Development of a Guidance, Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite

by

Simon Nolet

B.Ing., Université de Sherbrooke (1998)
M.Eng., Massachusetts Institute of Technology (2001)

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

Doctor of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2007

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

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

Certified by .....
Jonathan P. How
Associate Professor of Aeronautics and Astronautics

Certified by .....
John J. Deyst Jr.
Professor of Aeronautics and Astronautics

Certified by .....
Edmund M.C. Kong
Senior Principal Engineer, Orbital Sciences Corporation

Accepted by .....
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Development of a Guidance, Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite

by

Simon Nolet

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

Abstract

The capability to routinely perform autonomous docking is a key enabling technology for future space exploration, as well as assembly and servicing missions for spacecraft and commercial satellites. Particularly, in more challenging situations where the target spacecraft or satellite is tumbling, algorithms and strategies must be implemented to ensure the safety of both docking entities in the event of anomalies. However, difficulties encountered in past docking missions conducted with expensive satellites on orbit have indicated a lack of maturity in the technologies required for such operations. Therefore, more experimentation must be performed to improve the current autonomous docking capabilities.

The main objectives of the research presented in this thesis are to develop a guidance, navigation and control (GN&C) architecture that enables the safe and fuel-efficient docking with a free tumbling target in the presence of obstacles and anomalies, and to develop the software tools and verification processes necessary in order to successfully demonstrate the GN&C architecture in a relevant environment.

The GN&C architecture was developed by integrating a spectrum of GN&C algorithms including estimation, control, path planning, and failure detection, isolation and recovery algorithms. The algorithms were implemented in GN&C software modules for real-time experimentation using the Synchronized Position Hold Engage and Reorient Experimental Satellite (SPHERES) facility that was created by the MIT Space Systems Laboratory. Operated inside the International Space Station (ISS), SPHERES allow the incremental maturation of formation flight and autonomous docking algorithms in a risk-tolerant, microgravity environment.

Multiple autonomous docking operations have been performed in the ISS to validate the GN&C architecture. These experiments led to the first autonomous docking with a tumbling target ever achieved in microgravity. Furthermore, the author also demonstrated successful docking in spite of the presence of measurement errors that were detected and rejected by an online fault detection algorithm.

The results of these experiments will be discussed in this thesis. Finally, based on experiments in a laboratory environment, the author establishes two processes for the verification of GN&C software prior to on-orbit testing on the SPHERES testbed.

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

Acknowledgments

This piece of work would not have been possible without the help of a lot of people around me. Over the 5 1/2 years spanned by this research, I had the chance to work with incredible people. This is where I want to thank them for all their help. I apologize in advance to whoever I might have forgotten in this list.

I first want to thank my committee, Prof. David Miller, Prof. Jonathan How, Prof. John Deyst, and Dr. Edmund Kong. They were the ones who kept me on track throughout this research. They supported me all the way through, and provided me with great ideas when it was needed the most. Prof. David Miller should be especially acknowledged for having given me the opportunity to work on the SPHERES project. I owe him tremendously. He was also very patient and helpful at every step of my doctorate, including when writing this thesis. Moreover, he is the one who kept the SPHERES team alive and led it to the ISS, despite continuous delays in the program, following the Columbia accident. Dr. Edmund Kong should be also particularly acknowledged for giving me day-to-day support for the first four years of my research. He was a constant source of ideas for research avenues and a great team leader.

Thanks also to my readers, Prof. Olivier de Weck and Dr. Ray Sedwick, for their help not only when writing my thesis, but also at different stages of my doctorate program, whether it was help in preparing for the Qualification or the General Exams.

A lot of my learning comes from the tremendous people I worked with on a day-to-day basis. I want to thank here the whole SPHERES team, who changed a lot over the years I was part of it (Alvar Saenz-Otero, Mark Hilstad, Allen Chen, John Enright, Dustin Berkovitz, Swati Mohan, Nick Hoff, Chris Mandy, Amer Fejzic, Soon-Jo Chung, Danielle Adams, and all the UROPs and other students who worked part-time with us). SPHERES is a team effort and everybody fulfilled their role, which facilitated my life a lot. In particular, I want to thank Dr. Alvar Saenz-Otero, who deserves enormous credit not only for his help with everything regarding the SPHERES facility, but also for his support in this research. The work presented in this thesis relied on the SPHERES facility, and Alvar is in my mind the one who

contributed the most to its design. I also want to acknowledge the students I worked with from the ACL, especially Arthur Richards (whose aura of intelligence still hovers around MIT), Louis Breger and Georges Aoude. Other guest scientists gave me great ideas throughout this research, including Dr. Edward Wilson from Intellization.

Another group of people that deserves particular credit is the whole team at Payload Systems, Inc. In particular, I want to thank Steve Sell, Stephanie Chen, Javier de Luis, Joe Parrish, Edison Guerra, John Merk and Joanne Vining. They are the ones who put SPHERES together. My research heavily relied on two things: a reliable testbed, and a testbed on ISS. They provided me with that.

I also want to thank the people from the different NASA and DoD centers for their support when performing experiments. In particular, thanks to Linda Brewster, Richard Howard and their team from NASA MSFC, John Yaniec and his team from the JSC Reduced Gravity Program, and, finally, Mark Adams, Major Matthew Budde and their team from the DoD STP office.

