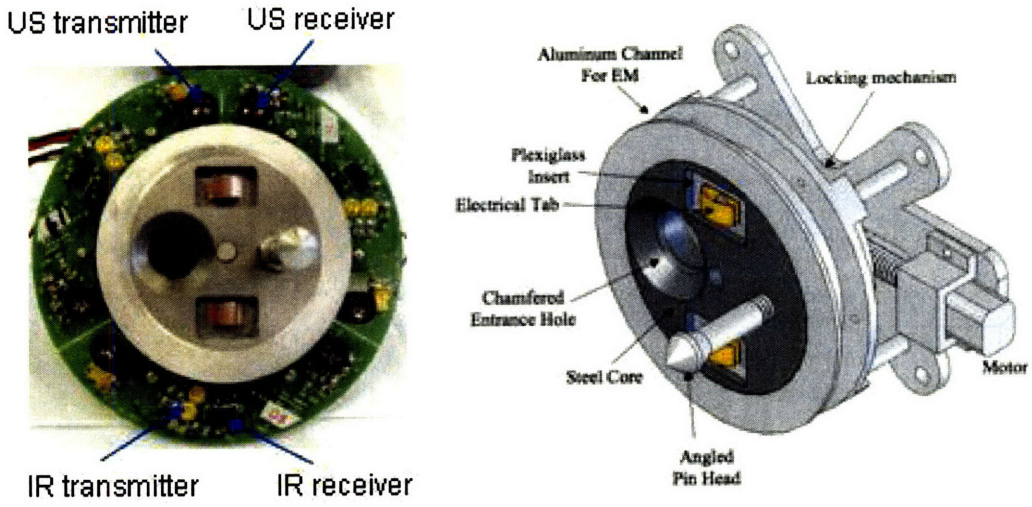


Figure 2-4: Universal Docking Port. Left: Picture of hardware, with sensors labeled. Right: CAD drawing of UDP

with 840g of liquid CO_2 at 860 psi. Twelve thrusters are symmetrically positioned around the satellite to provide control about all three axes, enabling simultaneous attitude and translation control. The thruster geometry is specified by the direction of the force and torque produced by that thruster(see Figure 2-5). For a single satellite, the geometry is given by Table 2.1.

For multiple SPHERES satellites attached to each other, the geometry is determined through which the faces the satellites are connected. Tables 2.2 and 2.3 give the thruster geometry for two satellites connected by their +X (expansion port) and -X (velcro), respectively.

2.3 Software

SPHERES software consists of a set of core functionalities (*SPHERESCore*), and additional user-selectable library functions. *SPHERESCore* is responsible for handling interrupts and interfacing with the hardware. The library utilities provides guest scientists with the ability to use pre-defined utilities, to expedite testing. By having already available controllers, estimators, etc., users are able to develop only that which they want to test, using the existing algorithms to fill in the blanks [6].

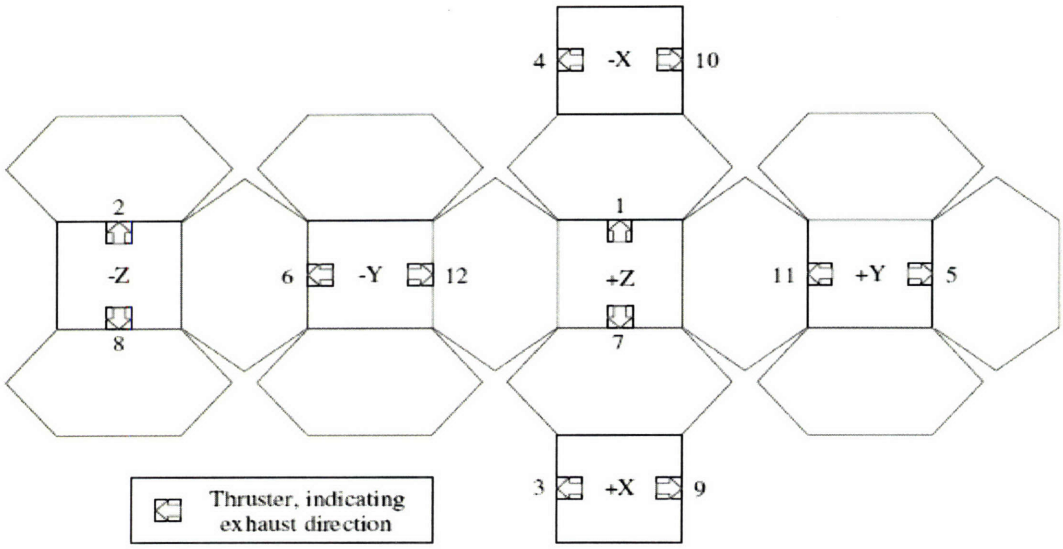


Figure 2-5: Two-dimensional exploded view of thrusters one the SPHERES satellites [11]

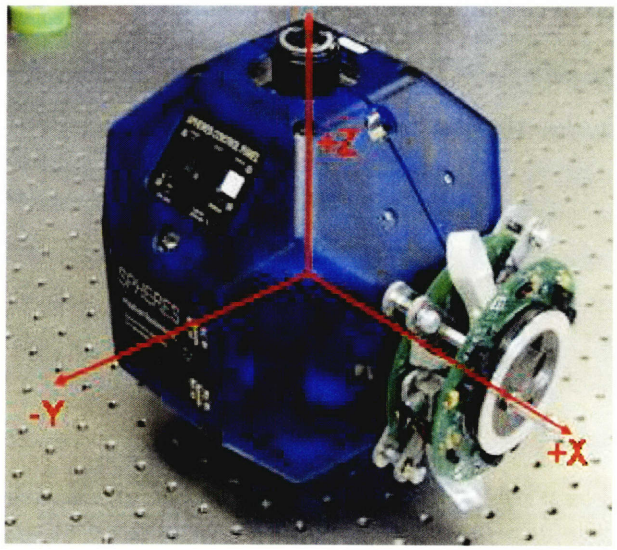


Figure 2-6: SPHERES Body Frame Axes

Table 2.1: Mapping of the SPHERES thrusters to Forces/Torques on the satellite

Thruster Number	Force			Torque		
	x	y	z	x	y	z
1	+1				+1	
2	+1				-1	
3		+1				+1
4		+1				-1
5			+1	+1		
6			+1	-1		
7	-1				-1	
8	-1				+1	
9		-1				-1
10		-1				+1
11			-1	-1		
12			-1	+1		

2.3.1 Physical Properties

The satellite physical properties are handled by a single file. This file is part of *SPHERECore* and is transparent to the rest of the code. It loads the satellite mass and thruster properties: mass, inertia, center of mass, thruster strength, thruster direction, and thruster location. The values stored in Flash memory (specific to each satellite) are automatically loaded when the satellite is reset. The satellite-specific flash values are maintained in the file *spheres-physical-properties.c*. Users can also set other values, such as the values from the CAD model. The mass and thruster properties are set into working variables, local to this file. These parameters can be accessed by 'get' and 'set' functions. Examples of parameters that can be accessed are dry mass, total mass, fuel mass, inertia, center of mass (dry and wet), thruster direction and location.

2.3.2 Controllers

Several simple controllers are already available through the *SPHERESCore* library. This work made use of the following controllers:

- PD Attitude Controller

Table 2.2: Thruster directions of two SPHERES satellite (satellite A and B) attached via their +X faces, as Forces/Torques matrix, in Satellite A's frame

Thruster	**Force**			**Torque**		
Number	x	y	z	x	y	z
1/1A	+1				+1	
2/2A	+1				-1	
3/3A		+1				-1
4/4A		+1				-1
5/5A			+1	+1		
6/6A			+1	-1		
7/7A	-1				-1	
8/8A	-1				+1	
9/9A		-1				+1
10/10A		-1				+1
11/11A			-1	-1		
12/12A			-1	+1		
13/1B	+1				+1	
14/2B	+1				-1	
15/3B		+1				-1
16/4B		+1				-1
17/5B			+1	+1		
18/6B			+1	-1		
19/7B	-1				-1	
20/8B	-1				+1	
21/9B		-1				+1
22/10B		-1				+1
23/11B			-1	-1		
24/12B			-1	+1		

Table 2.3: Thruster directions of two SPHERES satellite (satellite A and B) attached via their -X faces, as Forces/Torques matrix, in Satellite A's frame

Thruster	**Force**			**Torque**		
Number	x	y	z	x	y	z
1/1A	+1				+1	
2/2A	+1				-1	
3/3A		+1				+1
4/4A		+1				+1
5/5A			+1	+1		
6/6A			+1	-1		
7/7A	-1				-1	
8/8A	-1				+1	
9/9A		-1				-1
10/10A		-1				-1
11/11A			-1	-1		
12/12A			-1	+1		
13/1B	+1				+1	
14/2B	+1				-1	
15/3B		+1				+1
16/4B		+1				+1
17/5B			+1	+1		
18/6B			+1	-1		
19/7B	-1				-1	
20/8B	-1				+1	
21/9B		-1				-1
22/10B		-1				-1
23/11B			-1	-1		
24/12B			-1	+1		

Table 2.4: Attitude controller gains for different configurations

Configuration	Mass (kg)	Inertia (kg/m^2)	PD Gains		PID Gains		
Satellite	4.3	0.0225	0.0036	0.0135	0.00450	0.0001800	0.01434
Satellite plus Battery Pack	4.494	0.0331	0.0053	0.01986	0.00662	0.0002648	0.02110
Satellite + Satellite	8.60	0.0450	0.00720	0.0270	0.009	0.00036	0.2869
Satellite + Single Puck Air Carriage	12.43	0.067	0.01072	0.04020	0.01340	0.0005360	0.04271
Satellite plus Battery Pack + Single Puck Air Carriage	12.624	0.0776	0.01242	0.04656	0.01552	0.0006208	0.04947
Satellite + Satellite + Single Puck Air Carriages	24.86	0.134	0.02144	0.08040	0.02680	0.0010720	0.08543

- PD Position Controller
- PID Attitude Controller
- PID Position Controller

The proportional-derivative (PD) controller is based on the following control law (Equation 2.1).

$$u = K_{p_x}x + K_{p_y}y + K_{p_z}z + K_{d_x}v_x + K_{d_y}v_y + K_{d_z}v_z \tag{2.1}$$

where x, y, and z are state errors (e.g., attitude) and v_x, v_y, and v_z are the derivative errors (i.e.,angular rate). The control input (generally called *ctrlControl* in *SPHERESCore*) is given by u.

The Proportional-Integral-Derivative (PID) controller is based on control law given by Equation 2.2.

$$u = K_{p_x}x+K_{p_y}y+K_{p_z}z+K_{d_x}v_x+K_{d_y}v_y+K_{d_z}v_z+K_{i_x}x\Delta t+K_{i_y}y\Delta t+K_{i_z}z\Delta t \tag{2.2}$$

K_p, K_d, and K_i represent the different gains for each axis. Gains for these controllers have been pre-calculated for nominal SPHERES configurations. Nominal configurations consist of SPHERES, standard single puck or three puck air carriages, and docking ports. Gains have been calculated for the configurations in Table 2.4. The bandwidth (ω_n) for attitude control used was 0.4 rad/s and the damping ratio (ζ) used was one. For Integral gains, the time constant (τ) was set as 20s (Table 2.4). Equations for the gains are presented in Section 3.1.1.

2.3.3 Estimators

Two estimators were used: relative estimation and gyro-only estimation. The relative estimator is based on the beacons and receivers of the docking port. The relative estimator is an Extended Kalman Filter (EKF) that calculates the relative state between the two docking faces. The components of the thirteen element state vector are relative position (x, y, z), velocity (v_x, v_y, v_z), attitude (represented by four quaternions), and angular rate $(\omega_x, \omega_y, \omega_z)$. The EKF allows for the propagation of non-linear states (i.e, attitude quaternions). A pre-filter is used to screen the measurements prior to input into the estimator to improve performance. Several heuristics are used for the pre-filter. One example of a heuristic is to reject a measurement from a face if it is significantly greater than the remaining measurements on that face. Another example is to reject a measurement if it leads to a distance that is greater than 7m since the beacon cannot be that far away, therefore it is most likely the result of multi-path. The estimator is initialized with an initial state. The state is propagated to the next time step using the dynamics model. The state estimate is then updated based on sensor measurements. The cycle is then repeated for the next time step. The gyro-only estimator works in a similar manner, but uses only gyroscope measurements to estimate the attitude and angular rates of the satellite [11] [12].

2.3.4 Mixers

The mixer that is currently used on SPHERES was developed based on a single satellite configuration and converts control input (u in Equation 2.1 and 2.2) to thruster on/off times. First, the control input is converted into the body frame of the SPHERES (Figure 2-6). The thruster geometry is hard coded as a matrix that converts thruster commands (which thrusters are firing) to the control input, called the Mixing matrix. In order to convert the control input to firing times, the control input is multiplied by the inverse of the Mixing matrix to get the thruster commands. The thruster commands are then converted to firing on/off times based on duty cycle, minimum pulse length, and thruster strength [11].

2.4 Summary

The SPHERES testbed was the baseline for this study. Much of the work was either directly implemented on SPHERES, or the SPHERES models were used in simulation. The content of this chapter provided an overview of the main aspects of the SPHERES testbed as they relate to this work. To get a detailed understanding of the aspects mentioned or how they interact, please consult the references specified. Further sections in this thesis will reference hardware and software algorithms introduced in this section.

Chapter 3

Methodology and Implementation of Reconfiguration on SPHERES

This chapter will utilize the scenario of a space tug docking to a passive payload to illustrate the aspects of implementing reconfiguration. In order to accomplish this task, the tug must reconfigure its control algorithm, as the system's properties change through docking, in order to maintain control performance throughout the mission phases. The system is defined as the tug element and the payload element. The scenario of the space tug has some inherent assumptions. First, the tug is a propellant-based tug, which performs assembly through docking and undocking maneuvers. Second, the tug is a rigid body system, modeled with no disturbances. The tug is modeled after a SPHERES satellite, with six degree-of-freedom actuation and full state observability. Finally, the tug is responsible for maintaining authority over all other payloads.

The implementation of reconfiguration in this thesis assumes centralized control. In general, one satellite updates the configuration, calculates desired control input, and calculate firing times for all actuators in the system. In a situation where the payload is passive (no actuation or sensing capability) and there is only one active satellite (tug), this distinction is unnecessary. However, when the payload is another satellite, it is important to distinguish that only the tug keeps track of the configuration and enables control. The remaining satellites only actuate their thrusters based

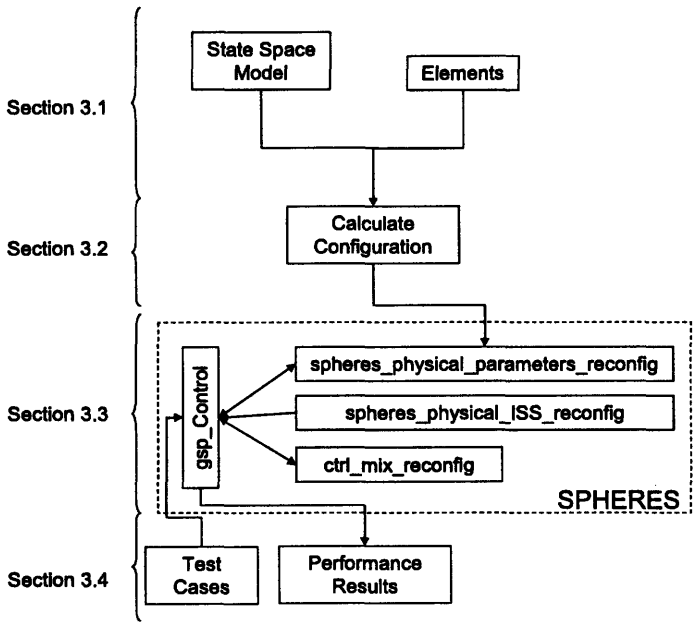


Figure 3-1: Chapter 3 Roadmap

on firing times received from the tug. Hence, the tug is designated as the 'Master.' This is explained in more detail in Section 3.1.4. This chapter will step through the methodology represented in Figure 3-1 using the space tug scenario.

3.1 Defining a Configuration

The first aspect of implementing reconfiguration is to establish a methodology for defining a *configuration.* Specification of a *configuration* should give enough information that the relative position and orientation of every element is known. The following steps were used to formulate a method for defining a configuration.

1. Identify the aspects of the system that will change due to the addition or removal of a payload.

2. Determine how these changes impact the control system.

3. Develop a method for characterizing the configuration of the system, in terms of what attachments are currently on, where others could be attached, etc.

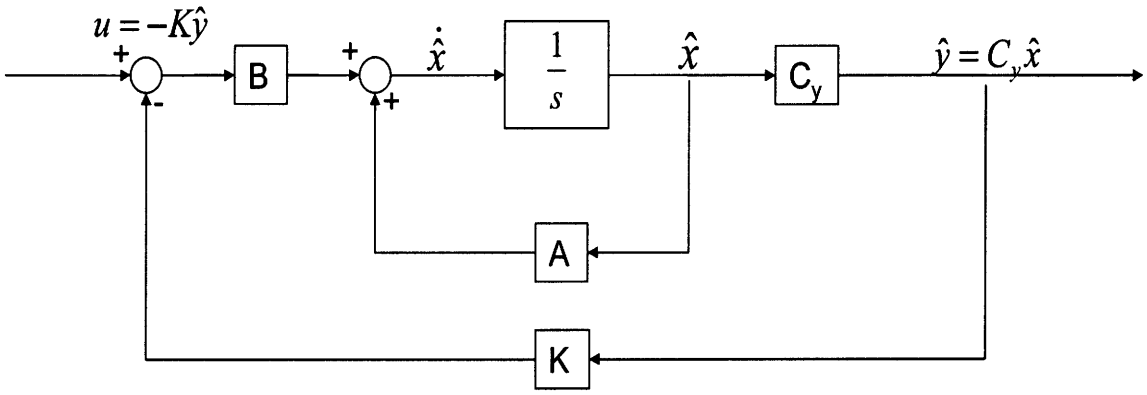


Figure 3-2: Representative continuous time system block diagram

The first two steps are completed by creating a representative state-space model for the tug. The purpose of the state space model is to identify what properties change during docking and how they affect the performance of the tug. The state space model is used to determine which properties should be tracked as the *configuration* changes. The third step is accomplished by building a database of elements and developing a framework for how to represent their inter-connections to form a *configuration.*

3.1.1 State Space Representation

The state space representation allows us to obtain a high-level view of the system, and determine what components are most affected by a change in configuration. A generic state space model is created to provide an linear approximation to an actual system. Since state space modeling is for linear systems, and most tugs (e.g. SPHERES) are non-linear systems, the state space model is only used as a tool to qualitatively identify parameters that change as the configuration changes. Figure 3-2 gives a representation of the system in block diagram form to highlight the different aspects that need to be updated.

The idealized system is assumed to have the following characteristics:

- Discrete Time Linear system model (shown as continuous time in Figure 3-2)
- Rigid Body dynamics, double integrator system
- Fully controllable (6 DOF propellant based control)

- Fully observable (all states directly observable)
- No disturbances
- No feed forward term (i.e., D matrix = 0)

The state space representation (Equation 3.1) allows us to clearly see what aspects of the entire system need to be updated in order for the modified configuration to function to the level that the previous configuration did. This analysis considers the following matrices: A, B, C, and K matrices. The state vector used in the analysis is the full state of the spacecraft: position ($x \quad y \quad z$), velocity ($v_x \quad v_y \quad v_z$), attitude (given by Euler angles)($\Theta_x \quad \Theta_y \quad \Theta_z$), and angular rate ($\omega_x \quad \omega_y \quad \omega_z$).

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \\ u &= -Ky \end{aligned} \tag{3.1}$$

A matrix

The A matrix is the state transition matrix: it describes how the state changes with no external inputs in the inertial frame. Since this model is a discrete time system, the equations of motion are based on $F = ma$. The following equations, which identify the structure of the A matrix, are obtained by assuming no control input,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{t+\Delta t} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_t + \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_t \Delta t, \quad \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}_{t+\Delta t} = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}_t + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_t \Delta t$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_{t+\Delta t} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_t, \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_{t+\Delta t} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_t \tag{3.2}$$

where Δt is the time increment of propagation, dependent on the control rate. The rotation angles given in Equation 3.2 are converted to quaternion form, as a sequence

of rotations about the body axes of the satellite. The quaternion propagation is a non-linear propagation that uses the angular rates (Ref. [11]). By combining the above equations, the following A matrix shown in Equation 3.3 is obtained. The structure of the A matrix (the propagation of the state) is not dependent on the mass properties of the vehicle. This matrix would have to be updated if the system changes such that it is no longer a double integrator system. Flexible systems, where the state matrix would need to include that flexibility and vibrations in the model (i.e. spring-mass system), would not be modeled as double integrator systems. Also, if the control rate changes, Δt changes, would require an update of the A matrix.

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

B matrix

The B matrix is the control input matrix: it describes the impact of the control input on the vehicle state. In the propellant-based system, the control input, u, is provided by the thrusters and is given by $\begin{bmatrix} F_x & F_y & F_z & T_x & T_y & T_z \end{bmatrix}$, where F is the force and T is the torque on the tug in three dimensions. These inputs are related to the states by,

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_{t+1} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_t + \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} \left(\frac{1}{m}\right) \Delta t$$
$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_t = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} I^{-1} \Delta t \tag{3.4}$$

Unlike the A matrix, the B matrix contains parameters that can change when reconfiguration occurs. The effect of the control input is based on Newton's Laws,

$$\begin{aligned} F &= ma \\ T &= I\alpha \end{aligned} \tag{3.5}$$

where a is linear acceleration, m is the vehicle mass, F is the force imparted, T is torque imparted, I is the vehicle inertia, and α is the angular accelerations. Equation 3.2 is updated to include the effect of the control inputs. Thus, the B matrix is given by the following matrix.

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\Delta t}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\Delta t}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\Delta t}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_x^{-1}\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & I_y^{-1}\Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & I_z^{-1}\Delta t \end{bmatrix} \tag{3.6}$$

Though the control input does not change, the firing times calculated from the

control input changes based on thruster geometry. As mentioned in Section 2.3.4, the mixer converts the desired control input into thruster on/off times. In order to do this, the mixer needs to know the location of a thruster so that it can determine how much torque is produced by opening that particular thruster. For thruster force, the orientation of a thruster relative to the body frame prior to docking may be different that its orientation to the new body frame defined after docking. Thus, the location of the thrusters with respect to the center of mass (CM) is also a vital parameter to be updated. In order to calculate the thruster position, we must first calculate the new CM with respect to a fixed point on the satellite. (See Section 3.1.4).

C matrix

The C matrix is the sensor output matrix; it relates the states (x) to the sensor outputs (y). Estimators relates the sensor outputs (y) and noise inputs (w,u) to the state estimates ($\hat{x}$). The number, location, and orientation of the sensors change the size and parameters of the C matrix. This is analogous to how the thruster configuration affects the B matrix. A SPHERES satellite, without a UDP, has twenty-four ultrasound sensors, three gyroscopes, and three accelerometers for a total of thirty individual sensors. Thus, for the 12x1 state vector, the C matrix would be 30x12. The entries of C relate each of the states to the output sensors. For example, the angular rates are given directly by the gyroscopes. Therefore, to accurately update the C matrix, the sensor locations and orientations for all thirty sensors must be updated with respect to the new CM.

K matrix

The K matrix contains the controller gains used to stabilize the system. These gains are calculated as a function of bandwidth (ω_n), damping coefficient (ζ), inertia (I), mass (m), and time constant (τ). Of these parameters, the inertia and mass are the ones that change after the addition of a payload. The equations used to calculate the gains for PD attitude control are given by

$$K_p = \omega_n^2 I_{zz}; K_d = 2\zeta\omega_n I_{zz}.$$

For PID attitude control, the equations used are

$$K_p = I_{zz}\left(\omega_n + \frac{2\zeta\omega_n}{\tau}\right); K_i = I_{zz}\left(\frac{\omega_n^2}{\tau}\right); K_d = I_{zz}\left(2\zeta\omega_n + \frac{1}{\tau}\right)$$

The PD position gains are given by

$$K_p = \omega_n^2 m; K_d = 2\zeta\omega_n m$$

PID position gains are given by

$$K_p = m\left(\omega_n + \frac{2\zeta\omega_n}{\tau}\right); K_i = m\left(\frac{\omega_n^2}{\tau}\right); K_d = m\left(2\zeta\omega_n + \frac{1}{\tau}\right).$$

These gains calculations assume principal axes inertias, and are shown for the Z-axis.

3.1.2 Implementation of State Space Representation

As described in previous sections, the state space representation enabled the identification of parameters that would cause control performance to change after reconfiguration. To summarize, these parameters are mass, inertia, center of mass, thruster locations and orientations, and sensor locations and orientations. Therefore, these are the minimum set of parameters that must be tracked for each element in the the system. The distinct value of these parameters specifies a configuration. In later sections, these parameters will be enumerated for different elements and configurations.

3.1.3 Elements

The elements are defined as the smallest non-divisible components of the system. As mentioned previously, there are certain key parameters that need to be tracked. The specification of a configuration is dependent on knowing the parameters of all of the elements in the configuration. In addition to knowing the parameters of each

of the elements, it is also important to know how they are positioned relative to other elements in the configuration. To determine relative placement, one element is assigned to be the reference point, designated as the 'Master.'

Certain characteristics of the 'Master' are assumed: it is the central controller of the system; it has computational capability; it has full actuation and sensing capability. The assumption of a central controller is made because, as discussed previously in this section, there is assumed to be only one centralized controller. This controller is responsible for sensing and estimating the state of the system, calculating necessary control inputs, and relaying the firing times to the all applicable elements. It greatly facilitates the task to make the reference and the controller the same element. Having two separate elements for the two tasks would require additional computation to convert between the reference frame and the body frame of the controller. Given that the reference and the controller are the same element (designation 'Master'), it is important that the Master have computational capability (to calculate the new geometry and new configuration, as well as inertia properties) and to have sensing and actuation capability (to estimate and control about the trajectory error). The Master is generally the tug, as in the scenario; thus, it is assumed that elements are either connected to the tug or free floating in the vicinity.

Thus, the location of all remaining elements are referenced from the Master. If elements are free floating, they are identified in terms of their position with respect to the Master. If elements are attached to the Master, they are referenced by the docking interface by which they are connected. Thus, by knowing the element, as well as the docking interface to which it is attached , the configuration can be calculated. For this work, the Master is always assumed to be a SPHERES satellite. Possible payloads are given by Table 3.1. Note that all of the payloads listed are passive; they do not have actuation or computation capability. The inertia matrix is about the CM of the payload and the x-axis is through the primary interface. The CM is located with respect to the primary interface location.

Table 3.1: Mass Properties of Payloads

Payload	Mass (kg)	Inertia ($kgm^2 * 10^{-7}$)	CM ($m * 10^{-2}$)	Span ($m * 10^{-2}$)
UDP	0.376	$\begin{bmatrix} 2720.0 & -87.4 & 0 \\ -87.4 & 1620.0 & 0 \\ 2010.0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 2.48 & 0.318 & 0.00165 \end{bmatrix}$	$\begin{bmatrix} 6.97 & 9.68 & 7.62 \end{bmatrix}$
Air Carriage	7.57	$\begin{bmatrix} 330403.55 & 52171.52 & 6257.38 \\ 52171.52 & 428170.30 & -211.62 \\ 6257.38 & -211.62 & 455219.66 \end{bmatrix}$	$\begin{bmatrix} 0.27 & -0.01 & -11.13 \end{bmatrix}$	$\begin{bmatrix} 31.735 & 17.145 & 17.627 \end{bmatrix}$
Counterweight	0.558	$\begin{bmatrix} 3751.23 & 0 & 0 \\ 0 & 2175.71 & 0 \\ 0 & 0 & 2175.71 \end{bmatrix}$	$\begin{bmatrix} -1.27 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 2.54 & 6.35 & 6.35 \end{bmatrix}$
Battery	0.238	$\begin{bmatrix} 1451.8 & 0 & 0 \\ 0 & 1832.9 & 0 \\ 0 & 0 & 1786.3 \end{bmatrix}$	$\begin{bmatrix} 1.848 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 3.696 & 5.952 & 6.146 \end{bmatrix}$

3.1.4 Multiple Masters

Thus far, in the scenario chosen (space tug docking to a payload element), the only possible Master is the tug itself. This is because all payload elements are assumed to be passive (no computation, sensing, or actuation capability). However, now consider a scenario where the tug docks to a non-passive element (i.e. active element). If this element also fulfills all of the requirements to be a Master, how is a Master chosen?

First, the distinction between scenarios with only one possible Master (designated *Sub-assemblies*) and scenarios with multiple possible Masters (designated *Assemblies*) is made. Sub-assembly configurations are calculated by systematically adding one element at a time and calculating the geometry and inertia parameters at each intermediate step. Assembly configurations are calculated the same way, except they are the systematic addition of sub-assemblies, not elements. For an Assembly, one active element is arbitrarily chosen as the Master.

However, in the case of the Assembly reconfiguration, the number of sensors and actuators changes. This changes the size, as well as the content, of the B and C matrices. Communication is now required between the active elements. Sensor measurements must be radioed from all active elements to the Master, and the thruster commands must be radioed from the Master to the active elements. Figure 3-3 shows the flow of calculation for updating a configuration with multiple Masters. First, the Sub-Assemblies are calculated, using the parameter information of the elements. Next, the Sub-Assembly configurations are sent to the Master of the Assembly. The Assembly Master then calculates the Assembly configuration. The final Assembly con-

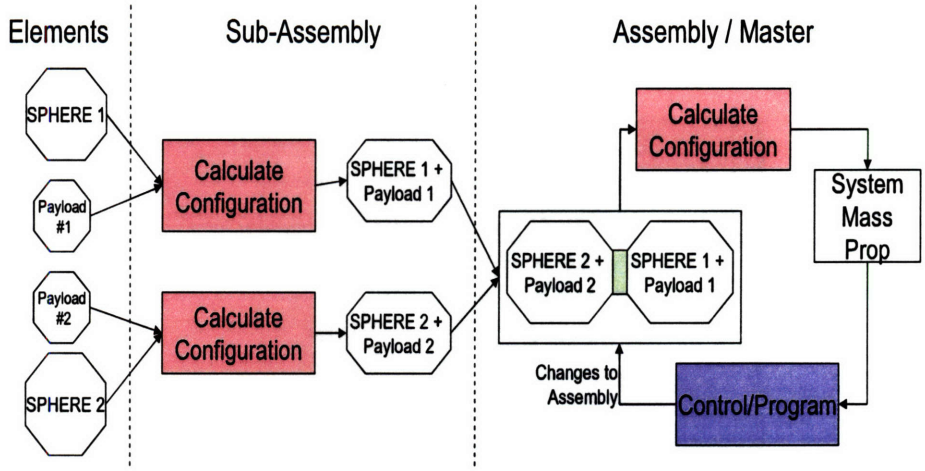


Figure 3-3: Flow diagram for update of the configuration for system with multiple potential Masters

figuration is the input for the control system. If the Assembly configuration changes during execution, the configuration is re-calculated by the Assembly Master based on the new relative positions of the Sub-Assemblies. The selection of the Master is arbitrary and currently pre-selected by the human user.

3.2 Calculating a Configuration

From Section 3.1, the parameters that need to be directly updated are: mass, inertia about the CM, center of mass, thruster locations with respect to the CM, and sensor locations with respect to the CM. To this list, the dimensions of the element are added (designated as *span*) to simplify the calculation the CM. Calculating these parameters for a configuration involves the sequential additional of individual elements until reaching the final configuration. The following section steps through the calculation of the baseline configuration: single SPHERES satellite with a docking port attached to its expansion port (+X face, SPHERES body frame)(Figure 3-4). This configuration is representative of the space tug in the example scenario. When mounted on an air carriage, this configuration is used often in ground testing (see Sections 4.2 and 4.3).

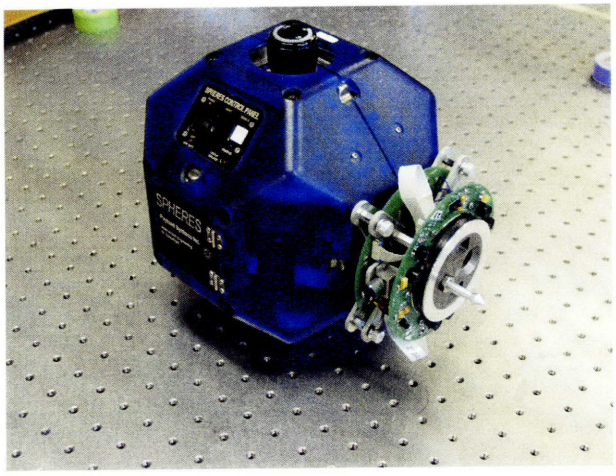


Figure 3-4: Photograph of a SPHERES satellites with a UDP attached

Table 3.2: MassProp structure parameters

Parameter	**Description**
Mass	in kg
Inertia	3x3 tensor about CM
Center of Mass	3x1 array, in m, given from Master's primary interface port
Span	3x1 array of dimensions, in m
Thruster locations	array of size equal to number of thrusters locations given about the CM in m, only included if applicable, else array is empty
Thruster directions	array of size equal to number of thrusters numbers (1,2,3) represent X, Y, and Z axes respectively sign of entry specifies direction (+ or -)
Thruster valid	array of size equal to number of thrusters non-zero entry specifies if thruster is okay to use for actuation
Sensor locations	array of size equal to number of sensors, locations given about the CM in m, only included if applicable, else array is empty

Table 3.3: Inputs and Output matrix for Configuration Calculation code

Function	Description	Input	Output
propertiesSPHERE	Initialize SPHERE element parameters	N/A	MassProp structure
propertiesDockingPort	Initialize UDP element parameters	N/A	MassProp structure
propertiesAirCarriage	Initialize Air Carriage element parameters	N/A	MassProp structure
propertiesBatteryPack	Initialize Battery Pack element parameters	N/A	MassProp structure
Node-Config	Calculate sub-assembly parameters	Master element Payload element Interface	MassProp structure
Node-Config	Calculate assembly parameters	Master Sub-Assembly Payload Sub-assembly Interface	MassProp structure

3.2.1 Inputs

The baseline configuration is a Sub-assembly, only the SPHERES satellite fulfills the requirements for being designated the Master. The other elements (namely the UDP) are passive objects; it is assumed that each passive element has only one possible interface. There are two inputs for calculating the configuration: the parameter values of the elements and the interface on the Master where the payload element is connected. The parameter values are consolidated into a single data structure, designated *MassProp structure* (Table 3.2). In addition to the parameter values, the interface input is provided as a string, with the name of the interface face (ex. '+X').

Thus, each element is initialized such that the structure is created (e.g. *propertiesSPHERE function* in Table 3.3). The input to the Calculate Configuration box in Figure 3-3 (function *Node-Config* in Table 3.3) is Master, payload, and interface location. Each successive run of *Node-Config* outputs the *MassProp* structure with the new payload attached. A flag is set in *Node-Config* to check for the existence of thrusters or sensors such that it can be used to calculate configuration for both Sub-assemblies and Assemblies.

3.2.2 Outputs

The output of the code is another *MassProp* structure with the values of the new system. This Sub-assembly configuration is added to the database if it is needed for future use. The Sub-assembly configuration can now be an input to build an Assembly.

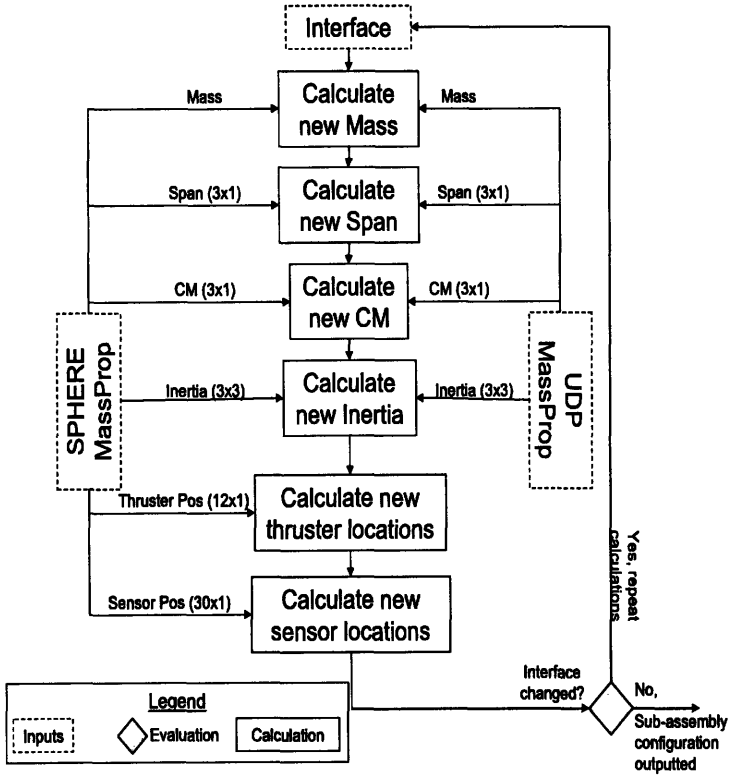


Figure 3-5: Flow diagram of code to calculate configuration

3.2.3 Code Walk-through

For the baseline configuration of a SPHERES satellite with a UDP, the initial configuration starts out with just the single SPHERE. First, the SPHERE *MassProp* structure and the UDP *MassProp* structure are initialized. Then, the Sub-assembly (SPHERE plus UDP) is calculated by calling *Node-Config* with the SPHERE MassProp structure, the UDP *MassProp* structure, and the interface of '+X' as inputs. The steps of calculation in *Node-Config* are depicted in Figure 3-5. The rest of this section steps through each calculation box in Figure 3-5.

Mass, Span, Center of Mass

The calculations for the mass, span, and center of mass are straightforward. The new total mass is simply the sum on the individual masses. The new span is calculated based on the interface specified. The total spans are added in the axis of attachment, and the max span between the two objects is taken for the remaining two directions

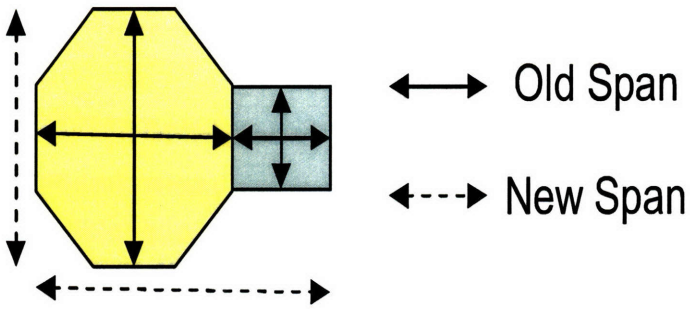


Figure 3-6: Schematic of the update of the span

(Figure 3-6). The spans (i.e., dimensions) for each element are defined in the body frame of that element. Therefore, when adding spans, one must rotate the body frame of the payload element according to the docking location and orientation. Thus, the space axes are all aligned and can then be added.

The calculation of the CM is important because it has a direct impact on the thruster and sensor locations (as given with respect to the CM), and thus subsequent behavior of the system under actuation. The following assumptions were made in the calculation of the CM:

- The location of the CM of the object is given with respect to the primary docking interface
- The CM of the object is given with respect to the unattached edge of the primary interface
- The primary interface is along the X body axis of the element
- The Payload is smaller and less massive than the Master . Thus, the CM will stay inside the Master.

First, the CM of the payload element is calculated with respect to the interface point. By adding the interface location to the CM of the payload element, one can get the location of the payload element's CM in the Master's frame (Figure 3-6). Next, the center of mass is calculated by using Equation 3.7.

$$X_{cm} = \frac{\sum m_i x_i}{\sum m_i} \quad Y_{cm} = \frac{\sum m_i y_i}{\sum m_i} \quad Z_{cm} = \frac{\sum m_i z_i}{\sum m_i} \tag{3.7}$$

where x_i, y_i, and z_i are the CM locations for element i, m_i is the mass of element i, and the the summation is over the number of elements. If the payload element is being attached to the Master's primary interface and has an unattached docking interface, the primary interface is updated to be the unattached docking interface of the payload element. Referring to Figure 3-6, the unattached docking interface would be the right side of the payload (i.e., square box). The location of the CM of the Sub-assembly then is given with respect to this 'new' primary interface in the Master's body frame. This allows for subsequent attachments to the unattached docking interface. For example, when attaching the UDP to the SPHERE's primary interface, although the primary interface is now occupied, the SPHERE+UDP Sub-Assembly still has docking capability on the unattached docking interface of the UDP. Thus, the UDP face now becomes the primary interface of the SPHERE-UDP Sub-assembly. If the payload element does not have an unattached interface, the original primary interface is retained.

Inertia

Four steps are used to calculate the new inertia matrix (depicted in Figure 3-7):

1. The element inertia matrix about the element CM is calculated about the interface point using the parallel axis theorem,

$$I_{ip} = I_{cm} + mr^2$$

   where I_{cm} is the inertia about the CM of the element (3x3 diagonal tensor with principal inertias), I_{ip} is the inertia about the interface point (3x3 diagonal tensor with principal inertias), m is the mass of the element, and r is a 3x3 matrix where the diagonal entries are the distance from the element CM to the interface point.