The staff in the MIT SSL has also been very supportive to me over the years. Thanks to Paul Bauer, who always had answer to ANY question I asked him, and who was enormously helpful when working with hardware. Thanks also to Marilyn Good, Sharon Leah Brown and Kathryn Fischer (from the ACL) for their help and kindness with office related activities.

Merci aussi au Fonds québécois de la recherche sur la nature et les technologies (anciennement FCAR) pour leur soutien financier tout au long de mes études graduées au MIT.

I now want to thank the dearest persons in my life, my whole family, whom I owe the most for their love and infinite support, including countless sacrifices they made for me. Without them, I would not have made it. Plus particulièrement, merci à mon frère Étienne et à sa conjointe Catherine pour tous les encouragements. Merci, maman et papa, pour vous avoir tant sacrifiés pour moi, et pour toutes les fois où vous êtes venus pour vous occuper de Claire. Merci aussi à tous ceux qui sont passés avant moi, incluant mes grands-parents (spécialement à mon grand-père Cogné qui m'a donné le goût pour les sciences lorsque j'étais enfant), et mon parrain Raymond.

Merci aussi à Claire, dont la naissance a été le moment le plus heureux de mon doctorat. Claire, j'ai vraiment hâte de pouvoir te lire ces quelques lignes dans les années qui vont venir. Là, j'espère juste pouvoir me rattraper pour le temps que je n'ai malheureusement pas pu passer avec toi comme tu l'aurais tant mérité. Tu es vraiment l'enfant idéale qu'un parent puisse espérer. But above all, the person who was the most supportive is my wife Xue'en. Merci, Xue'en, tu es sans doute la raison même pour laquelle je suis encore là aujourd'hui. Sans toi, je n'y serais pas arrivé. Je t'aime!

Finally, I want to dedicate this thesis to a special person for me, a person who has been fighting for her life for almost two years now. This person is my mother-in-law. When I torn ligaments in my knee while playing softball with the lab's team, she came over all the way from San Francisco and help me out for a whole month. When Claire was born, she came again, this time for four months, to give us a hand, such that I can pass my General Exam that summer. A year later, only two weeks after Xue'en defended her thesis, we learned that she had brain cancer, terminal phase. Her fight was a constant inspiration for me, and I am so happy that, against all odds, she is still there such that one week from now, I will be able to tell her in person that I made it.

Mommy, this one is for you...

Contents

List of Figures

List of Tables

List of Acronyms

2-D	Two dimensions
3-D	Three dimensions
A/D	Analog to digital signal converter
ACL	Aerospace Controls Laboratory
AVGS	Advanced Video Guidance Sensor
CAM	Collision avoidance maneuver
CEV	Crew Exploration Vehicle
CG	Center of gravity
DARPA	Defense Advanced Research Projects Agency
DART	Demonstration of Autonomous Rendezvous Technology
DoD	Department of Defense
DOF	Degree-of-freedom
EKF	Extended Kalman filter
ESA	European Space Agency
ETS-VII	Engineering Test Satellite VII
FD	Failure detection
FDI	Failure detection and isolation
FDIR	Failure detection, isolation and recovery
FPGA	Field Programmable Gate Array
GN&C	Guidance, navigation and control
GPS	Global Positioning System
GSP	Guest Scientist Program
GUI	Graphical user interface

INS	Inertial navigation sensors
IR	Infrared
ISS	International Space Station
LP	Linear program
LQR	Linear-quadratic regulator
KF	Kalman filter
MILP	Mixed-integer linear programming
MIT	Massachusetts Institute of Technology
MoIRs	Moment of inertia ratios
MOSR	Mars Orbiting Sample Retrieval experiment
MPC	Model predictive control
MSFC	Marshall Space Flight Center
MVM	Mission and vehicle management
NASA	National Aeronautics and Space Administration
NASDA	National Space Development Agency
NRL	National Research Laboratory
OS	Operating system
PD	Proportional-derivative
PDF	Probability density function
PF	Particle filter
PID	Proportional-integral-derivative
RF	Radio frequency
RSO	Resident space object
RV	Rendezvous
RVSF	Rendezvous flight software
S/N	Serial number
SPHERES	Synchronized Position Hold Engage and Reorient Experimental Satellites
SSL	Space Systems Laboratory
SWARM	Synchronized Wireless Autonomous Reconfigurable Modules
TOF	Time-of-flight

TRL	Technology Readiness Levels
UKF	Unscented Kalman filter
U/S	Ultrasonic
U.S.	United States
V&V	Verification and validation
XSS	Experimental Satellite System
μ-g	Microgravity
1-g	Earth's gravity