2. The inertias are rotated to align body frames according to the primary interface

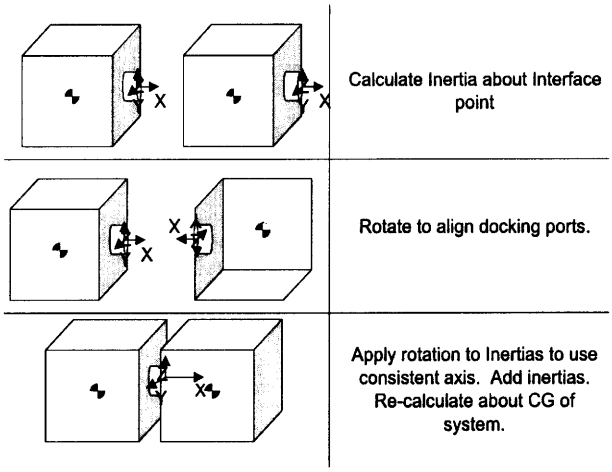


Figure 3-7: Method for updating inertia by shifting Inertias to interface point

and orientation, where the rotation matrix R is given by

$$R = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

3. The inertias are added, $I_{sum} = I_{ip_1} + I_{ip_2}$, where I_{sum} is the inertia of the Sub-Assembly about the interface point.

4. The inertia of the Sub-assembly (I) is calculated about the CM of the Sub-assembly, using the parallel axis theorem again,

$$I = I_{sum} + mr^2$$

where r is a 3x3 matrix with CM locations as the diagonal entries.

Thruster Location, Direction, and Validity

All thruster parameters must be updated to ensure the proper update of the B matrix. The thruster locations are calculated in two steps. The first step is to calculate the thruster positions about the interface point. This involves adding the center of mass of the Master to the locations that are about the CM. The second step is to subtract

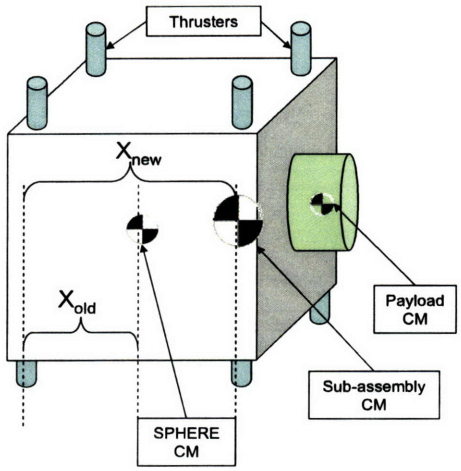


Figure 3-8: Schematic depicting the update of the thruster positions

the new CM from the locations with respect to the interface point, to get the locations about the new CM (see Figure 3-8). For the baseline configuration of SPHERE plus UDP, the thruster direction (Table 2.1) and thruster valid parameters do not change. This is because the UDP is a passive element, smaller in span and less massive than the SPHERE. Thus, the number of thrusters do not change and thruster directions do not change because the CM remains inside the SPHERE. Only the thruster locations with respect to the CM need to be updated.

However, when the payload element has actuation capability or is sufficiently massive to affect the thruster directions, these parameters are calculated offline by inspection. Tables 2.2 and 2.3 give the force/torque matrix for two SPHERES satellites attached along their X body axes (see Section 2.2). From the force-torque matrix, the thruster directions can be derived. The validity of the thrusters is currently used to mitigate plume impingement, though there are other applications (e.g., FDIR). Figure 3-9 shows two SPHERES attached where the inner sensors have been disabled. Analogous to Figure 3-9, inner thrusters would be disabled since their thruster directions cause plume impingement on the other satellite.

Sensor Locations

The sensor locations are updated the same way as the thruster locations. However, this has not been fully implemented or integrated such that a single estimator is able

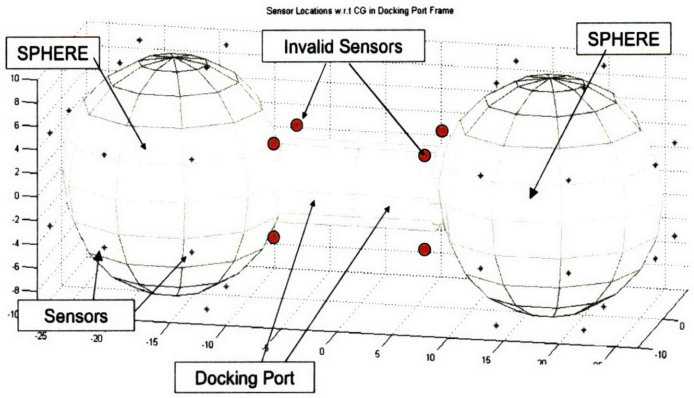


Figure 3-9: Diagram identifying invalid sensors due to configuration

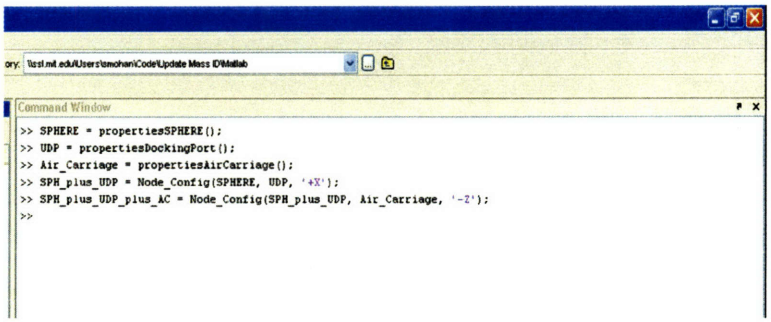


Figure 3-10: Matlab Command line input to calculate mass properties

to use all sensors, or even use the updated locations. Figure 3-9 shows the invalid sensors for a two satellite case. These receivers are blocked by the presence of the other satellite.

3.2.4 Code Execution

In order to obtain the mass properties of a SPHERE with a docking port attached, one would follow the given steps (graphically shown in Figure 3-10).

1. Initialize *MassProp* structure for elements (i.e., SPHERE and UDP)

2. Define assembly sequence. Designate Master.

3. Run *Node-Config* with Master and payload. Save output *MassProp* structure.

4. Repeat previous step until all payloads are attached.

3.3 Implementation on SPHERES

All configurations were calculated off-line and coded as a look-up table in *SPHERESCore*. There were three main files that need to be updated in *SPHERESCore* to fully access the look-up table of configurations and enable reconfiguration: *spheres-physical-properties* (A matrix), *ctrl-mix* (B matrix), and *est-USBeaconGyroUpdatePvarEKF* (C matrix). Of these three files, two were updated, renamed *spheres-physical-properties-reconfig* and *ctrl-mix-reconfig*. The gain update currently is not implemented as a separate function, but calculated off-line and set in the *gspControl* interface (during each maneuver).

3.3.1 Spheres-Physical-Properties-Reconfig

Due to the memory and processing limitations of the SPHERES, the configurations are not calculated online. A minimal set of configurations that are expected to be used during the test are included in array format. The configurations are pre-calculated using the method described in Section 3.2. The data was stored in a structure called *MassProp*. It is similar to Table 3.2 with a few modifications to simplify calculations. Sensor locations are not included because the estimator has not been updated yet to make use of the data. The *MassProp* structure implemented in *spheres-physical-properties-reconfig* is given by Table 3.4. The new parameters are stored as user-selectable configurations, accessed by the variable *Config*. The configurations currently present are given by Table 3.5. The *Config* variable is accessed through functions *sysConfigGet* and *sysConfigSet* in the file *spheres-physical-properties-reconfig*.

3.3.2 Ctrl-mix-reconfig

The SPHERES mixer, a function that translates requested control input into thruster firing durations, is updated to account for the change in thruster locations, due to the attachment of a payload. The basic outline of the reconfigurable mixer is given below (also see Figure 3-11). Only sections that are different from the original *ctrl-mix* are

Table 3.4: MassProp structure parameters

Parameter	Description
Mass	in kg
Span	3x1 array of dimensions, in m
Center of Mass (Wet)	3x1 array, in m, given from Master's primary interface port accounts for fuel mass
Center of Mass (Dry)	3x1 array, in m, given from Master's primary interface port only accounts for dry mass
Inertia	3x3 tensor about CM
Inertia inverse	inverse of 3x3 inertia tensor
Thruster locations	array of size equal to number of thrusters locations given about the CM in m, only included if applicable, else array is empty
Thruster directions	array of size equal to number of thrusters numbers (1,2,3) represent X, Y, and Z axes respectively sign of entry specifies direction (+ or -)
Thruster valid	array of size equal to number of thrusters boolean entries, non-zero entry specifies that thruster is okay to use for actuation

Table 3.5: List of Configurations implemented in SPHERESCore

Config	Name	Description
1	SPHERES Flash	Default parameters that are stored in SPHERES flash memory
2	SPHERES CAD	parameters calculated from SPHERES CAD model in 2003
3	SPHERES+UDP+CB+SAC	SPHERES with a UDP attached on +X face, Counterbalance on -X face, mounted on a single puck air carriage
4	2 SPHERES docked	2 Config 3 sub-assemblies docked through their +X faces (UDP)
5	SWARM Assembly 1	SPHERE mounted on a Node, with 2 UDP, on a three puck air carriage
6	SWARM Assembly 4	Assembly 1 -X UDP docked to +X UDP of another Assembly 1
7	SWARM Assembly 6	Assembly 1 -X UDP docked to +X UDP of Assembly 3 (SPHERE on a Node docked to a Sub-Aperture)
8	SWARM Assembly 8	Assembly 1 -X UDP docked on -X UDP of Assembly 3 and +X UDP docked to +X of Assembly 2

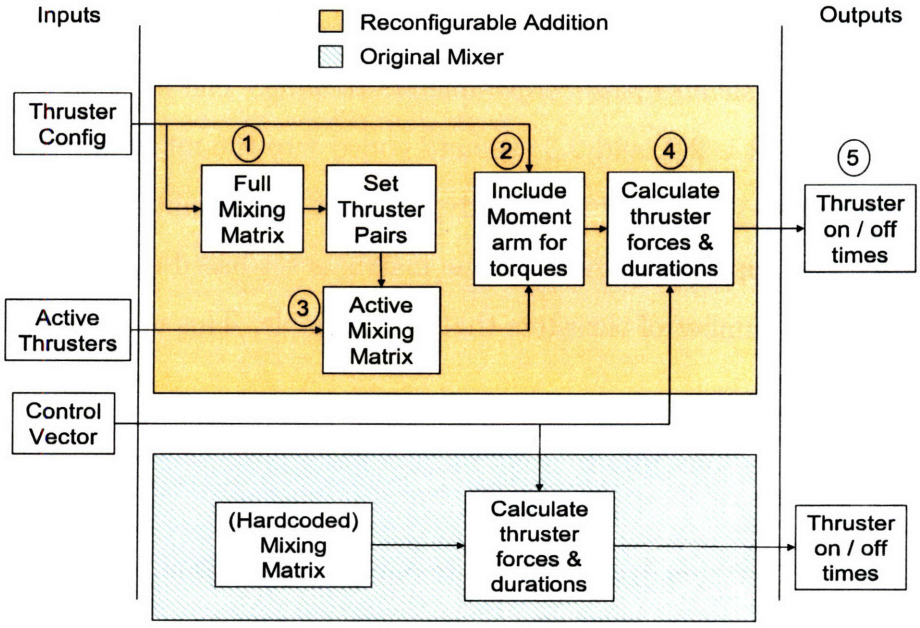


Figure 3-11: Reconfigurable mixer flow diagram

described here. More information about *ctrl-mix* can be found in Section 2.3.4 and in Ref. [11].

1. Load updated Thruster Configuration, then calculate Mixing Matrix. The Mixing Matrix specifies which thrusters contribute to which forces and torques. For example, Thruster 3 produces -X force and +Z torque (Table 2.1).

2. Use thruster locations to incorporate rotation arm, defined as the perpendicular offset of thrust axis from the CM, into the Mixing Matrix.

3. Use *thrusters valid* parameter to condense the Mixing Matrix to only contain valid thrusters.

4. Use Mixing Matrix and desired control input (forces and torques) to determine how much impulse each thruster needs to produce (Ns for force, Nms for torque).

5. Convert thruster impulse to on-off times, given the control period, duty cycle, and thruster strength.

Two cases are considered in *ctrl-mix-reconfig*: 12 thrusters, and 24 thrusters. The input structure of the code remains the same, with the additional interface to the *Config* variable. *Config* specifies the current configuration of the system and is accessed through *spheres-physical-parameters-reconfig*. The force/torque matrices specified by Tables 2.1, 2.2, and 2.3 are hard coded into the mixer. First the control input is changed so that it is in the body frame of the satellite (same as previous mixer). Next, the appropriate force/torque matrix is set based on the configuration. Also, this sets the number of thrusters that are available. This is important in terms of sizing matrices.

Thruster Pairs

The thrusters are arranged in pairs to create pure translations and rotations. In order to determine the thruster pairs, the force/torque matrix is analyzed to determine which sets have opposing forces. The pairs are calculated using a series of iterative

Table 3.6: Thruster Pairs for 24 (2 SPHERES connected through -X) and 12 thrusters, respectively

Pair	Thr	Thr	Force	Torque
1	0	12	+X	+Y
2	1	13	+X	-Y
3	6	18	-X	-Y
4	7	19	-X	+Y
5	2	8	+Y	+Z
6	3	9	+Y	+Z
7	14	20	-Y	-Z
8	15	21	-Y	-Z
9	4	23	+Z	+X
10	5	22	+Z	-X
11	10	17	-Z	-X
12	11	16	-Z	+X

Pair	Thr	Thr	Force	Torque
1	0	6	+X	+Y
2	1	7	+X	-Y
3	2	8	+Y	+Z
4	3	9	+Y	-Z
5	4	10	+Z	+X
6	5	11	-Z	-X

loops, and identifying which thrusters have opposing forces and have not already been assigned. For the case of 24 thrusters (when two SPHERES are docked via -X (velcro) faces), there is a non-unique set of possibilities. In this situation, the first feasible solution is taken.

Active Pairs

Table 3.6 gives the thruster pairs for the different scenarios that are mentioned in this work. After determining the full set of pairs, the set is down selected to a maximum of six active pairs based on the valid thrusters. Six pairs is the minimal set that provides full six DOF control. For example, when two SPHERES are docked (as in Table 3.6 for 24 thrusters), pairs 1 and 2 both provide +X translation, but only one pair is necessary. The selection heuristic used preferentially selects the outer most thrusters first, since they provide more torque. The selection is accomplished by appropriately re-setting the *thruster valid* parameter (e.g., from zero to one). This feature can also be utilized to assist in FDI and plume impingement studies.

Mixing Matrix

The inverse of the mixing matrix converts the control input into firing time durations. The active pairs are used to create the mixing matrix by selecting the column of the force/torque matrix that corresponds to the first thruster in the thruster pair. Only one thruster from the pair is needed since the thrusters in a pair are opposing. Thus, the second thruster is simply the negative of the first thruster. The column corresponding to the first thruster of the pair is included as a row in the mixing matrix. First, the full mixing matrix is created using the columns of the first thruster for each thruster pair. Then, the full mixing matrix is condensed to the 6x6 version by eliminating rows that do not correspond to the active thrusters. Next, the moment arm is added to the torque columns of the mixing matrix. A separate function calculates the moment arm by extracting the location of the thruster with respect to the CM in the direction perpendicular to the thruster direction. For example, for a thruster whose direction is +X, the moment arm r would be the distance from the CM in the YZ plane ($r = \sqrt{y_{cm}^2 + z_{cm}^2}$). This value is then divided into the torque columns of the mixing matrix. Finally, the mixing matrix is inverted. The inverse is used to convert the control input into thruster durations.

3.3.3 Estimator

Currently, online update of sensor locations is not implemented in *SPHERESCore*. However, the interface was added for updating the beacon locations when using the docking port. The UDP relative estimator was already set up to accept measurements from three transmitters and three receivers. Two functions were added to the *spheres-physical-properties-reconfig* file: *sysBeaconRxGet* and *sysBeaconTxGet*. This allows the locations of the receivers and transmitters to be set via the function, instead of hard coded in the estimator.

3.4 Control Impact of Reconfiguration

An important quantity to assess is how much impact reconfiguration has on control performance. To date, most of the docking and servicing performed in space has been with payloads that are significantly smaller than the servicing tug. In this case, since the dynamics of the servicing tug are minimally changed, the control performance is retained. However, how does this performance change when the tug is of comparable or lesser mass than the payload that it is moving around? The following experiments attempt to characterize the impact of reconfiguration by comparing the performance obtained when using the same controller for a test maneuver, before and after reconfiguration. A 3-DOF simulation was created to obtain a theoretical baseline (x ans y translation in the horizontal plane, and Θz rotation about the vertical). Then, experiments were conducted in 6-DOF on board the ISS.

3.4.1 Simulation

A simple Simulink model was created to approximately model the SPHERES satellite. The following cases were considered:

1. SPHERES satellite only

2. SPHERES satellite with a battery used as a proof mass

3. SPHERES satellite with a SPHERES satellite used as a proof mass

Case 1 was used as the baseline. Since this is the nominal state of the satellite, the performance achieved in this configuration is set as the target. Cases 2 and 3 were run with two separate gains: gains from Case 1 and gains specifically calculated for this configuration that would achieve the same closed-loop performance as Case 1. The objective is to quantify the performance difference between using the old gains and the new gains. The model used (see Figure 3-12) is a discrete time 3-DOF state space representation of the system. A step input in angle is applied to the Z rotation angle, corresponding to 180°. Hence, the dynamics are propagated based on $T_z = I_{zz}\alpha$. The

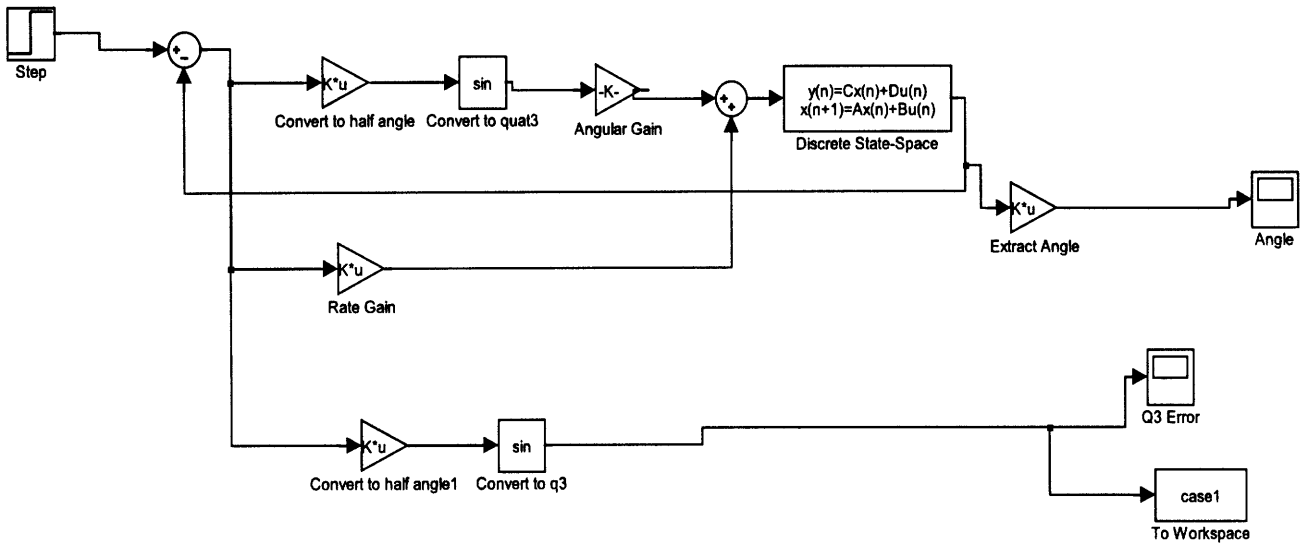


Figure 3-12: Simulink block diagram

Table 3.7: Test set-up and results for simulation cases

Case	Mass (kg)	I_z (kgm^2)	Gains	T_s (s)	T_r (s)
1_{sim}	4.16	0.2140	Baseline	35	25
2_{sim}	4.618	0.2383	Baseline	300	50
2_{sim}	4.618	0.2383	Case 2	35	25
3_{sim}	8.76	0.3498	Baseline	400	50
3_{sim}	8.76	0.3498	Case 3	35	25

matrices for the discrete state space model are given by

$$x(t+\Delta t) = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ \frac{\Delta t}{I_{zz}} \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x(t)$$

The control input is calculated by a PD controller, applying an angular (P gain) and an angular rate gain (D gain). The method for calculating gains is given in Section 3.1. The bandwidth (ω_n) is set at 0.3 rad/s and the damping ratio is set to be critically damped ($\zeta = 1$). The current angle and the angle error is outputted for analysis. The parameters and results for the three cases are given in Table 3.7, where I_z is the principal inertia about the Z-axis, T_s is the settling time, and T_r is the rise time.

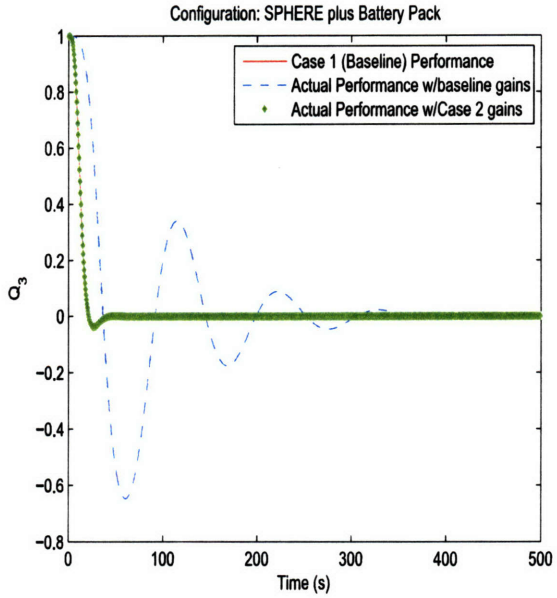


Figure 3-13: Simulated step response of Case 2_{sim}, single SPHERES satellite plus battery proof mass

As seen from Figures 3-13 and 3-14, the performance of the updated gains matches the target performance of Case 1 (baseline). The performance of Cases 2 and 3, when using Case 1_{sim} gains is severely under damped. This is expected since Case 1_{sim} has a smaller mass and inertia than Case 2_{sim} or Case 3_{sim}. Note that this performance difference is for the ideal system modeled; effects such as thruster saturation and delay between control calculation and commanding are not modeled. To determine performance under these realistic effects, the experiment was conducted on hardware, specifically on the SPHERES ISS testbed in 6-DOF.

3.4.2 Hardware

To validate the simplistic Simulink models, equivalent cases were run on the SPHERES testbed on board the ISS. Three test cases were run and compared against the baseline (SPHERE only) configuration.

1. SPHERE + battery proof mass: Two 90° Z-axis rotations. First rotation using SPHERE only (Case 1_{sim}) gains. Second rotation using updated gains for actual configuration. Equivalent to Case 2_{sim}.

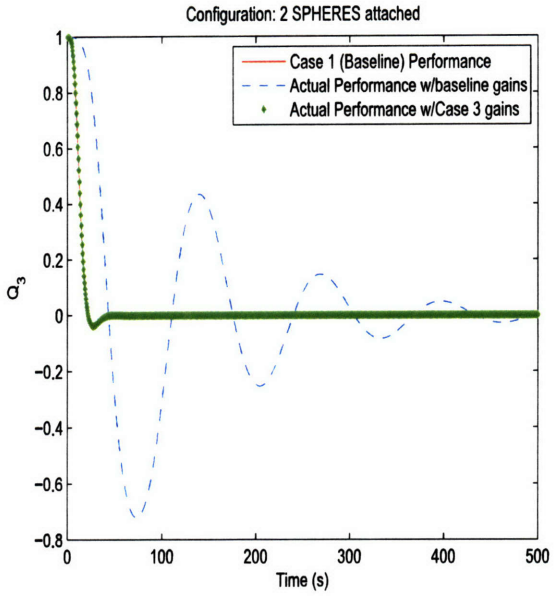


Figure 3-14: Simulated step response of Case 3_{sim}, SPHERES satellite plus satellite proof mass

2. SPHERE + SPHERE proof mass: Two 90° Z-axis rotations. First rotation using SPHERE only (Case 1_{sim}) gains. Second rotation using updated gains for actual configuration. Does not include thruster configuration update.

3. Two active SPHERES attached: Two 180 deg Z-axis rotations using thrusters on both satellites. Only using gains for updated configuration. Equivalent to Case 3_{sim}.

Results from the ISS tests are given in Table 3.8, where speed of response is defined as

$$speedofresponse = \frac{x_{max} - x_{min}}{t_{max} - t_{min}},$$

P_o is the percent overshoot, and payload mass fraction is the percent increase of mass from the baseline SPHERE only configuration. Data was taken during ISS Test Sessions six and eight. The speed of response was used as the distinguishing metric for Case 1_{iss} and Case 2_{iss}. For Case 1_{iss}, the rise time and settling time (not shown) were essentially the same between baseline and Case 1_{iss} gains. For Case 2_{iss}, the test was not run long enough for the system to settle, thus the speed of response was used. Results for attitude control performance of the baseline SPHERE only (Case

Table 3.8: Test set-up and results for hardware (ISS) cases

Case	Mass (kg)	I_z (kgm^2)	Gains	Speed of Response (1/s)	P_o	Payload mass fraction
1_{iss}	4.618	0.2383	Baseline	0.0675	0	11%
1_{iss}	4.618	0.2383	Case 1	0.0676	0	11%
2_{iss}	8.76	0.3498	Baseline	0.04	45°	110.6%
2_{iss}	8.76	0.3498	Case 2	0.059	45°	110.6%
3_{iss}	8.76	0.3498	Baseline	$T_s = 40s$	30°	110.6%
3_{iss}	8.76	0.3498	Case 3	$T_s = 20s$	20°	110.6%

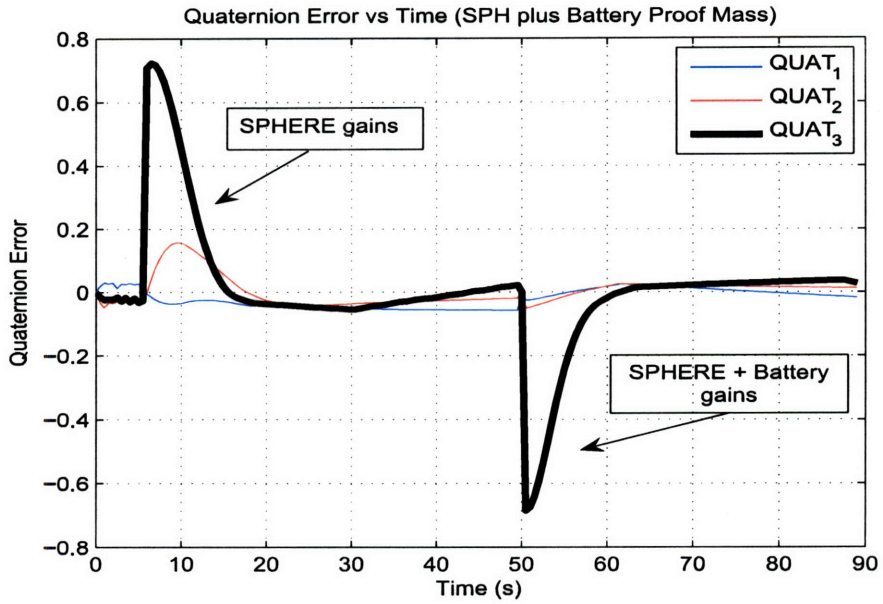


Figure 3-15: Quaternion error for Case 1_{iss}, two step input rotations in ISS with battery proof mass (TS006 data)

1) configuration can be found in Ref [23].

Figure 3-15 shows the results from the Case 1_{iss} test. $QUAT_3$ is the Z-axis quaternion, which was given two step inputs of 90° rotation. $QUAT_1$ and $QUAT_2$ are the X and Y axes respectively, and are commanded to hold attitude at zero degrees. The first step response is for the SPHERE only baseline gains, whereas the second step response is with the updated gains for the Case 1_{iss} SPHERE plus proof mass configuration. The speed of response is slightly faster Case 1_{iss} than the baseline, though not significantly. This is reasonable considering the percent mass increase between the Case 2_{iss} and the baseline is just 11%. Figure 3-16 shows the performance of the Case 2_{iss} test of a SPHERES plus a SPHERE proof mass. The first step response (using SPHERE only baseline gains) is highly under damped. The response completes al-

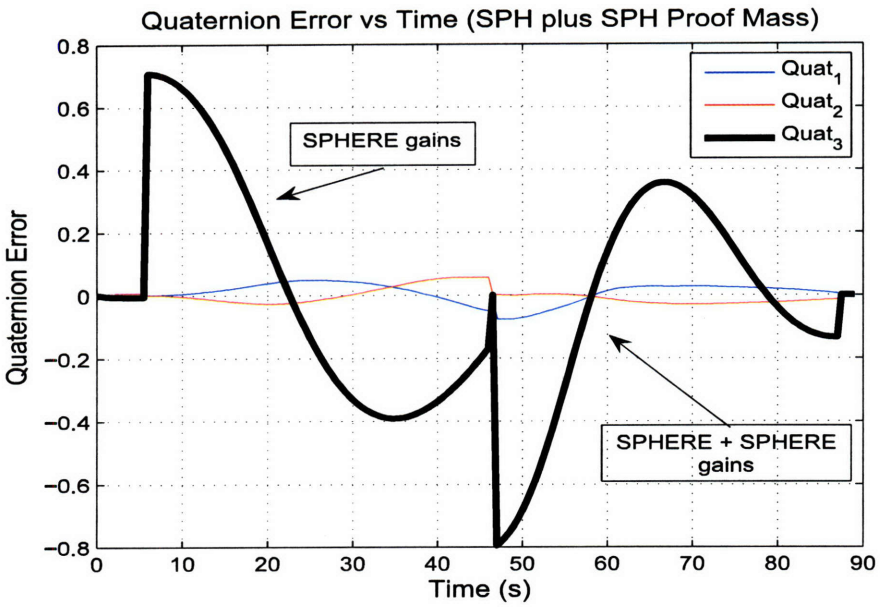


Figure 3-16: Quaternion error for Case 2_{iss}, two step input rotations in ISS with SPHERE proof mass (TS006 data)

most one full oscillations, during the time interval of the step response. In the second step response, the performance is improved. The peak overshoot decreases from an error overshoot of 0.4 to 0.37. Also, the second step response begins its second oscillation during the given time interval, as is seen by the slope of the line at t=86s. The mass increase for Case 2_{iss} is 110.6 percent. Note that the thruster configuration is not updated. Thus, the SPHERE has only half of the actuation capability about the Z-axis. Since only thrusters on one SPHERE are used, only one of the two thrusters for the pair causing +Z torque actually provides Z torque. The second thruster is located close to the interface of the two SPHERES and is essentially pushing at the CM of the system. This performance demonstrates the effects of thruster saturation, since the desired control input could not be fully actuated at a given time step. Instead, the maximum control input possible within the capability of the SPHERE was actuated. This is what causes the under damped system in the second step response, instead of matching the baseline SPHERE only performance. Figure 3-17 shows the performance of Case 3_{iss}, two SPHERES satellites attached using thrusters on both satellites for both rotations. As is expected the performance is much closer to the baseline test than Case 2_{iss}. Attitude control was demonstrated for joint thruster

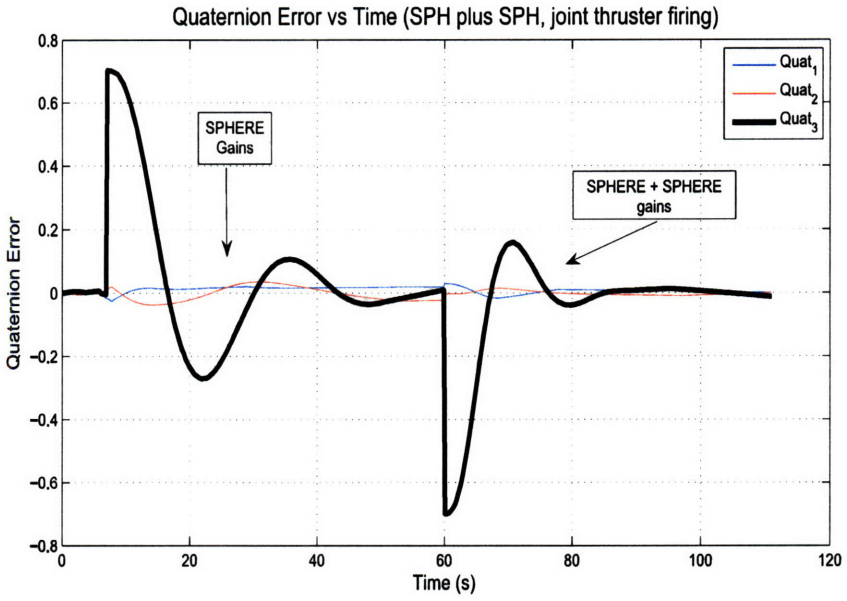


Figure 3-17: Quaternion error for Case 3_{iss}, two step input rotations in ISS with SPHERE proof mass and joint thruster firing (TS008 data)

firing to within 5°. There was a 50% decreased in settling time and a 33% decrease in peak overshoot for Case 3_{iss}. Figure 3-18 shows the performance of Case 3_{iss} configuration under step inputs of multiple axes at same time. Comparison of the three tests show that the update of gains has a more significant impact in Case 2_{iss} and 3_{iss} than in Case 1_{iss}. The greater percent difference between the SPHERE only baseline case and the actual configuration plays an important part of determining how important the update of the controller gains is in a given scenario.

3.5 Summary

The implementation of reconfiguration was demonstrated to be successful for the following key reasons.

- When using the gains derived for the true configuration, performance for test cases 2_{sim},3_{sim}, 1_{iss}, 2_{iss}, and 3_{iss} matches the baseline test case 1_{sim}.
- Experimental results from the ISS matches the performance predicted by the simulation.

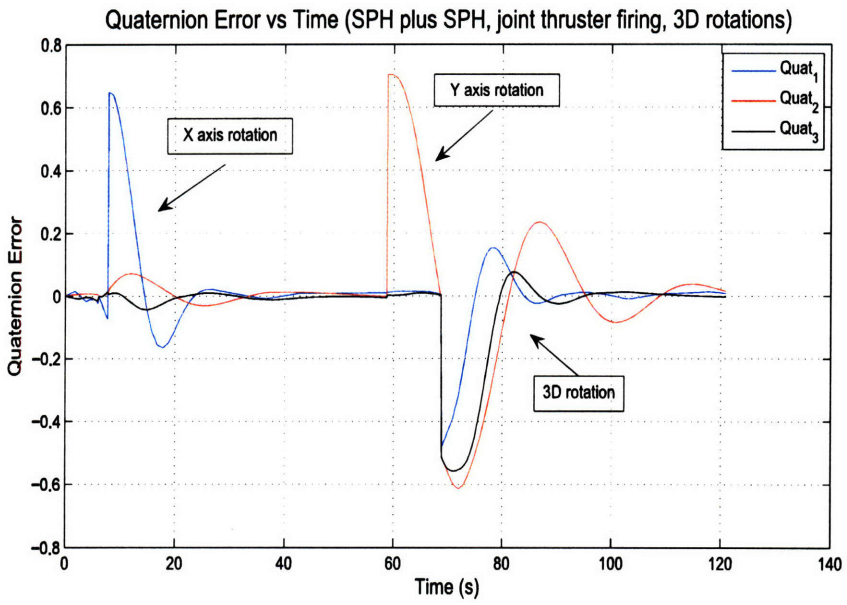


Figure 3-18: Quaternion error for Case 3_{iss} configuration under 3D step inputs (TS008 data)

- The importance of realistic effects, such as thruster saturation, is demonstrated through the performance of test case 2_{iss}, which does not achieve baseline performance.

- Reconfiguring the model and re-deriving the control gains becomes important when the payload mass fraction is large.

Through these results, the implementation of reconfiguration on SPHERES is verified to be accurate. This implementation will be used for hardware demonstrations in Sections 4.2 and 4.3.

Chapter 4

Applications of Reconfiguration in Hardware and Simulation

4.1 Overview

This chapter explores the applications of reconfiguration to mission scenarios mentioned in Chapter 1, namely on-orbit servicing, assembly, and operations. Each of these systems are related in that their configuration changes significantly over the course of their lifetime. For example, servicing missions are adding or removing components from the payload, to fix or replace broken modules. Assembly missions are constantly changing configuration of the tug (docking and undocking from elements) to move pieces around to build the structure. Formation reconfiguration involves moving formation flown elements around with respect to each other.

This chapter considers how reconfiguration affects performance on a system level. This work considers the following scenarios:

1. multi-satellite servicing sequence,
2. assembly sequence,
3. formation change for formation flown satellites

In Scenario 1 and Scenario 2, the servicing/assembly sequences cannot be accom-

plished without successful update of physical parameters (Chapter 3). On the systems level, the issue of concern is not the technological feasibility of reconfiguration, but the impact of the reconfiguration on the performance of the system. Scenario 3 is a control issue in obtaining the accuracy necessary, while accounting for constraints such as safety and obstacle avoidance, fuel-balancing, and optimal path planning. Four applications were considered that address these three scenarios through hardware demonstration or simulation.

1. Hardware demonstration of automated assembly sequence. (Section 4.2)

2. Hardware demonstration of formation geometry reconfiguration. (Section 4.3)

3. Simulation to assess the impact of the type of mass property update method to use for assembly. (Section 4.4)

4. Simulation to assess impact of modularity for servicing and assembly missions for space telescopes. (Section 4.5)

4.2 SWARM (Application #1)

On-orbit servicing and assembly is a critical enabling technology for use on large scale structures in space. The goal of the SWARM (Self-Assembling, Wireless, Autonomous, Reconfigurable Modules) project is to develop and mature algorithms for autonomous docking and reconfiguration, to be used as the building blocks for autonomous servicing and assembly. The SWARM testbed (Figure 4.2) consists of several elements: SPHERES satellites, Universal Docking Ports, Air Carriages, Nodes, and Sub-apertures. The first three elements are described in detail in Chapter 2. The SWARM Node is a baseplate that mounts to the Air Carriage, with a universal adapter (gold frames in Figure 4.2). Hence, it provides generic floating capability and was used to float SPHERES satellites and Sub-apertures. Each Node has the ability to hold up to four docking ports. The docking ports are mounted on adjustable posts, to give coarse alignment.

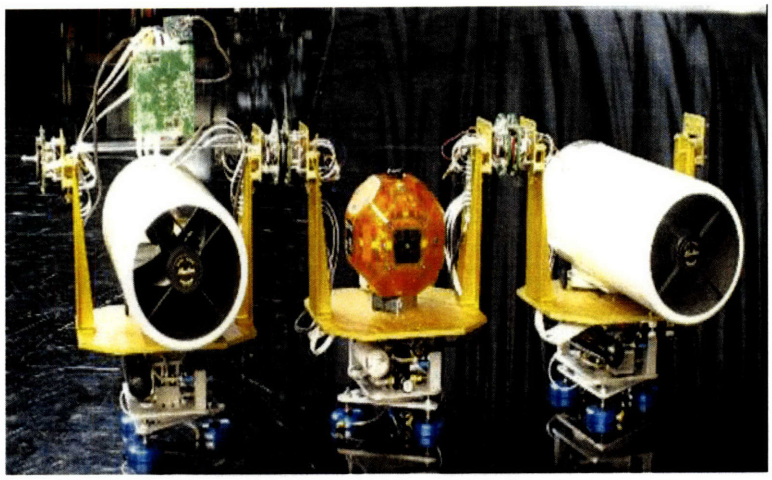


Figure 4-1: SWARM testbed modules at MSFC test

The Sub-apertures were used as representative payloads for servicing and/or assembly. In particular, the goal was to use a SPHERE, mounted on a SWARM Node and Air Carriage, to dock with two sub-apertures. Success in achieving this goal would determine the basic steps in assembling a sparse aperture optical telescope. The goal was accomplished through a hardware demonstration at the MSFC Flight Robotics Laboratory in July, 2006. The objectives of the tests are enumerated below.

1. Verify integration of SWARM hardware. This involves integrating UDPs with new avionics boards capable of commanding multiple UDPs simultaneously, as well as identifying the mass properties for all SWARM elements.

2. Demonstrate relative estimation using SWARM navigation hardware. The relative estimator used the metrology rings on the UDPs to calculate the relative states between UDPs.

3. Demonstrate autonomous docking using relative estimation. The relative estimator was used to provide states which the controller used to dock two elements together. Docking was demonstrated through a phased sequence that included alignment, approach, berthing, and capture.

4. Demonstrate control after docking through reconfiguration. Implement reconfiguration such that the configuration can be updated after docking, and demonstrate control after reconfiguration.