Nomenclature

a	[scalar]	Slope in the phase plane of the desired velocity profile used by the glideslope algorithm
A	[scalar]	Area in meters^2
$\mathbf{A}$	$[l \times l]$	Jacobian matrix of $\mathbf{f}()$
$\mathbf{b}$	[3×1]	Vector of gyroscope biases in radians/second
B	[scalar]	Blowdown multiplier affecting all the thrusters caused by a sudden pressure drop
$\mathbf{B}$	$[l \times N]$	Control effect matrix
c	[scalar]	Speed of sound in meters/second
$\mathbf{c}$	$[p_k \times 1]$	Vector of bounds
C	[scalar]	SPHERES tank capacity in kilograms
$\mathbf{C}$	[6×1]	Vector of commanded forces and torques
$\mathbf{d}$	[3×1]	Vector of displacements in meters
D	[scalar]	Duty cycle in %
$\mathbf{D}$	$[3 \times N]$	Matrix of unit vectors indicating the thrust direction in the body frame
$\mathbf{E}[]$		Matrix of expectation functions
$\mathbf{f}()$	$[l \times 1]$	Vector of nonlinear functions expressing the dynamics of the system
$\mathbf{F}$	[3×1]	Vector of commanded forces in the body frame in newtons
$\mathcal{F}$	$[N \times 1]$	Vector of thruster strength for each ON/OFF thruster in newtons

g	[scalar]	Earth's gravitational constant (9.81 meters/second^2)
$\mathbf{G}$	$[p_k \times l]$	Matrix of constraint functions on the states
$\mathbf{h}$	$[n_k \times 1]$	Vector of expected measurements given the states in meters
$\mathbf{H}$	$[n_k \times l]$	Jacobian matrix of $\mathbf{h}$
I	[scalar]	Inertia in kilograms.meters^2
$\mathbf{I}$	[3×3]	Inertia tensor (matrix) in kilograms.meters^2
$\mathcal{I}_{sp}$	[scalar]	Specific impulse in seconds
$\mathcal{I}$		Identity matrix
J	[scalar]	Cost to be optimized
K_d	[scalar]	Derivative gain
K_i	[scalar]	Integral gain
K_p	[scalar]	Proportional gain
$\mathbf{K}$	$[l \times n_k]$	Kalman gain matrix
l	[scalar]	Length of the state vector
$\mathbf{L}$	[3×N]	Matrix containing the x-y-z location in the body frame of each thruster in meters
m	[scalar]	Mass of the satellite in kilograms
M	[scalar]	Very large number as compared with the elements of the state vector
$\mathcal{M}$	$[N \times 6]$	Thrust mapping matrix
n	[scalar]	Number of measurements collected
$\mathbf{n}$	[3×1]	Unit vector defining the direction of the Euler axis for a rigid body rotation
N	[scalar]	Number of thrusters
$\mathbf{N}[]$		Vector of normally distributed functions
p	[scalar]	Number of constraint functions
$\mathbf{p}()$		Vector of probability density functions
P	[scalar]	Pressure in pascals (unless specified otherwise)

$\mathbf{P}$	$[l \times l]$	Covariance matrix
$\mathcal{P}$	[scalar]	Control period in seconds
$\mathbf{q}$	[4×1]	Attitude component of the state vector (quaternions)
$\mathbf{Q}$	$[l \times l]$	Process noise covariance matrix
$\mathcal{Q}$	$[l \times l]$	State weighting matrix
$\mathbf{r}$	[3×1]	Position component of the state vector in meters
R	[scalar]	Ideal gas constant joules/(kilogram·kelvin)
$\mathbf{R}$	$[n_k \times n_k]$	Measurement noise covariance matrix
$\mathcal{R}$	$[N \times N]$	Control input weighting matrix
$sgn()$	[scalar]	Signum function (returns 1, 0 or -1 depending on the sign of the input)
S	[scalar]	Number of steps in the problem
$\mathbf{S}$	$[l \times (l-1)]$	Matrix used to transition from the reduced form to the expanded form
t	[scalar]	Time in seconds
T	[scalar]	Temperature in kelvins
$\mathcal{T}$	[scalar]	Total period used by the glideslope algorithm in seconds
$\mathbf{T}$	[3×1]	Vector of commanded torques in the body frame in newtons.meters
u	[scalar]	Velocity in meters/second
$\mathbf{u}$	[N×1]	Control input vector
$\mathbf{v}$	[3×1]	Velocity component of the state vector in meters/second
ΔV	[scalar]	A change in velocity in meters/second
W_i^{mean}	[scalar]	Weighting parameter associated with the sigma-point vector i (UKF)
W_i^{cov}	[scalar]	Weighting parameter associated with the sigma-point vector i (UKF)

$\mathcal{W}$	$[\Pi \times 1]$	Vector or particle weight (PF)
$\mathbf{x}$	$[l \times 1]$	State vector
$\mathbf{X}$	$[l \times \Pi]$	Matrix where each column represents a sample (particle) of a PDF (PF)
$\mathbf{y}$	$[p_k \times 1]$	Vector of binary variables
Y	[scalar]	Discrete state variables for the hybrid PF representing the current mode
z	[scalar]	Calibrated measurement in meters
$\mathbf{z}$	$[n_k \times 1]$	Vector of calibrated measurements in meters
$\mathcal{Z}$		Matrix containing the measurement history up to time t
α	[scalar]	Adiabatic index
γ	[scalar]	Design parameter for the UKF
$\boldsymbol{\gamma}$	$[(2l+1) \times 1]$	Vector of parameters necessary to reconstruct the covariance matrix (UKF)
Γ	[scalar]	The magnitude of the position component of the state vector in meters
δ_y	[scalar]	The magnitude of the projection of the position vector along the y-body axis of the target in meters
δ_z	[scalar]	The magnitude of the projection of the position vector along the z-body axis of the target in meters
ϵ	[scalar]	Some tolerance close to zero
$\boldsymbol{\varepsilon}$	$[l \times 1]$	Vector of independent zero-mean white noise processes with covariance $\mathbf{Q}$
ζ	[scalar]	Damping ratio of the complex poles for a second order system
θ_e	[scalar]	Attitude angular error around the Euler axis in radians
θ_{pitch}	[scalar]	Pitch bearing angle in radians
θ_{yaw}	[scalar]	Yaw bearing angle in radians