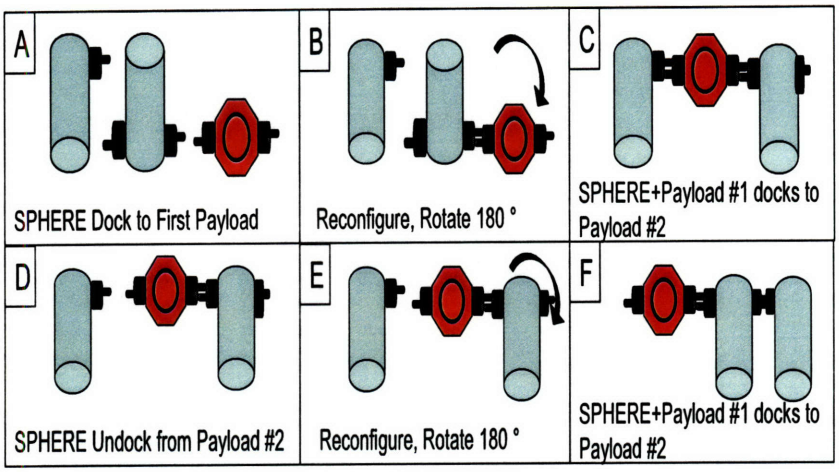


Figure 4-2: SWARM Modules at MSFC test

This work focuses on Objective 4. Detailed discussion about the first three objectives can be found in the References [12] and [13]. Additionally, discussion of the reconfiguration implementation can be found in Chapter 3. This section focuses on the assembly sequence and the results of the reconfiguration at NASA MSFC.

4.2.1 Assembly Sequence

There were three assembly scenarios:

- Minimum: Dock tug to sub-aperture. Performed controlled rotation.
- Baseline: Dock tug to sub-aperture. Rotate 180°. Dock to second sub-aperture.
- Goal: Perform 'Desired' assembly. After successful assembly, undock from one sub-aperture. Rotate and dock unoccupied docking port on the sub-aperture connected to the tug to the free floating sub-aperture.

The assembly sequences are shown in Figure 4-2. The Minimum (A-B) and Baseline (A-B-C) assembly sequences are subsets of the Goal (A through F) assembly sequence. Thus, the sequences can be broken down into sub-tests, each adding successively more maneuvers. The maneuver sequence for the Goal sequence, with notes as to the ending maneuver of the other two sequences, is given in Table 4.1. All maneuvers as given are for the SPHERES tug, since it is the only element with actuation capability.

Table 4.1: SWARM maneuver Goal assembly sequence. Letters in parentheses correspond to Figure 4-2

Maneuver	Description	Assembly Scenarios
Kalman Filter Initialization	Initialize estimator	Minimum, Baseline, Goal
Attitude Control	Null rates and point to Payload 1	Minimum, Baseline, Goal
Transverse alignment	Align transverse to the approach direction, while maintaining attitude	Minimum, Baseline, Goal
Approach Payload 1	Approach Payload 1 to a set berthing location (A)	Minimum, Baseline, Goal
Berthing	Control attitude and position to within docking tolerances (A)	Minimum, Baseline, Goal
Capture	Open loop firing to enter docking port (A)	Minimum, Baseline, Goal
Reconfigure	Update configuration. Sets new parameters (B)	Minimum, Baseline, Goal
Rotate 180°	Rotation to align other docking port to Payload 2 (B)	Baseline, Goal
Attitude Control	Point to Payload 2 (first docking port of 2nd sup-aperture)	Baseline, Goal
Transverse alignment	Align transverse to the approach direction, while maintaining attitude	Baseline, Goal
Approach Payload 2	Approach Payload 2 to a set berthing location (C)	Baseline, Goal
Berthing	Control attitude and position to within docking tolerances (C)	Baseline, Goal
Capture	Open loop firing to enter docking port (C)	Baseline, Goal
Undock	Open docking mechanism and open loop push back (D)	Goal
Reconfigure	Update configuration. Sets new parameters (E)	Goal
Rotate 180°	Rotate to face free UDP on Payload 1 towards Payload 2 (E)	Goal
Attitude Control	Point to Payload 2 (2nd docking port of 2nd sup-aperture)	Goal
Transverse alignment	Align transverse to the approach direction, while maintaining attitude	Goal
Approach Payload 2	Approach Payload 2 to a set berthing location (F)	Goal
Berthing	Control attitude and position to within docking tolerances (F)	Goal
Capture	Open loop firing to enter docking port (F)	Goal

4.2.2 Results

Testing at MSFC accomplished the Minimum assembly sequence. Further assembly sequences were not attempted. Results from the estimation and docking can be found in Reference [12]. Docking tests worked sufficiently well to transition to reconfiguration. Tests verified that the proper state was reached and the appropriate properties were set. However, accurate closed-loop control was not achieved due to two main reasons: movement of CM outside the thruster envelope and insufficient thruster authority.

CM Outside Thruster Envelope

The movement of the CM outside of the thruster envelope occurred because the payload (Sub-aperture) was larger and more massive than the SPHERES tug. This had two primary effects. First, it caused a decrease in firing efficiency. To create a translational motion, the tug must fire in the opposite direction from the desired direction of movement to create sufficient torque (Figure 4-3). This caused a decrease in thruster authority, since some authority was being wasted.

The second effect was the impact on the determination of the thruster pairs.

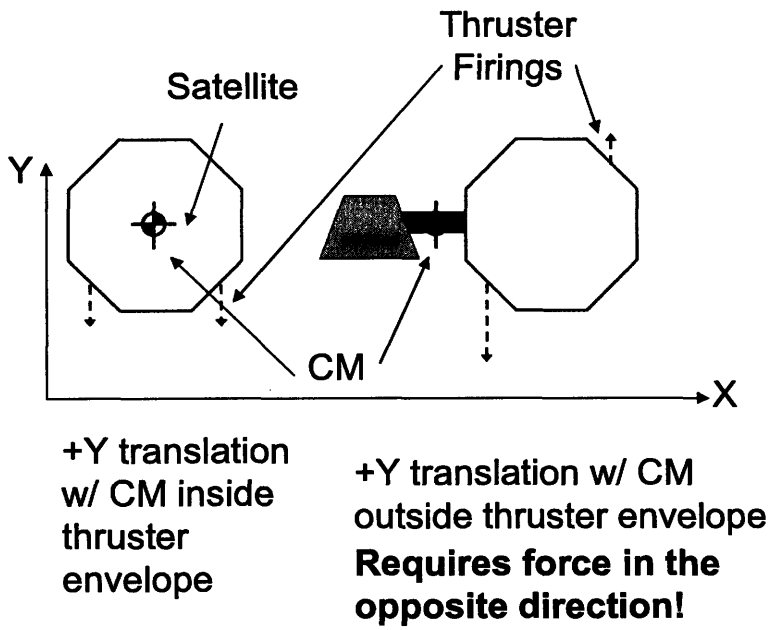


Figure 4-3: Movement of CM outside of thruster envelope

The reconfiguration implementation assumes that the thrusters are symmetric with respect to the object. For the tug+payload case, since the center of mass was located outside of the thruster window, the thruster configuration changed. The force/torque matrix develops a singularity. There is no longer symmetry in the z-axis torque and y-axis forces. In other words, in order to not have a singular matrix and to assign the thruster pairs correctly, the thrusters that produce +Y force, must provide opposing Z torques, so as to cause a pure translation. This assumption breaks down when the CM is not inside the thruster envelope because a pure rotation is no longer possible.

Insufficient Thruster Authority

The second limiting factor for the testing environment, that hampered the success of reconfiguration, was the limited thruster authority. The SPHERES thrusters were designed for microgravity operations, and only provide 0.22 N of thrust when firing two thrusters at once. This amount of thrust was minimally sufficient to maneuver one SPHERES tug around on the MSFC floor, taking into account the slopes, scratches, and other variations in the surface. When the mass was increased to be that of a tug+payload, there was insufficient authority to enable translational motion. Contin-

uous firing of +X thrusters was insufficient to cause a measurable change in position of the sub-assembly. However, there was some rotational authority demonstrated. Open loop maneuvering and docking was demonstrated.

4.2.3 Future Work

There are two aspects of future work: fixing limitations found during testing at MSFC and taking the next step in science.

Improvements to code

Two major items need to be implemented to improve the code. The first item is to update the reconfiguration mixer to not be dependent on thruster pairs. This will allow for individual control of thrusters, and not have it based on the assumption that the thrusters are symmetrically located. This will enable using the thruster locations to actually determine what the firing pattern should be based on the location of the CM. This will improve the efficiency of the mixer.

The second item is to create a propulsion stage to increase thruster authority to improve the quality of testing on flat floors. Preliminary work has already been done to identify the thrust requirement for the new propulsion stage. As an initial study, we assumed that if we could produce the same acceleration produced on the single SPHERES tug, we would have sufficient thrust authority. A rough calculation was made using the mass of the SPHERES tug system and the force of the SPHERES thrusters. A SPHERES tug system is a SPHERE + Node + two UDPs + Air Carriage. The total mass of the SPHERES tug system is 12.6 kg: includes the SPHERES tug (4.1kg), Node with 2 docking ports (3.844kg), three-puck air carriage system (4.62kg). The thrust provided by the SPHERES is a function of the regulator pressure. At MSFC Flight Robotics Laboratory flat floor facility, the SPHERES tug was run at about 30 psi, our normal operating pressure. Figure 4-4 shows the acceleration as a function of the regulator pressure. From the figure, we estimate the acceleration to be 0.015 m/s^2. Now, assuming total mass of 30kg, representing a

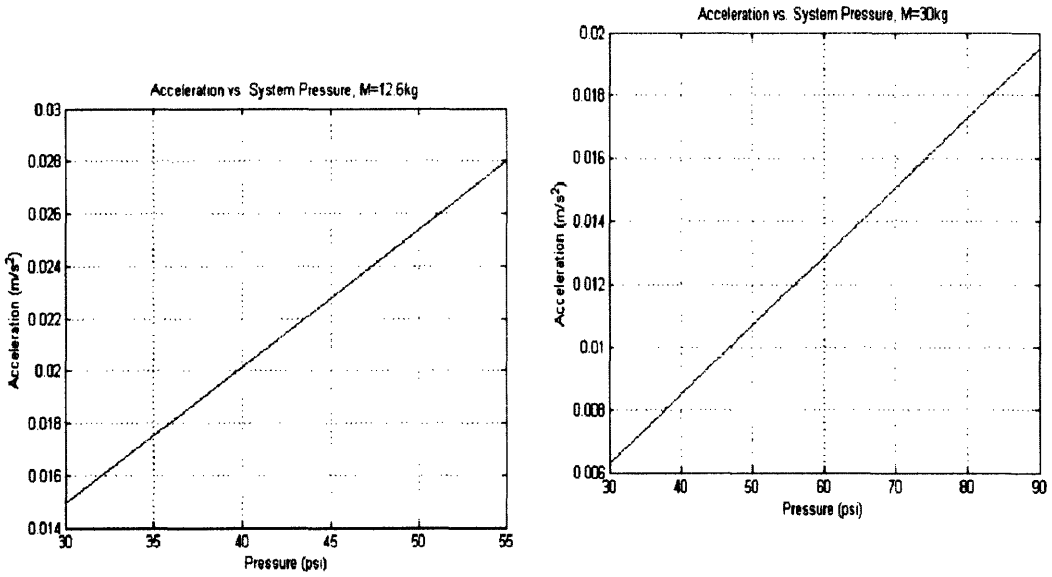


Figure 4-4: Acceleration vs. pressure for total mass of 12kg and 30kg

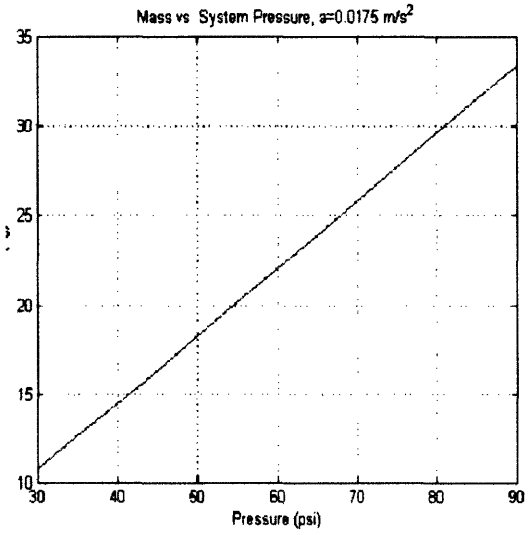


Figure 4-5: Mass vs pressure for acceleration = 0.0175 m/s^2

SPHERES tug (12.6kg) docked with a Sub-aperture (17.7kg), the graph on the right in Figure 4-4 shows acceleration as a function of pressure, when the mass is 30kg. We want to achieve an acceleration of 0.015 m/s^2 with this mass; so, we would need a pressure of at least 70 psi. Figure 4-4 shows the trend of system mass (includes SPHERES tug system) at 0.0175 m/s2. Since the tug always fires two thrusters at a time, the force is 0.11 N/per thruster * 2 = 0.22. This is a representative acceleration achieved at the flat floor. Without significant modification to the SPHERES, or the design of a separate propulsion module, we are limited to a total mass of about 25 kg. In practice, the limit might be even lower, when accounting for limitations of friction, linearization of thrust model, and other factors not accounted for.

Assembly of Flexible Bodies

A major assumption of SWARM was that all of the elements were rigid bodies. Once the tug and payload are docked, it is assumed that the new structure is completely rigid as well. In reality, this will not always be the case. All elements have some amount of flexibility, whether in the payload structure itself, or in the docking mechanism. SWARM Phase 2 proposes to study the techniques involved in assembly of a structure with flexible payloads. Figure 4-6 is the test matrix of different configurations that would demonstrate the science objectives. The design of the flexible beam module is not yet final, but it assumed to be shown excitable by the SPHERES thrusters.

4.3 SIFFT (Application #2)

SIFFT (Synthetic Imaging Formation Flight Testbed) is a testbed developed to explore the principles and requirements for the Stellar Imager mission concept. SIFFT develops and demonstrates algorithms for autonomous precision formation flight. There are four objectives for this program:

- Formation Capture: create a formation with an arbitrary initial satellite position topology

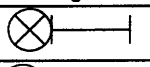
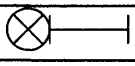
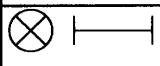
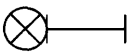
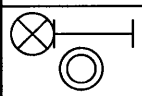
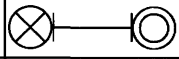
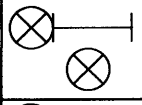
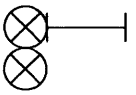
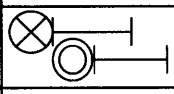
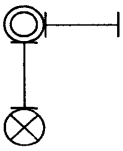
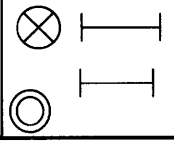
T #	Initial Config	Initial Configuration	Assembly Steps	Science Obj	Final Configuration
1	SPH+FB			Demonstrate control	
2	SPH floating FB floating (translation constrained)		Dock to SPH to FB	Dock to a flexible structure	
3	SPH+FB floating SPH fixed		Dock flexible end to fixed sphere (parallel to flexible beam)	Docking control with non-minimum phase zero	
4	SPH+FB floating SPH fixed		Dock SPH end of SPH+FB to fixed SPH	Demonstrate control docking perpendicular to SPH	
5	-SPH+FB -SPH+FB		- Dock SPH+FB (FB end) to SPH+FB (SPH end)	-Maintain control of 1st flex. beam while docking 2nd perpendicularly	
6	-SPH floating -2 FB floating -RWA SPH fixed		-SPH dock to FB -Dock FB end to RWA SPH -SPH Undock -SPH dock to 2nd FB -Dock FB perp. On RWA SPH	-Combine all science objectives in a full assembly sequence -Explore effects of undocking on dynamics and control	

Figure 4-6: Test Matrix for SWARM Phase 2

- Formation Geometry Maintenance: controllers to maintain formation within tolerances

- Formation Geometry Reconfiguration: be able to change to a separate, distinct geometry, independent of the type of geometry

- Synthetic Imaging Maneuvers: perform maneuvers to map u-v imaging plane

This work focuses on the geometry reconfiguration objective, with some implementation of geometry maintenance. Formation reconfiguration is defined as the method by which an array of satellites in an initial configuration are safely and optimally moved to achieve a desired configuration. The movement is a series of maneuvers commanded to the spacecraft that transition it from the initial state to the final state. Hence, important aspects that should be considered are: target state update, path planning from initial to final state, and obstacle avoidance. This work focuses on the target update, with future work intended to address the remaining two aspects.

Work culminated in a hardware demonstration at MSFC Flight Robotics Laboratory's flat floor facility in September, 2006. The week long testing incrementally added complexity, to finally demonstrate a three-satellite triangle configuration that

rotates and expands. This section presents the implementation methodology, test description, and results from the MSFC testing. Plans for future work are also addressed.

4.3.1 Mission Concept

The objective of SIFFT is to advantage technology for the Stellar Imager mission. Stellar Imager is a UV/Optical deep space telescope to image stars and observe the universe at a 0.1 milli-arcsecond resolution. The current concept design for SI is a Fizeau Interferometer with 20-30 one-meter primary mirrors, distributed over a parabolic virtual surface. The diameter of the virtual surface can vary from 100m up to as much as 1000m, depending on the angular size of the target object. The focal length linearly scales with the diameter of the primary array, with focal lengths of 1 km and 10 km corresponding to diameters of 100m and 1000m, respectively. The mission concept requires the expansion (or contraction) of the entire array every time a new target is selected. Therefore, it is important to demonstrate the key maneuvers of formation expansion and formation rotation.

4.3.2 Hardware Implementation

The SIFFT testbed consists of three SPHERES satellites. In order to determine an optimal formation for the three satellite, a Golay-3 (equilateral triangle) formation was used, which are optimized for compactness in the array's auto-correlation function. The auto-correlation function is a set of u-v points in the Fourier dimension. The advantage of having compact arrays is that one can obtain full u-v coverage with apertures of the smallest size, when the array is used in Fizeau interferometric mode. However, SI does not operate in this mode. Therefore, the array must be expanded and rotated to achieve full u-v coverage.

The above configuration (Figure 4-7) was used as the basis for designing the demonstration test. The demonstration test had the following sequence:

- Kalman Filter initialization

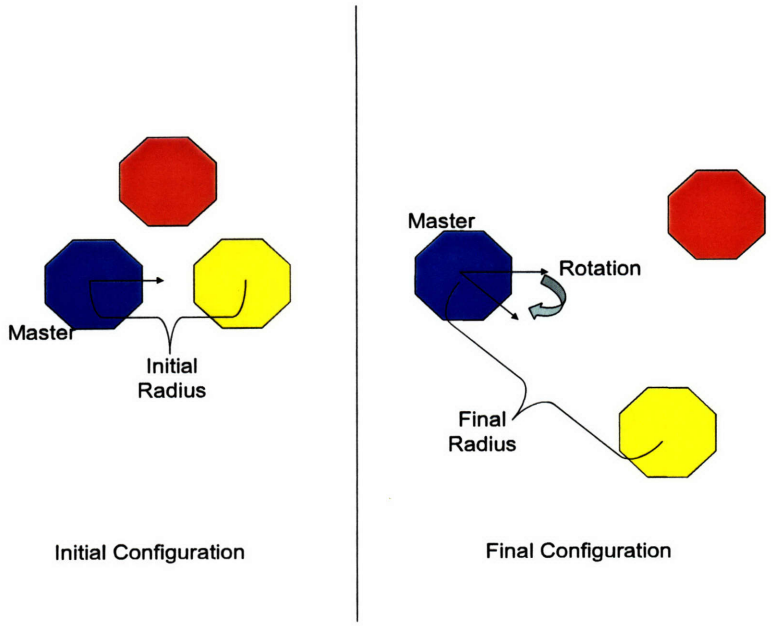


Figure 4-7: Schematic of rotation and expansion maneuver

- Attitude control enabled
- Move to Formation #1
- Move to Formation #2

Kalman Filter Initialization

Out of the three satellites, one satellite was designated the Master. In a true space environment, such as the Lagrange points, there is no global reference frame or sensing system, as there is in LEO or on Earth (i.e., GPS). Therefore, a reference within the system must be identified in order to define the state of the system. This is also the scenario at the MSFC flat floor. In the lab facility at MIT, the five beacons provide a test volume, inside which the full state can be determined relative to the center of the test volume. In MSFC, this frame is not available, so all state determination must be done relatively. Thus, one satellite is designated as the Master, and the two remaining satellites are labeled the Followers. The Followers estimate their states by sensing the beacons on the Master satellite.

The UDP relative estimator (described in Section 2.3.3) is used in this test ses-

sion. A ten second initialization period is allotted for the filter to converge. The initialization is an important step because the beacons on the UDP have a finite cone of reception of 30° half-angle. All sensors on the Follower satellites are used, thus the satellites can be in any initial attitude, as long as their position is within the cone of reception of the reference beacons. For this work, it is assumed that the non-Master satellites start positioned within the cone of reception of the beacons.

Attitude Control enabled

After Kalman Filter initialization, the Follower satellites enable attitude control. This initial maneuver allows for a small maneuver to assess the controllability of the satellites. For example, beginning with attitude control only allows for a check of the convergence of the state, prior to large slewing maneuvers to change positions, which can help to prevent collisions. The attitude control is enabled using a simple PD attitude controller. The Follower satellites are oriented to match their desired attitude with respect to the Master. A PD controller was selected because of its simplicity, since this maneuver only needs to get the satellites in the ballpark of the desired attitude. This maneuver terminates after six seconds.

Formation #1

After achieving the proper attitude, the satellites move into the first formation. A PID position controller is used in conjunction with a PD attitude controller. The satellites are given a fixed time in this maneuver to traverse to the desired position and hold there. In a real scenario, there would likely be two maneuvers, one to get there and one to stay there. Since our test was primarily to determine if we could get there and leave, the length of time spent at each formation was not critical.

Formation #2

This maneuver updates the position targets to correspond to the second formation. The same controllers are run to control to this new position target. In the demonstration test, the Master SPHERE employs an estimator using solely gyroscope infor-

Table 4.2: Test Matrix with sub-tests performed and success rate information

Sub-Test	T	W	R	F	Success	H/W Fail	S/W Fail	% Success
3	X				4	6	4	28.6
3				X	3	2	2	42.9
4		X			1	2	0	33.3
4			X		5	8	3	31.3
4				X	3	6	4	23.1

mation to execute a 30 degree rotation. This rotation in the Master causes a rotation of the entire array.

4.3.3 Results

Testing at MSFC was done incrementally, with each test successively adding a maneuver or a satellite. The full list of sub-tests performed are given below.

1. Reference Fixed, one Follower, Formation #1
2. Reference Fixed, one Follower, Formation #1 and Formation #2
3. Reference Fixed, two Followers, Formation #1 and Formation #2
4. Reference Attitude Control, two Followers, Formation #1 and Formation #2 (w/rotation)

Table 4.2 shows the sub-tests performed over the course of the week, with statistics on the performance. Sub-tests 1 and 2 were performed in the lab at MIT prior to arriving at MSFC, and not repeated as an individual test at MSFC.

The overall success rate of the final sub-test (sub-test 4) was 28%. There were several reasons that the tests did not always succeed:

- Estimator Divergence: The tests on Monday, and initially on Tuesday, were limited by estimator divergence. On Tuesday, it was determined that the one of the UDP avionics boards was not utilizing the receivers properly. This board was previously a development board that had been recently converted to be a fully functional board. This board was switched to be placed on the Reference SPHERE, and only needed to enable the beacons. After this switch, the

estimator began working. However, later tests revealed that a large radius of formation also degraded estimator performance, since the power from the beacons drops off as a function of distance squared. The performance was found to degrade after a radius of about 2 m. Therefore, the final radii for the formations were selected to be 1.25m and 1.85m.

- Hardware/Floor Conditions: The majority of the tests were hindered by the finite resources available, such as running out of propellant on either the carriage or the satellite. Additionally, the satellites were sensitive to scratches and slopes in the floor. A higher bandwidth controller, combined with a propulsion module with more thrust authority would decrease the satellite sensitivity to these parameters.

- Communication Loss: Another factor that limited the success was communication loss. One or both satellites frequently failed to establish communication with the laptop. This resulted in test start times that were not synchronized. The communication loss could have been triggered by the viewing angle of the antenna to the satellite. Due to the set-up on the floor, the antenna was often more than a meter away from the center of the formation.

Though there were limitations to the testing, overall the main objectives were met. Tests successfully achieved the Formation #1 from arbitrary initial conditions, maintained the formation, and switched to Formation #2. Accurate positioning was demonstrated, even under the conditions of large translational movements by the Master satellite. Figures 4-8 to 4-10 show results from MSFC testing. Figure 4-8 shows the results for Sub-Test #3, two Followers with respect to a fixed reference ((0,0) on the plots). There is a slight steady state error seen, on the order of 10 cm. The precision of control, measured by the oscillation at the desired target point, is approximately 5 cm. The straight line in the figure shows the clean transition between the two formations. Figure 4-9 shows the results of Sub-Test #3, delineated by maneuver. Figure 4-10 shows the results of Sub-Test #4 where the reference is now floating and rotates 20° in Formation #2. Only a single satellite is shown to highlight

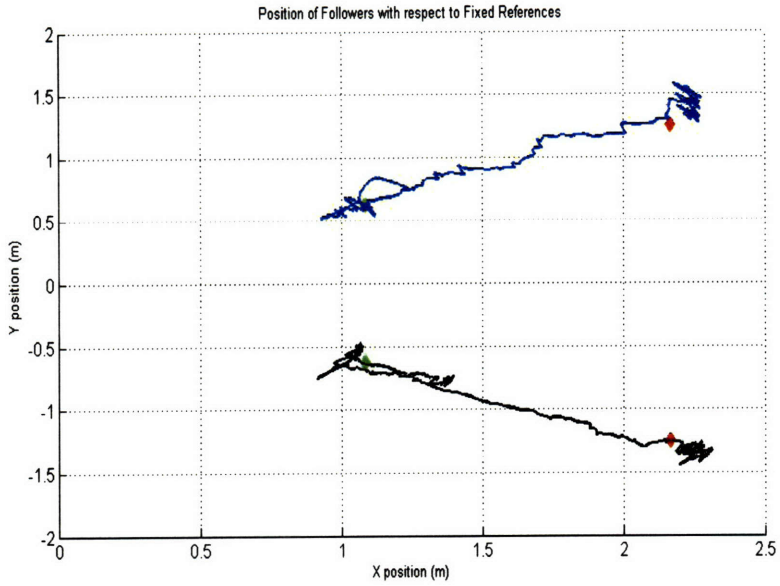


Figure 4-8: XY position of Followers with fixed reference

for clarity of presentation. The curved trajectory of the black lines (Formation #2) represents the rotation of the frame. The steady state error, determined by graphical inspection, is on the order of 10 cm. The precision of the control is about 5 cm. Overall, the performance of the tests was found to be on the order of a few centimeters.

4.3.4 Future Work

The work described has implemented one of the three aspects of formation reconfiguration. The update of the target vector demonstrates that the satellites can update their targets and maneuver to the new desired position. However, for a real space telescope, simply maneuvering is not enough. The maneuvering must be implemented such that it is safe and optimal. These two requirements add in the additional complexity of obstacle avoidance and path planning. In order to implement these two aspects, further development must occur. The method currently implemented is decentralized control; each satellite separately controls its own state, without knowledge of the states of the other satellites. In order to perform effective obstacle avoidance, each satellite must have a way of detecting objects in a certain radius around it. This can either be accomplished by centralized control, such that the master knows where

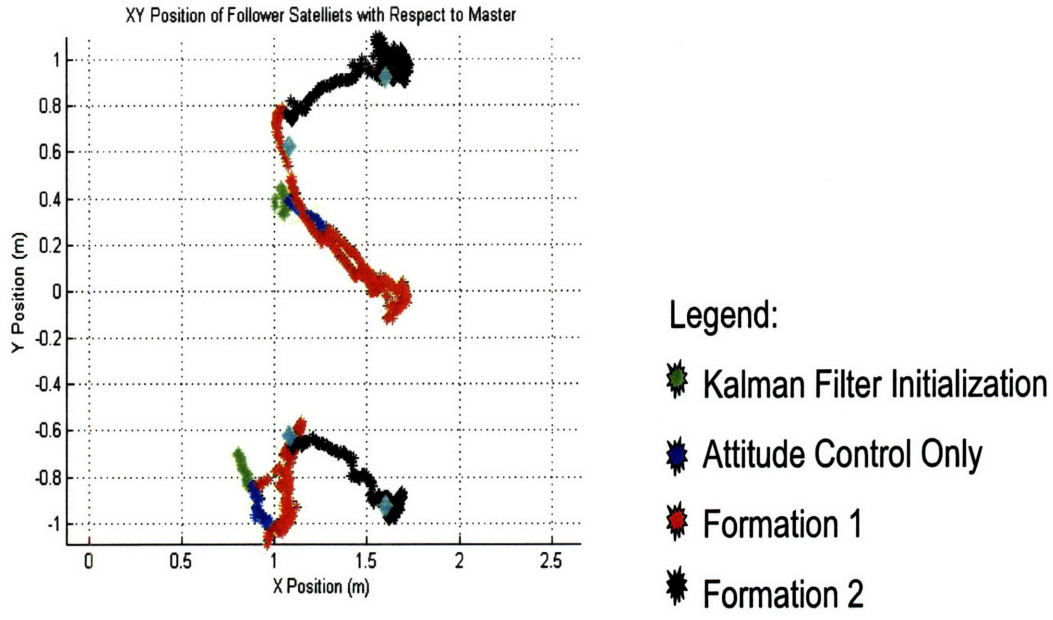


Figure 4-9: XY Position of Followers delineated by maneuver

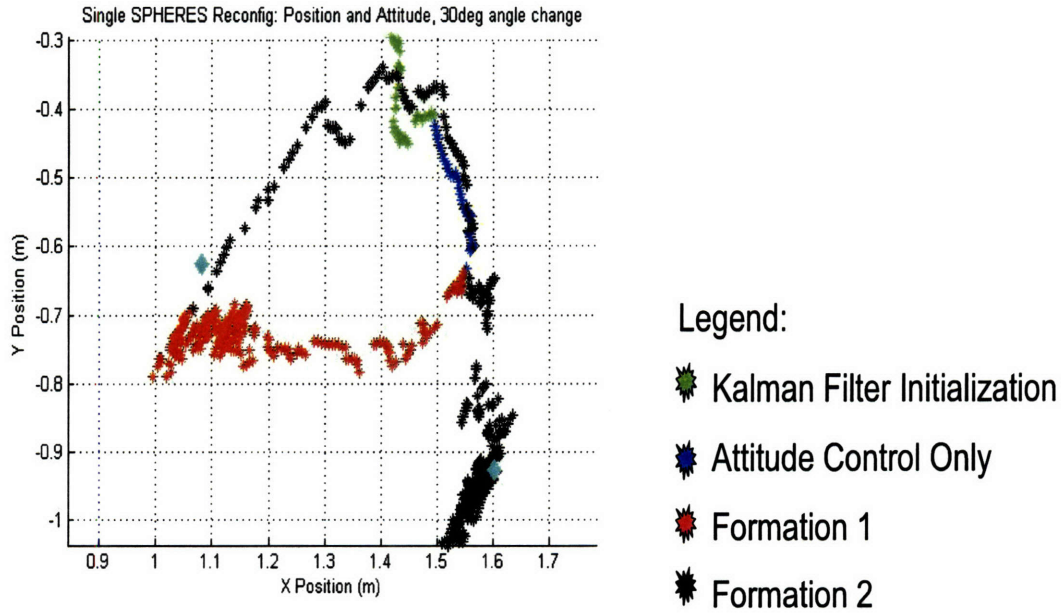


Figure 4-10: XY Position of Single Follower with Floating and Rotating Reference

each satellite is and can prevent collisions. This can also be accomplished by placing sensors on each satellite such that it can sense objects in its local proximity.

Optimality can be implemented by incorporating LQR controllers instead of nominal PD and PID controllers. Also, the controllers can be updated to include different constraints, such as minimum fuel and minimum time. This results in Bang-Off-Bang and Bang-Bang controllers respectively. Trajectory planning can also be implemented, such that it is not a straight line path between the initial and final positions. Especially when the number of satellites increases, safe trajectories will be key. This also becomes more important if the formation is contracting or if satellites have to cross each other.

4.4 Mass Property Update (Application #3)

Robotic assembly via assembler tugs involves using a separate spacecraft (tug) that executes maneuvers to move the pieces of the satellite (payload) to their appropriate locations. These maneuvers are calculated by the tug based on the given start and end points using the tug's on-board controller. When the tug docks to a payload, the mass properties of the system change. The mass, inertia, and center of mass of all must be updated to be the values of the tug-payload system. Hence, for an effective controller, the tug must update its mass properties at every docking maneuver to account for its current configuration. This requires the tug to have knowledge of the mass properties of the payload and docking configuration of the payload with respect to the tug. This study assumes that a single tug will be assembling or servicing many payloads, in number and type. Three main methods for obtaining the mass properties of the payload are considered.

1. The first method is to measure the properties of the payload prior to launch and hard code those values into a database on the tug. The tug accesses this database at every docking with a payload.

2. The second method is to measure the properties of the payload prior to launch

and store those values on the payload. During assembly, after a docking maneuver, the payload transmits its mass properties to the tug.

3. The third method is for the tug to use system identification techniques to characterize the mass properties of the tug-payload system on-orbit during assembly.

Each of these methods has operational implications, such as the choice of a communication system or the amount of fuel required to be launched. The choice of a mass property method, or combination of methods, could have a significant operational impact on the assembly. The parameters that could be affected include, but are not limited to, precision of the controller, time to assemble, risk, and overhead mass. This section investigates the first order effects of mass property update methods on on-orbit assembly.

4.4.1 Simulation Architecture

Six mass property update options are evaluated for their operational impact. The six options are:

1. Use hardcoded values for each payload type
2. Use communicated values
3. Use system identification
4. Use communicated values with hardcoded values
5. Use system identification with hardcoded values
6. Use communicated values with system identification

The simulation attempts to capture the interaction between these update options and the assembly process. For options using the hardcoded method (1,4,5), the parameter of interest is the words available in memory of the tug. The number of types of payloads is used to calculate the total memory space needed on the tug; it is compared to the amount available to see if the design is feasible. For options using the

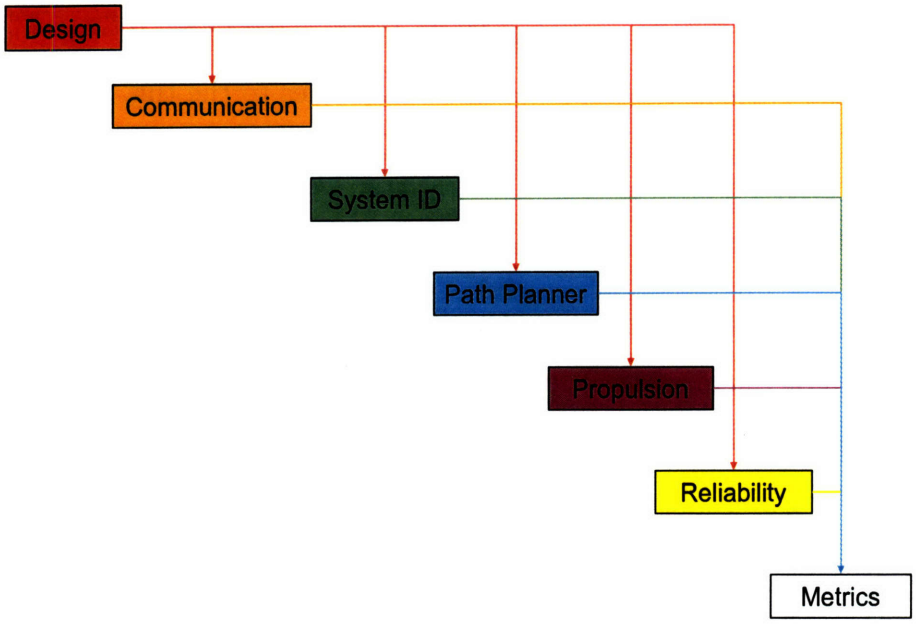


Figure 4-11: Mass properties update method impact simulation flow diagram

communicated method (2,4,6), the required mass of the RFID communication system is added to the tug and accounted for as an overhead mass. For options using system identification (3,5,6), the overhead mass consists of the extra fuel used to perform the identification maneuvers. Also, the time to perform these maneuvers is added to the total time for assembly.

Each option is considered for ranges of design variables. The design variables are number of different payloads, number of each type of payload, mass of each payload, and total capacity. The design variables are assigned based on the scenario. The design variables and parameters are incorporated into a simulation that runs each scenario and returns overhead mass, overall reliability, and total time to assemble for a fixed fuel mass available. The simulation flow is given by Figure 4-11. The scenario parameters are fed into the Design Module which converts them to the appropriate design variables. The design variables are fed into a series of modules that model the assembly process. The following sections detail the scenario selection, the rationale for the design parameters, and description of the modules.

Scenarios

Two scenarios are considered in this paper: a large space telescope and a fuel depot. These two scenarios are chosen to represent two applications whose desired functionality necessitates on-orbit assembly. The two scenarios are likely candidates for on-orbit assembly, but do not necessarily span the range of all possible requirements.

The need for on-orbit assembly for a space telescope is driven by the need for longer baseline telescopes. Current technology is nearing the limit as to what is possible to deploy and launch as a single aperture. The telescope design modeled in this study is a filled aperture segmented telescope that is analogous to the James Webb Space Telescope (JWST). The telescope is composed of hexagonal mirror segments. The following design variables are considered for the telescope: diameter of telescope, areal density of the telescope, and number of hexagonal segments.

The development of a fuel depot is mentioned in NASA's Moon/Mars vision. The existence of a fuel depot in space would facilitate frequent manned missions to the Moon or beyond. Fuel would be able to be launched separately, on EELVs, leaving additional mass for other cargo. The spacecraft would rendezvous at the fuel depot to obtain the needed fuel for the given mission. Thus, the initial launch mass of the manned mission is decreased because they only need to carry enough fuel to get to the depot, instead of to the final destination. The architecture of the fuel depot in this scenario is an array with a backbone structure. The tanks connect to the backbone structure as depicted in Figure 4-12 (right). The chosen elements stored at the depot are liquid hydrogen (LH2), liquid oxygen (LOX), and liquid methane (LCH4). These three elements were chosen as likely substances to be used in propulsion systems, one oxidizer and two fuels. Seven combinations of elements are considered to span a range of depot demand: only LH2, only LOX, only LCH_4, 50% LH_2 and 50% LOX, 50% LCH_4 and 50% LH_2, and 50% O_2 and 25% LCH_4 and 25% LH_2. These combinations determine the design variables that capture the mass of the tanks and the number of types of tanks. Other design variables considered are the maximum storage capacity of the depot and the total number of tanks. This study does not consider the effects

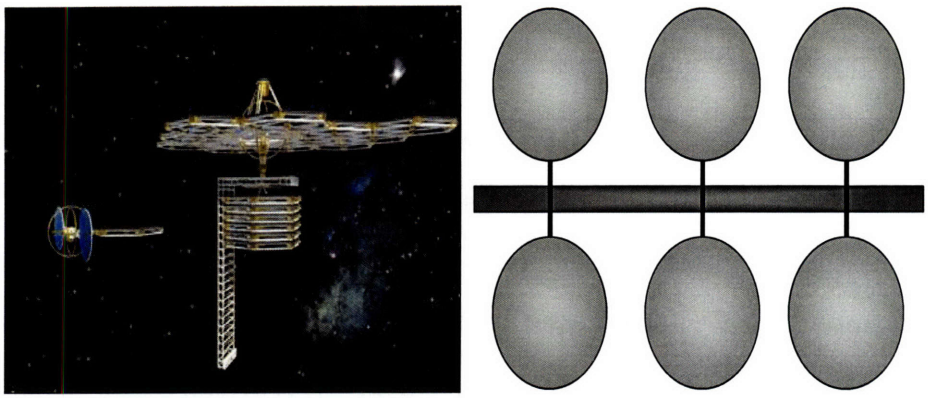


Figure 4-12: Images of Scenarios used: Space Telescope (left), Fuel depot (right)

of boil-off.

Tug Baseline

The operational impact of different mass property update methods depends in part on the capability of the assembler tug. For this study, SPHERES has been used as the benchmark capability when possible. Parameters of interest are the memory space, specific impulse, fuel expended for system identification maneuvers, and time to execute system identification maneuvers.

The memory space limits the amount of information that can be stored; hence, it serves as an upper bound on the number of mass property values that can be hardcoded into the assembler tug's memory. The maximum number of words that can be stored on SPHERES is 52,000 words. The core operational software requires a minimum of 30,000 words. Hence, the remaining 22,000 words can be used to store mass property values. For each tug-payload system, the mass property values that are stored include mass, center of mass, moments of inertia, inverse moments of inertia, and thruster locations with respect to the new CM. Distributed among the sets of data is the software for selecting a payload and assigning the new mass property values. After implementing this set on SPHERES for a variety of objects, such as docking ports and air carriages, the minimum number of words necessary per set is 1000 words. This corresponds to the SPHERES being able to store the mass property information for 22 distinct payloads.