$\boldsymbol{\theta}$	[3×3]	Attitude matrix changing a representation of vectors to the body frame
ι	[scalar]	Sum of the EKF innovations for a given set of measurements in meters
$\boldsymbol{\Lambda}$	[3×3]	Intermediate matrix used when expressing the reduced state transition matrix
$\boldsymbol{\mu}$	[3×1]	Vector of biased gyroscope measurements in radians/seconds
$\boldsymbol{\nu}$	[n_k×1]	Vector of independent zero-mean white noise processes with covariance $\mathbf{R}$
$\boldsymbol{\Xi}$	[4×3]	Matrix used to transition between the reduced and the expanded form
ρ	[scalar]	Separation (face-to-face) distance between two satellites in meters
$\boldsymbol{\rho}^{frame}$	[3×1]	Vector of U/S receiver location in the frame specified by the superscript
$\boldsymbol{\varrho}^{frame}$	[3×1]	Vector of U/S transmitter location in the frame specified by the superscript
Π	[scalar]	Number of particles (PF)
$\boldsymbol{\sigma}$	[l×2l]	Matrix formed from vectors of parameters used to compute the sigma-points (UKF)
ς	[scalar]	Raw measurement in meters
τ	[scalar]	Time constant associated with the integral control of a second order system
$\boldsymbol{\tau}$	[n_k×1]	Vector of time-of-flight measurements collected in seconds
$\boldsymbol{\upsilon}$	[n_k×1]	Vector of innovations in meters
$\boldsymbol{\Phi}$	[$l \times l$]	State transition matrix
φ	[scalar]	Density in kilograms/meter^3
$\boldsymbol{\chi}$	[l×(2l+1)]	Matrix formed from sigma-point vectors (UKF)

ω_n	[scalar]	Natural frequency of the complex poles for a second order system
ω_{pitch}	[scalar]	Angular rate of the line-of-sight to the beacon along the pitch axis in radians/seconds
ω_{yaw}	[scalar]	Angular rate of the line-of-sight to the beacon along the yaw axis in radians/seconds
$\boldsymbol{\omega}$	[3×1]	Angular rate component of the state vector in radians/second
$\boldsymbol{\Omega}$	[4×4]	Matrix used to relate the quaternion vector to its derivative
$\boldsymbol{\aleph}$	[3×3]	Intermediate matrix used when expressing the reduced state transition matrix

Superscripts:

$^{(-)}$	Value immediately prior to the update
$^{(+)}$	Value immediately after the update
$\hat{}$	An estimated value
$\tilde{}$	Reduced form (dimension $(l-1)$ instead of l)
$\dot{}$	First derivative
^{T}	Matrix transpose

Subscripts:

i	Indicates a particular measurement
j	Indicates a particular component of a vector
k	Relates to a particular time t_k
c	Relates to the chaser satellite
t	Relates to the target satellite

Coordinate frames:

body	Satellite's body coordinate frame
ISS	Coordinate frame attached to the ISS (global frame)

Chapter 1

Introduction

Autonomous docking permits novel space capabilities like satellite servicing and the Mars Sample Return Mission. Although these capabilities are envisioned to be available in the near future, the technologies required to perform safe autonomous docking need to be matured. Much research has been performed on algorithms tackling parts of the solution, but only a few organizations worldwide were ever able to accomplish an autonomous docking in space. Moreover, autonomous docking with a tumbling target has never been attempted on orbit, prior to this research, because of the risk involved in such a challenging mission. Therefore, there is a need to start building a legacy and to gain experience.

This thesis presents an approach that led to advanced autonomous docking, including with a tumbling target.[1] It covers, from the systems engineer point of view, the design process from the implementation of the guidance, navigation and control (GN&C) algorithms to the verification of the GN&C software prior to flight and the validation through on-orbit experiments. The outcome of this research is a series of validated GN&C software tools and processes, that have been tested both in simulation and with prototype hardware, for the on-orbit formation flight and autonomous docking strategies.

[1] A target which lost control authority around at least one of its body axes.

1.1 Motivation

In the U.S. space program, the interest in autonomous docking periodically surfaces in two areas [97]. The first one is autonomous delivery of cargo to the International Space Station (ISS) for resupply or reboost. The Russians currently provide that capability through their Progress resupply vehicles. At an early stage of development of the ISS, there was a concern that the Russians would not deliver their segment of the ISS, which consisted of the Zarya Control Module and the Service Modules to periodically reboost the ISS. NASA, which did not have the capability to perform autonomous docking, initiated studies that generated requirements for a potential replacement of the Service Modules in case the Russians could not deliver them. Later on, during the development of the former VentureStar Reusable Launch Vehicle designed to ferry cargo to and from the ISS, the need to perform autonomous docking reappeared. It is likely to remain a requirement for any future vehicle designed to resupply and reboost the ISS.