The SPHERES cold gas system has a specific impulse of 35s to 40s . Though this specific impulse is much lower than those expected for an assembler tug, 40s is used as the baseline I_{sp} to compare between options and also to be consistent with the system identification fuel and times used. The system identification parameters, fuel expended and time per identification test, are derived from test results from performing system identification on the SPHERES. The test consists of the SPHERES satellite firing the thrusters in a known pattern. The resultant movement can be measured with sensors and used to calculate the mass properties. The amount of fuel consumed is calculated by using the *fuel-remaining* variable in the SPHERES software. This variable estimates the fuel used by adding up the thruster on-times and assuming an average fuel mass loss of $3.521 * 10^{-7}$ kg/ms. The average fuel expended obtained from these methods is eight grams. The corresponding time for the system identification test is three minutes. Future work will include scaling the fuel mass and time values for system identification to be of the order of magnitude performed by an on-orbit tug.

Two design parameters that have not been benchmarked from SPHERES are reliability and the system used to communicate values from the payload to the tug. The communication system selected is an RFID-based system. Each payload has an RFID tag on the docking interface. The tag contains information of the mass, moments of inertia, and center of mass of the payload with respect to the corresponding docking interface. Assuming these twelve numbers (one for mass, nine for moments of inertia, three for CM) are coded to an accuracy of three significant figures, the corresponding memory required for the RFID tag is approximately 203 bits. Comparison of current commercially available RFID tags resulted in a maximum baseline tag of 76.2 mm^2 and 0.42 mm height. Assuming an average density of silicon, the corresponding weight is 5.27 g. The selected RFID tag is passive and thus does not required any power system on the payload. A baseline RFID reader was selected to be placed on the tug to interface to the payload tag. The specifications of the chosen reader are 11.98 in by 9.0 in by 1.72 in. The weight of the reader is approximately 4 lbs.

The choice of RFID over the SPHERES communication system is based on the

desire to keep the payload passive. A preliminary analysis using an antenna-based system on the payload to communicate the values resulted in significant overhead mass. The overhead mass accounted for the antenna mass, and the mass of the power system needed to keep the antenna active. Options using the communicated method performed approximately a factor of two worse than all other options and were impractical solutions. Preliminary analysis of RFID showed it produced feasible and practical results.

Reliability values are used as a measure of the success rate of the assembly process. The reliability considered is the reliability of the update method. The hardware used for modeling the reliability for hardcoded, communicated, and system identified values, are the reliability of the data processor, RFID system, and thrusters, respectively. For the data processor, the reliability value used is 0.9999. This number is derived from observed performance of the SPHERES microprocessor and compared against nominal reliability values for satellite command and data handling subsystems. The reliability values obtained for RFID systems from literature range from 0.8 to 0.997. For this study, an average reliability of 0.9 is used. This captures the inherent newness of RFID technology. The reliability of the thruster valves is 0.999 and is based on observed performance of the SPHERES thrusters over an operational lifetime of three years.

Module Descriptions

The simulation consists of six main modules: Design, Communication, SystemID, PathPlanner, Propulsion, and Reliability. Each module is run once, in the given order, for each scenario and set of inputs.

Design: The Design module takes in the scenario parameters and converts them to design parameters. A unique version of this module exists for each scenario. For the telescope scenario, the inputs are mirror diameter, number of loops of segments, and the areal density of the telescope. The number of mirror segments is computed using

the number of rings, according to Equation 4.1.

$$N_{seg} = 6N_{rings} + \sum_{i=1}^{N_{rings}-1} 6i \tag{4.1}$$

The number of segments is used, in conjunction with the mirror diameter, to calculate the effective area of each segment.

$$A_{seg} = \frac{\frac{1}{4}\pi D^2}{N_{seg}} \tag{4.2}$$

The mass of each segment is the effective area of segment times the areal density. Also, the effective area allows us to calculate the side length of each hexagon piece. The side length is given by Equation 4.3.

$$s_{seg} = \sqrt{\frac{2A_{seg}}{\frac{3}{\sqrt{3}}}} \tag{4.3}$$

The number of types of segments for the modeled telescope is limited to two; all of the segments are identical except the center piece requires a hole for the light to pass from the secondary mirror into the optics. The outputs of this module are the design parameters of mass of each segment, number of segments, number of types, and side length of each mirror segment.

For the fuel depot scenario, the inputs are storage capacity of the depot, number of tanks, and specified combination of elements stored at the depot. The number of elements is used to specify the number of types of tanks. For a single element depot, the amount per tank is calculated by assuming constant distribution.

$$M_{tank} = \frac{Capacity}{N_{tanks}} \tag{4.4}$$

The size of each tank is found by using the mass and liquid density of the elements to find the volume. Assuming spherical tanks, the radius is then given by Equation 4.5.

$$r_{tank} = \left(\frac{3}{4}\pi V_{tank}\right)^{\frac{1}{3}} \tag{4.5}$$

For a multi-element depot, the mass of each tank is held constant and the number of tanks for a given element corresponds to the percentage storage of the element. The outputs for this module are the number of types of propellants, number of tanks for each element, mass per tank, and radius per tank.

Communication: The Communication module determines the effect of using the communication update method by calculating the overhead mass associated with the communication system to transmit the values from the payload to the tug. Based on the values for the mass of the RFID reader and RFID tag, the mass of the communication system is given by Equation 4.6.

$$M_{comm} = M_{reader} + (N_{payload} * M_{tug}) \tag{4.6}$$

System ID: Similar to the Communication module, the System ID module calculates the effect of using the system identified method. Using SPHERES as a baseline, the results of a system identification test run on SPHERES are the basis for the values used in this module. Based on experimental results, the time per system identification test is three minutes and fuel used per system identification test on SPHERES is eight grams. The total time and fuel usage is calculated assuming one test per payload. These values are then incorporated into the time to assemble and overhead mass.

PathPlanner: The PathPlanner module takes in the defining measurement of the payload (ex. side length of mirror segment for telescope) and creates a matrix of start and end points that reflect the tug's trajectory during assembly. A unique version of this module exists for each scenario. For the space telescope, the mirror segments are initially stacked vertically in a tray. The tug starts at the top mirror segment. The module assumes a straight line between the CM of the mirror segment in the stack and the CM of the mirror segment in its assembled position. The tug then returns to the next mirror in the stack. The sequence of maneuvers is graphically depicted in Figure 4-13.

The same method is used to calculate the paths for the fuel depot scenario. The inputs to the path planner are the combination of elements in the depot, the radius

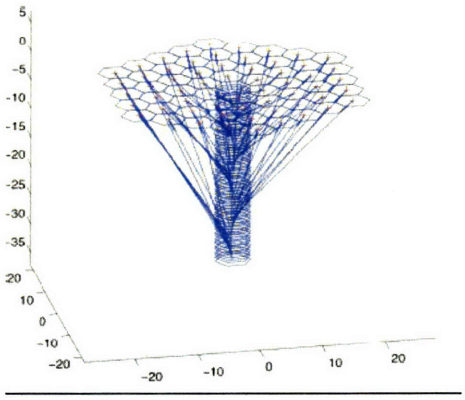


Figure 4-13: Maneuver paths of tug for assembly of telescope

of each element tank, and the total number of tanks. The reference length used to set the assembled positions of the tanks is the maximum radius of the three element tanks. The spherical tanks are initially stacked vertically. In the final assembled configuration, the tanks are in an array with a center backbone. One tank is docked on each side of the backbone structure. The separation distance between the assembled positions of the tanks is set to the maximum tank radius plus a separation distance of one meter. The total number of paths that must be completed is twice the number of payloads it must assemble because for each payload, the tug must complete a forward and return journey.

Propulsion: The Propulsion module takes in the path and computes the total assembly time for a set mass of fuel. The inputs to this module are the path, the mass of the tug, and the total fuel mass available. The module assumes an I_{sp} of 40s (baselined from SPHERES) and a firing efficiency of 70%. The firing efficiency represents the percentage of the fuel mass that is actually used to impart a velocity change. The total fuel mass available is the mass of the propulsion system. A tank fraction of 30% is used to account for tanks, thrusters, fuel lines, etc. Thus, the mass of propellant available is given by Equation 4.7, where f is the tank fraction used.

$$M_{prop} = 0.9 M_{fuel} (1 - f) \tag{4.7}$$

The module assumes a bang-off-bang firing scheme. The coast time is assumed to be 50% of the total maneuver time. The mass of propellant used for each maneuver

is based on a parabolic distribution of fuel, given by Equation 4.8 where i is the maneuver number and j is parabolic coefficient, and e is the thruster efficiency.

$$M_{prop,i} = \frac{M_{prop}e}{N_{trips}^{n(j)}\left(i^{n(j)} - (i-1)^{n(j)}\right)} \tag{4.8}$$

The corresponding maneuver time is derived from the rocket equation, as shown in Equation 4.9 where the path length is the sum of the squares of the difference between the start and end points, c is the coast fraction, and M_{total} is the total mass of the tug-payload assembly.

$$\Delta t_i = \frac{4L_{path_i}}{(c+1)I_{sp}gln\left(\frac{M_{prop_i}}{M_{total}}+1\right)} \tag{4.9}$$

The module loops through a range of parabolic coefficients and selects the optimal coefficient that minimizes the total time. The total time for the assembly is simply the sum of the maneuver times.

$$T_{assem} = \sum_{i=1}^{2N_{tanks}} \Delta t_i \tag{4.10}$$

The propulsion module outputs the minimum time for assembly, maximum thrust, maximum acceleration, and maximum total impulse.

Reliability: The reliability for each of the options is calculated based on the reliabilities of the corresponding hardware as given in Equation 4.11.

$$R_1 = R_{CDH} \quad R_2 = R_{RFID} \quad R_3 = R_{thruster} \tag{4.11}$$

For an option that includes more than one method, it is assumed that the methods are used in parallel. Hence, the reliabilities are given by Equation 4.12.

$$\begin{aligned} R_4 &= 1 - [(1 - R_{CDH})(1 - R_{RFID})] \\ R_5 &= 1 - [(1 - R_{CDH})(1 - R_{thruster})] \\ R_6 &= 1 - [(1 - R_{thruster})(1 - R_{RFID})] \end{aligned} \tag{4.12}$$

Table 4.3: Parameter Ranges for Simulation Run

Telescope					Fuel Depot				
Parameter	**Min**	**Inc**	**Max**	**Units**	**Parameter**	**Min**	**Inc**	**Max**	**Units**
Mirror Diameter	10	5	50	m	Storage Capacity	500	250	4000	kg
Number of Loops	1	1	6	n/a	Number of Tanks	10	50	400	n/a
Areal Density	20	5	40	kg/m^2	Variation of M_{seg}	0	5	25	%
Variation of M_{seg}	0	5	25	%	Combination of elements	1	1	7	n/a

Since these reliabilities are based only for the update option, they do not vary for each scenario or assembly configuration.

4.4.2 Results

Three primary metrics are used to analyze the results: overhead mass, time to assemble, and reliability. These metrics are also compared to number of objects and capacity, which represent the magnitude of the assembly task. Overhead mass is calculated as the sum of the additional mass that is required to support the update option. This consists of the mass associated with the RFID communication system, as well as the extra propellant associated with performing the system identification maneuvers. The time to assemble is based primarily on the path lengths for each maneuver; however, additional time is included to account for the time to perform the system identification maneuver. The reliability is calculated solely based on the update option, as discussed in the previous section.

Parameter Ranges

The simulation is run for two scenarios over a range of values. Table 4.3 lists the parameters varied for each scenario and the applicable ranges. For both scenarios, the mass update option is varied from one to six, and the mass of the propulsion system is varied from 100kg to 250kg, in 10kg increments.

Telescope

Figure 4-14 shows the simulation results for the given parameters specified in Table 4.3 for the space telescope. Preliminary results show that Option #1 performs better in terms of overhead mass. Overall, using the system identification method leads to

slightly smaller overhead masses when compared to the communicated method. Option #6, which is a combination of system identification and communicated methods, performs the worst. This is expected since it combines the overhead mass of both methods. However, the overlap between Option #6 and the communicated options indicate that it is not a strong difference in performance. Results also show that options using the system identification method increase faster in overhead mass as a function of the extent of the assembly, as modeled by the number of objects. In terms of reliability, Option #2 demonstrates a significantly poorer performance compared to all other options. The remaining five options are essentially indistinguishable in reliability performance given the limits of the modeling. The vertical bands reflect the varying number of objects. The increase in total time for assembly is a consequence of the increasing number of payloads to assemble, rather than any update option employed.

Fuel Depot

Figure 4-15 shows the simulation results for the fuel depot scenario for the parameters specified in Table 4.3. The results for the fuel depot parallel those found for the space telescope. The results for reliability are identical since the modeling is not based on any scenario or design parameters. Once again, Option #1 using the hardcoded method performs best for overhead mass. There is a more noticeable difference in the performance of Option #1 for the fuel depot, than for the telescope scenario. Options using the communicated method have a slightly larger overhead mass. However, the scaling of overhead mass with the extent of assembly shows that methods with system identification scale faster than those with other methods.

Future Work

In conclusion, results show that the hardcoded method performs slightly better in this simulation. Options using the communicated method perform slightly worse in terms of reliability, as is expected from the low technology readiness of RFID for space applications. Though some trends are noticeable, to first order, all options seem to

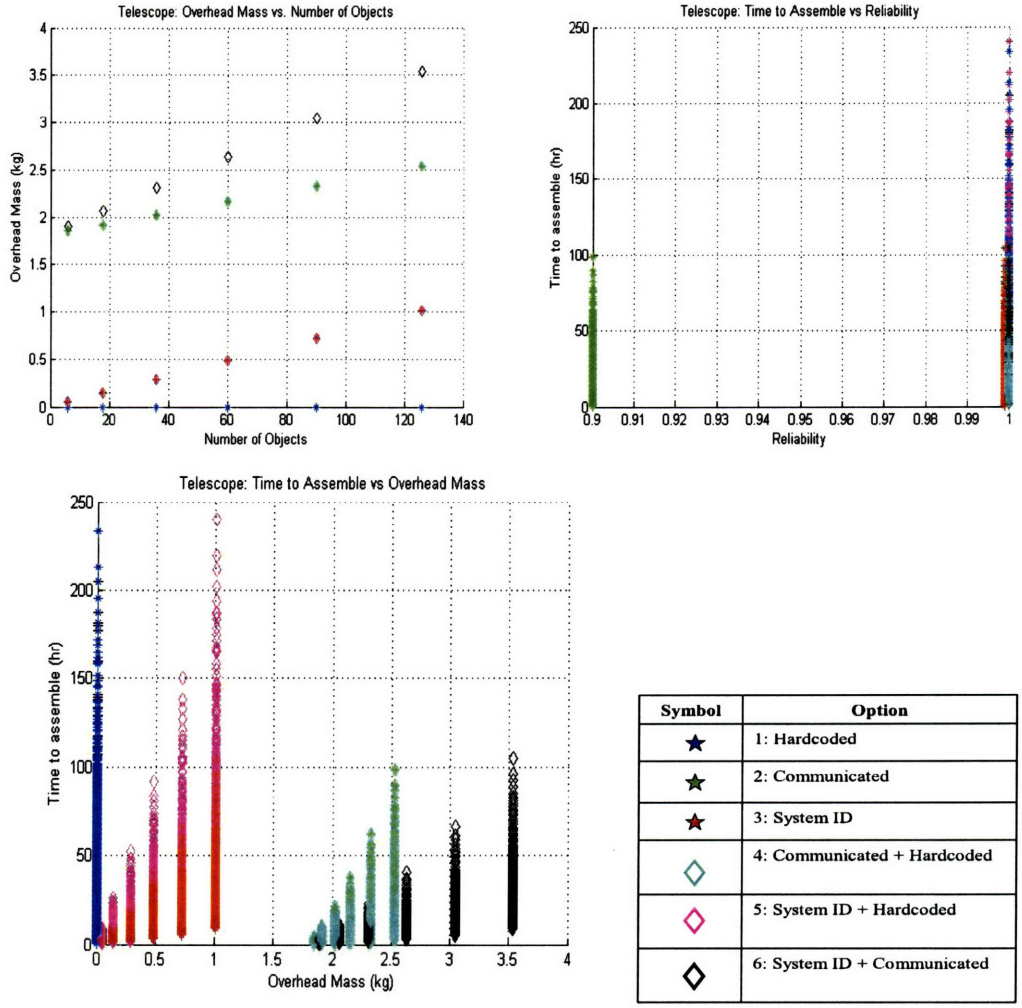


Figure 4-14: Telescope simulation results

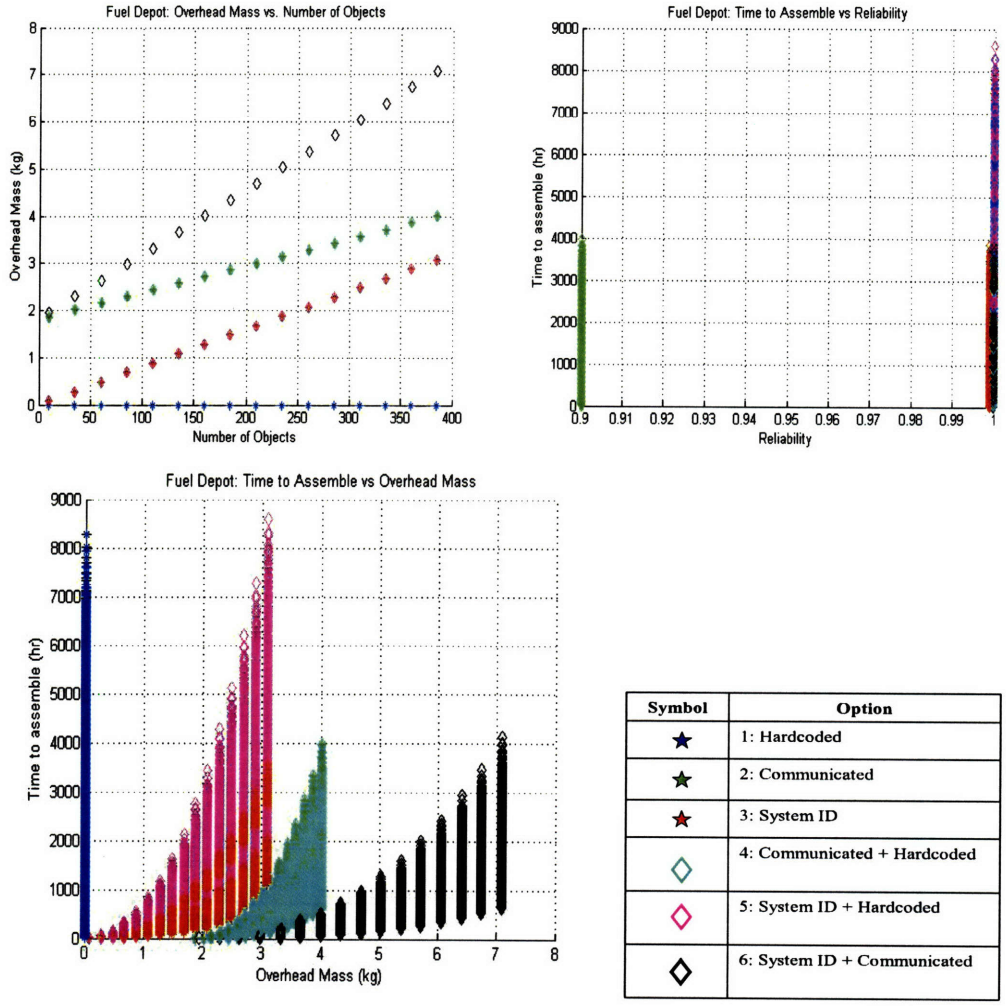


Figure 4-15: Fuel Depot Simulation Results

perform similarly. Further study is warranted to determine the effect of some modeling parameters on the performance of the methods. Future study should include scaling the assumptions from the baseline SPHERES values to those more appropriate for a space tug. This would include a more reasonable estimate for specific impulse, fuel expended for system identification, and time for system identification. Optimization techniques could be used to minimize the fuel and time spent by including it as part of the path maneuvers. Also, further study should be conducted to identify the effect of variations of the coded mass properties, whether on the tug or payload RFID, to actual mass properties. This could cause some performance issues in how precise the tug controller is able to target a particular location. The decreased performance could have an impact on highly sensitive payloads, such as mirror segments. Detailing the interaction when using multiple methods might also lead to some insights on a preferential combination of methods. All of these trades should be analyzed for a set of scenarios that encompass the range of requirements of on-orbit assembly. This includes extending the number of payloads, number of types of payloads, and sensitivity of payload. Overall, all options performed similarly with a slightly better performance for hardcoded methods. Further work as described should be conducted to distinguish second-order differences between the methods.