The execution of complex manned and unmanned missions to Mars is the second area in the U.S. space program that requires autonomous docking capabilities. From the mid 1970s, studies on Mars surface sample return missions have shown that separation of the lander (that collects the samples) from the orbiter (that propels the sample canister in its journey back to the Earth) significantly reduces the size and mass of the required spacecraft [114]. Upon return, a docking maneuver has to be performed between the orbiter and the sample capsule taking off from Mars. This docking maneuver must occur in Mars orbit and needs to be performed autonomously because of the long transmission delay with Earth-based ground controllers. Concerning manned missions to Mars, a docking maneuver is also likely to occur in Mars orbit to reduce the mass of the spacecraft. Because of the long duration of such a mission, this maneuver must occur nearly two years after the crew first left the Earth. With the crew not being able to practice their rendezvous (RV) and capture techniques for a long time, relying on them to perform a critical part of the mission is unsafe, hence requiring an autonomous docking capability [97].

More recently, after the U.S. government defined a new vision for Space Exploration in the 21^{st} Century, an approach for a long term exploration program has been developed [135]. It involves completing the assembly of the ISS, gradually building the Crew Exploration Vehicle (CEV) and returning to the Moon in the next decade. This is believed to be a stepping stone to the future exploration of Mars. All of the proposed mission architectures require the capability to perform routine autonomous docking, capture and in-space assembly. Some of these missions might involve a tumbling target in situations where direct human intervention or supervision is impossible.

Autonomous docking also enables other solutions for space systems designers, like satellite inspection, refurbishing and refueling. Increased autonomy allows for greater survivability in the following situations:

- when failures occur;
- when other spacecraft are in close proximity;
- when the transmission delay is too long for ground control to react to anomalies within a safe time frame;
- when the communication bandwidth is too limited to transmit all the relevant information in real time to the ground controllers.

In close proximity operations, task sequencing and execution must be robust to unexpected events. These events can be internal (e.g., a component failure), or external (e.g., an obstacle in the desired flight path).

Space missions such as satellite servicing introduce challenges that were not met in previous missions involving docking maneuvers. Because of the very nature of servicing missions, the target spacecraft might not be fully cooperative in the presence of hardware failure or fuel depletion. Moreover, when performing routine autonomous docking maneuvers, chances are that there will be situations where anomalies will occur and that something will not go as planned. During close proximity operations, the safe time frame to react to anomalies is reduced, when the satellite is tumbling,

because of the possibility of collisions with rotating appendages like solar panels and antennas.

Many satellites are faced with situations where they have to stabilize tumbling motions. The deployment after orbit insertion is an example. Typical booster tip-off rates range somewhere between 1 deg/sec and 4 deg/sec (Falcon and Minotaur launch vehicles). A thruster stuck-OFF failure on the satellite, when it attempts to stabilize the tumble, can result in an uncontrollable motion.

All of the aforementioned reasons justify the need for the development of a safe and efficient strategy to routinely perform autonomous docking maneuvers with both cooperative and uncooperative targets. These strategies must rely on computers to plan an approach, execute the plan, monitor the trajectory by analyzing sensor data and make mission-critical decisions in the presence of anomalies to ensure the safety of both spacecraft.

Unfortunately, space missions involving autonomous docking rarely publicize in detail the various GN&C algorithms used at different stages of the mission, not to mention the detailed selection, implementation and validation processes. Often, the algorithms are selected based on the legacy of previous missions flown by design engineers involved in the development of the GN&C architecture. This results in a conservative approach that is justified largely by economical reasons.

However, the many new challenges behind autonomous docking with a tumbling target might not be safely handled by algorithms traditionally used in automated docking missions. For example, resorting to a traditional safe mode might be insufficient to ensure safety when anomalies occur. The close proximity of the two satellites, and the inherent risk of collision, require the chaser to follow a collision avoidance maneuver (CAM), unless passive safety was ensured in the planning of the trajectory. A systems engineer working on such a mission has to cope with these challenges and be able to identify, implement and integrate an appropriate suite of algorithms that fulfill all of the mission requirements.

1.2 Previous autonomous docking and inspection missions

This section is necessary to put the context of this research in perspective with past and current missions. It provides an historical review of space missions involving autonomous docking or close proximity inspection, with emphasis on the anomalies encountered.

The first successful docking maneuver in space was performed on March 16, 1966, when astronauts Armstrong and Scott docked their Gemini 8 capsule to an Agena Target Vehicle. This docking maneuver, like all other docking maneuvers the U.S. ever attempted until the Orbital Express mission, heavily relied on the crew to control their spacecraft in the last meters of the maneuver. Historically, this manual approach caused every docking strategy to be tailored to a specific mission. No attempts were made to standardize them. Consequently, extensive crew training and system redundancy were required to ensure mission success. This largely contributed to the increase in costs associated with each docking maneuver [97].

On the other hand, from the very beginning, the Russians pursued an approach to docking that was automated, with standardized operations. For manned missions, the crew was relegated to monitoring and manual backup functions. The first automated docking maneuver was performed on October 30, 1967, when the Soviet experimental unmanned spacecraft Cosmos-186 docked with Cosmos-188 (Fig. 1-1). With the very low computational capabilities available in these early days of space flight, the control algorithm used during the automatic docking maneuver had to be extremely simple [72, 100]. After RV, when the spacecraft were in close proximity, the docking maneuver consisted of an accelerating thruster pulse to initiate the approach and a braking pulse seconds before contact. During the approach, the incoming chaser had a tendency to follow a curved trajectory because of the slight difference in the Keplerian orbits of both satellites. To maintain a straight line approach toward the target, the chaser had to constantly perform fuel-expensive trajectory corrections. Although capture occurred on the second attempt (out of range of ground tracking stations),

Figure 1-1: An early Soviet stamp illustrating the first automatic docking performed by Cosmos-186 and Cosmos-188.

electric connections were not completed because of a misalignment at contact [123].

The same automated approach was used in manned missions to the Salyut space stations during the early 1970s. The ground controllers and the cosmonauts remained available to gain control in case of anomalies. A series of malfunctions with the automatic docking system, and the need for frequent resupply of the Salyut space stations, led the Soviets to develop the Progress resupply vehicle in the mid 1970s (Fig. 1-2). Progress is a modified version of the Soyuz ship and is unmanned. It was used (and still is) to ferry cargo to space stations and to dispose of trash when it disintegrates high in the Earth's atmosphere upon return. It is equipped with a slightly more advanced docking system than the one used in the early ages of space flight. A total of 43 Progresses were launched to the Salyut-6 and Salyut-7 space stations, and all successfully completed their missions [134, 48]. After the Salyut era, Progress was modernized and used initially for the MIR space station from the mid 1980s to the mid 1990s, and then for the ISS. As of early 2007, it is still the only unmanned vehicle to dock with the ISS.

More recently, a series of experimental missions were flown to test and demonstrate advanced autonomous docking techniques. The first one was the Engineering Test Satellite VII (ETS-VII, Fig. 1-3), a Japanese led experiment. It was designed

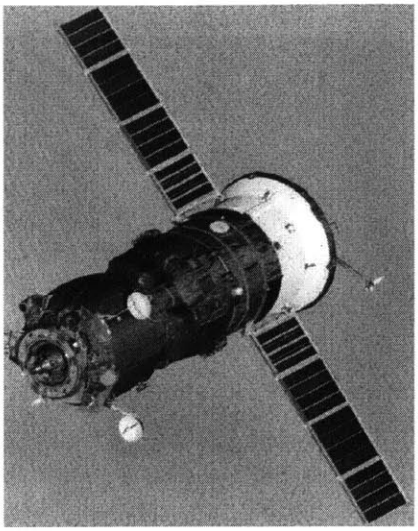


Figure 1-2: The Progress resupply vehicle.

to test and demonstrate the technologies that would be used on the H-II Transfer Vehicle, one of NASDA's contributions to the ISS [89, 65]. Two autonomous docking maneuvers were attempted. The first one started with the separation of the chaser. While maintaining attitude, the chaser moved to a holding point two meters away from the target. At that point, it maintained a constant separation of two meters, flying in formation with the target for 15 minutes. An approach command was then sent to the chaser. It initiated its approach at a speed of 1 cm/sec. Docking was successfully completed a couple of minutes later. The second experiment did not go as well. This time, after separation, the chaser was commanded to move to a holding point 525 meters away from the target. During separation, an attitude anomaly occurred causing the chaser to automatically execute a Disable Abort. It then flew to a retreat point 2.5 km away from the target. An approach maneuver was attempted twice, without success. Ground investigation found that the source of the problem was a failed thruster. A reconfiguration of the Rendezvous Flight Software (RVFS) was performed by the controllers. The docking was finally accomplished on the third attempt nearly three weeks after the first occurrence of the attitude anomaly. Although reconfigured on the ground, the sophisticated RVFS autonomously detected a failure, triggered a Disable Abort and safely prevented a collision.

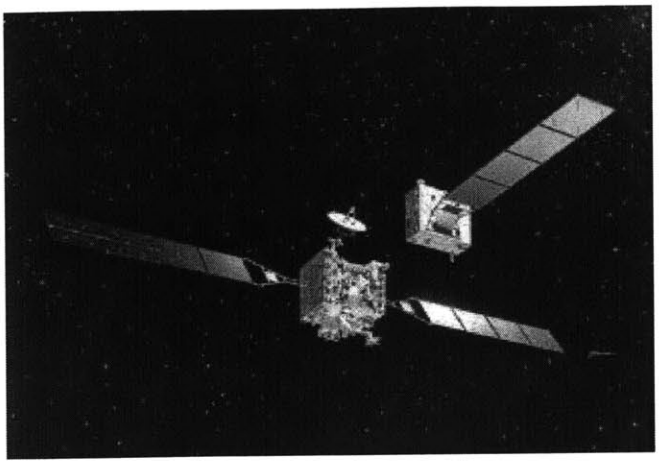


Figure 1-3: The Japanese ETS-VII experiment.