4.5 Modularity Analysis (Application #4)

4.5.1 Preliminary Modular Design

The preliminary modular design currently implemented is based on a Golay-3 Fizeau interferometer. This design is modeled off a hardware testbed at the MIT Space Science Laboratory (MIT-SSL) called ARGOS (Adaptive Reconnaissance Golay-3 Optical Satellite), a structurally connected interferometer. There are four types of modules: sub-apertures, beam combiners, structural elements, and the spacecraft bus. Each sub-aperture module contains a sub-aperture mirror, fast steering mirror, and (possibly) a reaction wheel assembly (RWA). The beam combiner module contains

the beam combiner optics. The structural element module and the spacecraft bus will vary based on the application. For ARGOS, the structural element consists of rings that connect each sub-aperture to the beam combiner. The spacecraft bus contains the electronics and communication equipment necessary for the interferometer. The bus could be a centralized single package or distributed around the satellite. The UDPs were used as the preliminary docking port design, with the addition of the SWARM node to provide the structure to allow multiple docking ports on a single payload.

4.5.2 Identification of System Trades

Identifying the system-level trades associated with spacecraft modularity was an important aspect of this study. Rendezvous-docking trades fall into two categories: telescope and module. Each trade is categorized into the group that it most impacts in terms of design modifications. For telescope design, some trade studies identified are: module segmentation, number of interfaces, and location of interfaces. The module segmentation, in terms of purpose (failure vs upgrade) and number (many small modules vs few large modules), can have an impact of the method for servicing, how often servicing occurs, as well as the cost of the servicing mission. Similarly, the number and location of the interfaces impact the ease of servicing. For example, if there are extra empty docking ports, one method of servicing could be to dock the new payload prior to release of the old payload. However, if location is a premium, the procedure for servicing may chance based on where the tug has to dock to pick up or drop off its payload. For module design, the design of the interface itself is critical to the ease of servicing. The overhead mass must not be too high, but sufficient to dock under most circumstances. The type of method for communicating the mass properties should also be considered, since the level of control after docking is important.

In order to evaluate these system trades, a set of quantifiable metrics have been identified by which to compare architectures. The main factor is the design of the servicing mission. Characteristics such as total mass, payload mass, number of servicing

missions, propellant used during the mission, and total servicing mission length are important to consider. Also, the number of interfaces is an important characteristic for measuring the complexity. Also, the impact to the telescope can be quantified by the total overhead mass added, compared to the total mass of telescope. In terms of the scientific added benefit, the percentage of scientific lifetime increased can be captured by the increased discovery efficiency or added lifetime in terms of mechanical parts. The performance of the servicing mission can be measured by the optical performance of the telescope prior to and after the servicing mission occurs. The error after servicing would include errors incurred from docking and mitigated by calibration.

4.5.3 Modularity Simulation

In order to provide a framework to assess the impact of modularity on servicing, a simulation was created in order to determine the effects. The main parameters that are inputs are: mission lifetime, type of interferometer, reason for servicing, and number of apertures. The outputs or metrics for the simulation are a subset of the metrics listed in the previous section. The initial metrics that have been modeled are: number of servicing missions, the size of the servicing mission, and the number of interfaces. The purpose of the simulation is to provide the initial modeling of a space telescope to determine how changing the model and characteristics of the module design impact the servicing mission.

Topology Map

The objective of the *Topology Map* function is to determine the highest level of modularity imposed by the type of interferometer. For a structurally connected interferometer, the entire spacecraft can be integrated or modularized. Each physical component is connected to the other. For a formation flown interferometer, the combiner and collector spacecraft are physically located kilometers away, attached only by 'soft' interfaces, such as communication. For a structurally connected interfer-

ometer, the highest level of modularity is for the spacecraft to be fully integrated. For a formation flown interferometer, there are three levels: fully integrated (if one part dies, must replace entire spacecraft), Combiner+collector (if any collector fails, replace all collectors), or each combiner and collector as a separate module. The topology map physically draws boundaries on where components can be placed. The high level modules are broken down further, later on in the simulation, to assess designed aspects of modularity, instead of inherent modularity. The high level module that is carried on is fully integrated for structurally connected interferometer (SCI) and one module per collector for formation flown interferometer (FFI).

Assign Components

The objective of the *Assign Components* function physically assigns components to the high level modules. For example, in a FFI, there are two types of modules: Combiner and Collector. The types of components in each module are identical, but the Combiner module has bigger solar arrays and more batteries. Also, the optics are different between the two. The *Assign Components* function captures the differences between the modules and highlights them when assigning the type and number of components to the modules.

Compute Modules

The *Compute Modules* function actually draws the boundaries of the modules that will be analyzed. The high level modules are broken down further first by purpose. The reason for servicing is taken to account when grouping modules. For example, if the main reason is to upgrade failed components, then modules are grouped by failure rate. Thus, when a module is replaced, the components with it that are close to failing are also replaced. If the reason for servicing is upgrade, the components are arranged by Moore's law, or some similar heuristic of when components would need to be upgraded to maximize performance.

The second parameter in the *Compute Modules* function is the level of modularity. To assess the impact of how much modularity is optimal, the level parameter is a

percentage modularity of the system. On a scale from 0 to 1, where 1 is 100% modular and there is one module for each component, to 0 where the telescope is fully integrated.

Time Simulation

After the modules are calculated, and each module has components assigned to it, a time simulation is run for the duration of the mission lifetime. The simulation time increment is two weeks. At every iteration, it initiates a random failure. If the random value is less than the probability of the part working at that time, then the part has failed. The failure rate data is taken from Hubble's components. The simulation assesses what modules have failed and sizes a servicing mission to repair them. The sizing currently is a proportional counter based on how many modules are present in the servicing mission.

4.5.4 Results

The simulation was run for the case of grouping based on failure rate data. Failure rate data was taken from an Aerospace Corporation document on the Hubble Space Telescope component failure rates. This data was condensed into a set of 17 components, selected to reflect the primary categories of components on a generic telescope [34]:

- Magnetometer
- Reaction Wheel Assembly
- Magnetic Torquer
- Star Tracker
- Rate Gyro
- Gimbal
- Solar Array

- Solid State Recorder
- Batteries
- Flight Computer
- Fine Guidance Sensor
- Communications Antenna
- Structures
- Instruments
- Primary Mirror
- Secondary Mirror
- Corrective Optics

The results shown in Figure 4-16 were run for the following input parameters: type of interferometer (SCI, FFI), mission lifetime (one value of 10 years), number of apertures (2-20), level of modularity (0:0.1:1), modularity groupings (only one grouping based on failure rate). Generally, though 100% modularity leads to more servicing missions, it is not an obvious linear increase. The variations are more pronounced in the formation flown case than in the structurally connected case. There also does not seem to be a strong correlation between the number of servicing missions and the number of apertures. Further improvements to the simulation include adding in a time delay for when the servicing actually occurs and more rigorous modeling on the impact of servicing multiple modules in a given mission [2].

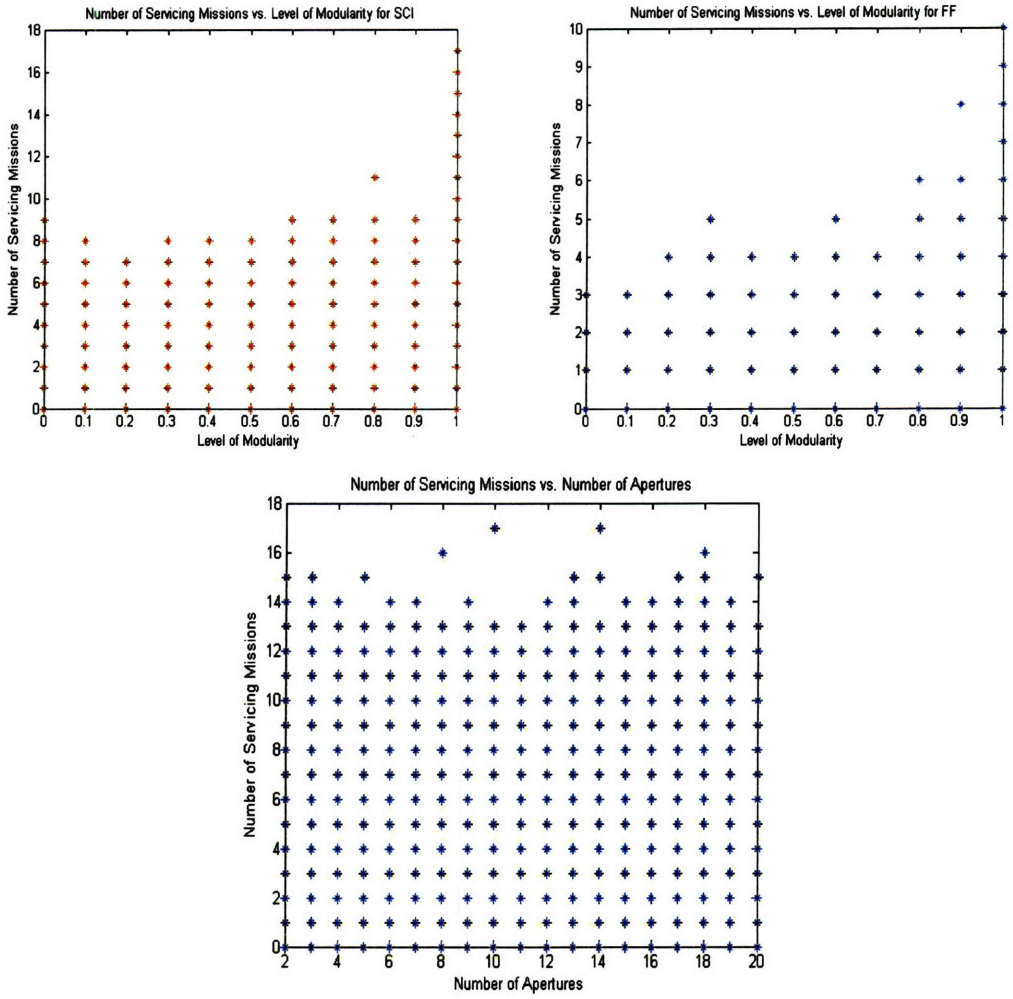


Figure 4-16: Modularity Simulation Results

Chapter 5

Conclusion

5.1 Thesis Summary

This thesis presented a methodology for implementing reconfiguration, with various applications and impacts to on-orbit servicing, assembly, and operations. The topics addressed in each chapter are summarized below.

Chapter One describes the motivation for implementing reconfiguration, and the overall benefits for space telescopes in particular. Chapter One also provides a background on the importance of modularity and the current status of research in the field for autonomous servicing and assembly.

Chapter Two presented an overview of the SPHERES testbed. In particular, this chapter calls out hardware aspects of the testbed that were particularly used or modified in this work. Of particular importance are the thruster geometry, the mixer that converts control inputs to firing durations, and the Universal Docking Port expansion port item.

Chapter Three describes the methodology for implementing reconfiguration on SPHERES. This chapter describes the steps for defining a configuration, calculating a configuration, and implementing on SPHERES. The implementation was tested through a set of simple maneuvers, whose results were shown. These results were compared to simulation, to test the accuracy of implementation. Also, they were compared to each other for two different configurations (battery proof mass versus a

SPHERE proof mass) to assess the control impact.

Chapter Four details the applications for which reconfiguration is used. Two hardware applications (SWARM and SIFFT) and two simulation applications (Mass Property Update and Modularity Analysis) are considered. The SWARM objective was to demonstrate autonomous assembly using SPHERES as a tug. Results from MSFC Flight Robotics Laboratory showed proper implementation but not execution, due to testing conditions. The objective of SIFFT was to demonstrate formation reconfiguration. Results showed good control to the centimeter level, with approximately five centimeter steady state error after reconfiguration. The objective of the Mass Property Update study was to determine if the method of obtaining the mass property values for reconfiguration has an impact on operations. Six options were considered using three methods. Simulation results tentatively indicated that there is a slight impact, mostly due to mass and reliability differences between different options. Finally, the objective of the Modularity Analysis was to develop a framework for assessing the impact of modularity on on-orbit servicing. This framework was developed and initial tests run for two parameters: number of modules and type of interferometer.

Chapter Five, this chapter, summarizes the thesis, as well as provides some conclusions. This chapter also details future work in each of the four applications mentioned in Chapter Four and in the implementation on SPHERES (Chapter Two).

5.2 Conclusions

This work was able to demonstrate that there is a control impact associated with the implementation of reconfiguration. Reconfiguration was implemented on the SPHERES testbed to provide a method for analyzing the control performance under different configurations. In general, reconfiguration provides significant improvement in control performance, provided that the payload is of comparable mass to the satellite. The amount of improvement, as a function of the payload size, is an area that should be explored further.

Each of the four applications demonstrates a different reason why reconfiguration is important. The range of applications allows for a broad understanding of how different applications influence the requirements on reconfiguration. Hardware applications, such as SWARM and SIFFT, demonstrated the control performance requirements of reconfiguration. The satellites must retain their maneuverability in the new configuration, in order to be effective. Also, the transitions between configuration must be smooth, so as to minimize downtime and perturbations in the system. For simulation applications, such as Mass Property Update and Modularity Analysis, it was demonstrated that the method of implementation of reconfiguration could have an impact on mission operations and design. Conclusions for the Mass Property Update simulation were that the choice of the method of obtaining mass property values for reconfiguration has an impact on the mass and overall performance of the assembly mission. For the Modularity Analysis, it was important to determine if the level of modularity has an impact based on the mission objective. The Modularity Analysis allowed the determination of methods for analyzing the impact of modularity on different mission scenarios.

In conclusion, this work allowed the assessment of the impact of reconfiguration, both in a direct implementation (as on SPHERES), and indirectly through performance analysis of applications.

5.3 Future Work

This work forms a good foundation for expansion. Future work is described below in terms on the implementation on SPHERES, and the four applications detailed in Chapter Four.

Future work for implementation of reconfiguration on SPHERES tests are:

- Add sensor position: Incorporate changing sensor configuration into reconfiguration. This includes updating the sensor locations and updating how the estimator reads in sensor measurements.

- Demonstrate position control: Preliminary work has already begun on implementing position control. Results have demonstrated position control to within 10cm. New gains must be calculated to ensure that position control to sub-centimeter level is achieved after reconfiguration.

- Online system calculation: Incorporate autonomy into SPHERES by calculating configurations online. Only the element properties would exist in memory, instead of pre-set configurations. This would expand the range of configurations possible during a single mission. Some issues arise in memory and computation on-board the SPHERES.

Future work on the SWARM and SIFFT applications are:

- Demonstrate reconfiguration on flexible structures: SPHERES are easily modeled as rigid body systems. It would be interesting to demonstrate reconfiguration of flexible structures, through docking, with minimal excitation of vibration.

- Full assembly sequence test. Reconfiguration and control after docking have been shown, but not the complete sequence of dock-reconfigure-move. This can be expanded to include multiple dockings, where the satellite must demonstrate control after undocking.

- Incorporate path planning and obstacle avoidance into maneuvering sequences

Future work on the simulations are:

- Incorporate cost into modularity simulation

- Add in assembly or servicing dynamics to get estimates of mission durations

- Incorporate time simulation for launching and executing servicing missions

- Perform trade studies on how modularity affects design of servicing missions

Bibliography

[1] Astronautix. Shuttle orbiter. http://www.astronautix.com/stages/shubiter.htm.

[2] Mark Baldeserra. A decision-making framework to determine the value of on-orbit servicing compared to replacement of space telescopes. Master's thesis, MIT, 2005.

[3] Caltech. Spitzer space telescope. www.spitzer.caltech.edu.

[4] Allen Chen. Propulsion system characterization for the spheres formation flight and docking testbed. Master's thesis, MIT, 2002.

[5] Lauren Clark. Mit's intelligent aircrafy fly, cooperate autonomously. *MIT News Office*, 2006.

[6] John Enright, Mark Hilstad, Alvar Saenz-Otero, and David W Miller. The spheres guest scientist program: Collaborative science on the iss. In *IEEE Aerospace Conference*. IEEE, March 2004.

[7] GSFC. Micro-arcsecond x-ray imaging mission. http://maxim.gsfc.nasa.gov/.

[8] GSFC. Si: The stellar imager. http://hires.gsfc.nasa.gov/si/.

[9] Harvard. Chandra x-ray observatory center. chandra.harvard.edu.

[10] RA Hess and C McLean. Development of a design methodology for reconfigurable flight control systems. In *38th Aerospace Sciences Meeting & Exhibit*. AIAA, January 10-13 January 2000.

[11] Mark Hilstad. A multi-vehicle testbed and interface framework for the development and verication of separated spacecraft control algorithms. Master's thesis, MIT, 2002.

[12] Nicholas Hoff. Design and implementation of a relative state estimator for docking and formation control of modular autonomous spacecraft. Master's thesis, MIT, 2007.

[13] Nicholas Hoff, Swati Mohan, Simon Nolet, and David W Miller. Docking and reconfiguration of modular spacecraft preliminary swarm testing at msfc. In *SPIE Defense and Security Symposium*, 2007.

[14] Space Telescope Science Institute. Hubblesite. http://hubblesite.org/.

[15] Carole Joppin and Daniel Hastings. On-orbit upgrade and repair: The hubble space telescope example. *Journal of Spacecraft and Rockets*, 43(3), May-June 2006.

[16] Lt Col Fred Kennedy. Orbital express space operations architecture. http://www.darpa.mil/tto/programs/oe.htm.

[17] NASA Jet Propulsion Laboratory. Planet quest: Missions - terrestrial planet finder. http://planetquest.jpl.nasa.gov/TPF/tpf-index.cfm.

[18] Elisabeth Lamassoure, Josephh H Saleh, and Daniel Hastings. Space systems flexibility provided by on-orbit servicing: Part 2. *Journal of Spacecraft and Rockets*, 39, 2002.

[19] Gregg Leisman. Analysis of on-orbit servicing architectures using microsatellites, advanced propulsion, secondary payload opportunities and the military spaceplane concept. *AIAA*, 2000.

[20] Attila Lengyel. Design of a small spacecraft to perform on-orbit servicing tasks. 2001.

[21] NASA. Main hubble page. http://hubble.nasa.gov/.

[22] NASA. Solar and heliospheric observatory homepage. http://sohowww.nascom.nasa.gov/.

[23] Simon Nolet. *Development of a guidance, navigation and control architecture and validation process enabling autonomous docking to a tumbling satellite.* PhD thesis, MIT, 2007.

[24] The Encyclopedia of Astrobiology Astronomy and Spaceflight. Terrestrial planet imager. http://www.daviddarling.info/encyclopedia/P/PlanetImager.html.

[25] Dr. Charles M. Reynerson. Spacecraft modular architecture design for on-orbit servicing. In *AIAA Space Technology Conference & Exposition 29-30 Sept. 1999 Albuquerque, NM.* AIAA, 1999.

[26] Dr. Charles M. Reynerson. Spacecraft servicing - first order model for feasibility and cost effectiveness. In *AIAA Space 2001 Conference*, 2001.

[27] Lennon Rodgers. Concepts and technology development for the autonomous assembly and reconfiguration of modular space systems. Master's thesis, MIT, 2005.

[28] Alvar Saenz-Otero. *Design Principles for the Development of Space Technology Maturation Laboratories Aboard the International Space Station.* PhD thesis, MIT, 2005.

[29] Alvar Saenz-Otero and David W. Miller. Spheres:a platform for formation flight research. In *UV/Optical/IR Space Telescopes: Innovative Technologies and Concepts II conference.* SPIE, August 2005.

[30] Joseph H. Saleh, Elisabeth Lamassoure, and Daniel E. Hastings. Space systms flexibility provided by on-orbit servicing: Part 1. In *AIAA Space 2001 Conference*, 2001.

[31] Space and Tech. Expendable launch vehicles. http://www.spaceandtech.com/spacedata/elvs/elvs.shtml.

[32] Andrew E. Turner. Cost-effective spacecraft dependent upon frequent non-intrusive servicing. *AIAA Space 2001 Conference*, August 2001.

[33] PKC Wang and FY Hadaegh. Optimal formation reconfiguration for multiple spacecraft. *AIAA*, pages 686–696, 1998.

[34] Helen Wong. Hubble space telescope reliability assessment, july 2002 model. Technical report, Aerospace Corporation, 2002.