At the turn of the millennium, the Air Force, NASA and the Defense Advanced Research Projects Agency (DARPA) gave their approval for the execution of a series of autonomous RV, inspection and docking experiments in space. These experiments allowed scientists and engineers to develop and demonstrate the technologies necessary for the U.S. to bridge the gap and acquire the capability to perform autonomous docking in space. The first of these experiments to fly was the Experimental Satellite System-10 (XSS-10), an inspection demonstration by a micro-satellite (Fig. 1-4). The mission objectives of XSS-10 were to demonstrate autonomous navigation on a pre-planned course, semi-autonomous proximity operations (with the ground operators sending scripted top-level commands), and inspection of a Resident Space Object (RSO) from a distance of 35 meters [29]. Two minor problems occurred during the mission. On its first attempt to get an attitude solution (*lost-in-space* solution), the star tracker was fooled by bright objects in its field of view (possibly ejection debris). At that point, the computer initiated entry into a second stellar acquisition mode. With a successful second attempt, the ground controllers resumed the mission. Also, a temporary loss of signal (telemetry dropout) occurred when the satellite was closest to the RSO. No harm was done, although it was not possible to confirm how close the satellite was to the RSO because of the telemetry dropout.

The next related mission to be launched was the Experimental Satellite System-11 (XSS-11), another inspection demonstration (Fig. 1-5). This micro-satellite is

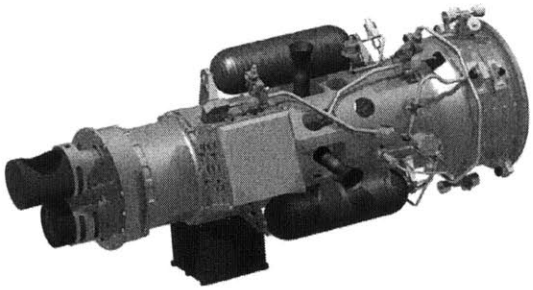


Figure 1-4: The XSS-10 micro-satellite.

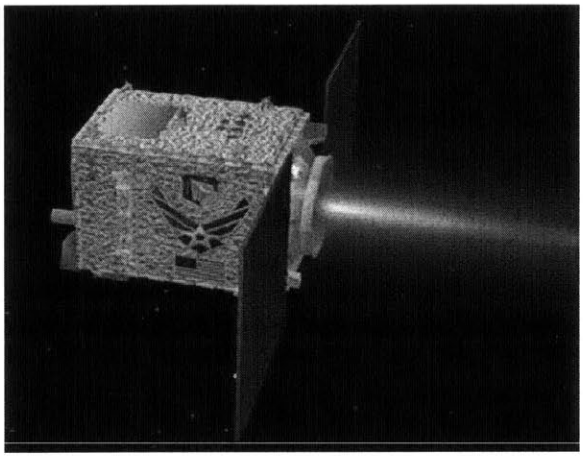


Figure 1-5: The XSS-11 micro-satellite.

designed to demonstrate technologies, including a scanning lidar [84, 83], to perform maneuvers around a space object without months of planning and with minimal ground support [135, 35]. The XSS-11 mission was originally planned to last between 12 and 18 months. It was intended to fly near and around other orbital objects (spent boosters and dead satellites owned by the U.S.) and inspect them. By the time of the writing of this thesis, no technical reports on the results of the mission were made public.

NASA's Demonstration of Autonomous Rendezvous Technology (DART) spacecraft (Fig. 1-6) launched a week after XSS-11, in April 2005. The objective of the DART mission was to demonstrate, in space, the hardware and software necessary for autonomous RV down to a separation of five meters with the target. The mission was intended to validate hardware like the Advanced Video Guidance Sensor (AVGS) [132] and ground simulation facilities using flight data [110]. The DART vehicle used

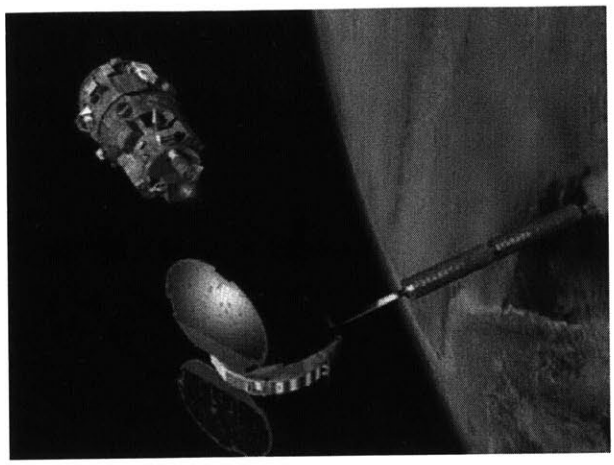


Figure 1-6: The DART rendezvous vehicle.

a linear static-gain feedback control law for both attitude and translational control, with a pulse-width modulated logic element [111]. A series of pre-computed jet-select maps were used for thruster actuation, which kept the computational load to a minimum. Unfortunately, the DART mission failed during the first approach to the target. The velocity measurement from DART's primary Global Positioning System (GPS) receiver was biased by 0.6 m/sec. This caused its navigation filter to diverge and its control system to fire thrusters to compensate for errors that did not exist. Also, DART's capability to autonomously avoid a collision proved to be ineffective as it eventually collided with the target. The inaccurate navigational information that caused DART's premature retirement resulted from a combination of [7]:

- an initial, unacceptable, calculated difference between DART's estimated and measured position that triggered a software reset;
- the introduction of an uncorrected, erroneous velocity measurement into the calculation scheme;
- a navigational software design that was overly-sensitive to erroneous data;
- the use of incorrect gain control in the calculation scheme.

The mishap investigation board concluded that there was an insufficient system-level understanding of the potential effects of complete or partial loss of functionality of

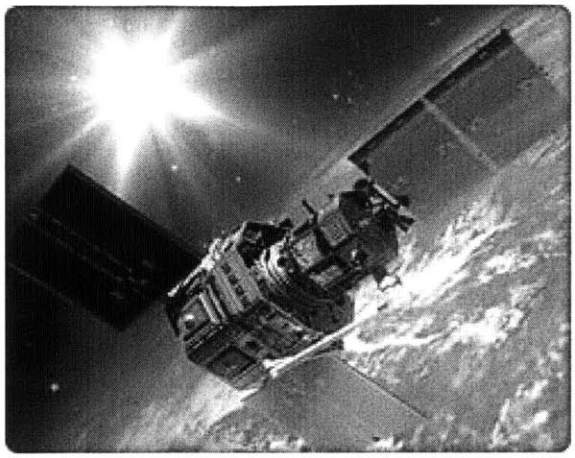


Figure 1-7: The Orbital Express flight demonstration.

relevant subsystems. The lack of thorough validation of math models and testing was also a significant causal factor in the mishap.

DARPA's Orbital Express Advanced Technology Demonstration (Fig. 1-7), launched in March 2007 and on-going, was the next mission dedicated to autonomous docking demonstrations in space. The goal of the program is to validate the feasibility of robotic, autonomous on-orbit refueling and reconfiguration of satellites to support a broad range of future U.S. national security and commercial space programs [115]. It will demonstrate critical technologies to enable on-orbit servicing systems including standard satellite servicing interfaces, autonomous GN&C systems, autonomous RV, proximity operations, docking, fluid and replacement unit transfer. Multiple docking scenarios with increasing difficulty will be attempted, gradually adding simulated failures to test the system's autonomous fault response capability. Some scenarios will involve an approach along a solar inertial approach corridor (spiral trajectory).

Each of the missions enumerated above involved an increasing level of autonomy. These missions provided valuable experience for future autonomous resupply missions to the ISS like the ones to be performed by the European ATV [32] or the Japanese HTV [66]. Advanced servicing missions like the Hubble Robotic Servicing and Deorbit Mission [18] will also benefit from the experience acquired in these past missions. Table 1.1 summarizes the missions covered in this historical review. It

Table 1.1: Historical review of autonomous docking missions

Mission	Year	Status	Type	Anomalies
Cosmos-186, Cosmos-188	1967	Success	Docking	Misaligned capture
ETS-VII	1998	Success	Docking	Thruster anomaly
XSS-10	2003	Success	Inspection	Navigation problem (confused star tracker), communication interruptions
XSS-11	2005	Unknown	Inspection	Unknown
DART	2005	Failure	Inspection	Navigation problem (biased velocity measurements), collision
Orbital Express	2007	On-going	Docking	Navigation problem (GPS-based system initialization problem)

is important to emphasize that anomalies occurred for each of these missions, with possibly the exception of the XSS-11 mission, which, however, has issued no public report. Moreover, none of the missions mentioned in Table 1.1 involved a tumbling target.

1.3 Previously proposed strategies for docking to a tumbling target

Although no autonomous docking missions to a tumbling target were ever attempted in space, the increasing interest behind satellite servicing missions is forcing engineers to reconsider this possibility. In a recent publication on satellite capturing strategies, Matsumoto et al. [80] presented a classification of different categories of satellite motion with respect to their angular rate and their attitude error. Following this classification, a satellite is considered to be tumbling if its angular rate is between 1 deg/sec and 18 deg/sec (3 RPM). Any motion with an angular rate higher than 18 deg/sec is categorized as a spin. The same classification is used below in describing the techniques developed over the years to accomplish docking with tumbling satellites.

From the 1960s to the mid 1980s, early docking navigation sensors like the Soviet

IGLA system could only provide range, range rate as well as bearing angles to the target (pitch and yaw rotation of the line-of-sight to the target). Roll information was only provided when the satellites were in close proximity [4]. Consequently, early control laws, ensuring RV and docking of two spacecraft, were constrained to use only these inputs. Aldrin [9] even proposed control laws using only line-of-sight angular information that can be measured by an astronaut during manned missions. These led to simple docking maneuvers performed in the same orbital plane as that of the target. Typically, the chaser initiated its approach by performing an orbit transfer maneuver from a parking orbit to the target satellite. With both spacecraft set to actively point at each other, the resulting trajectory appeared as docking with a tumbling target (which was actually controlling its tumbling rate). However, indications show that the resulting angular rate for typical orbits was of the same order of magnitude as the orbital rate ($\approx$0.07 deg/sec) and driven toward 0 deg/sec at docking [9]. Therefore, such a maneuver does not fall into the category of a docking to a tumbling target as per the classification mentioned above.

In the late 1990's, Tsuda et al. [120, 45] proposed an interesting technique that could allow docking to both spinning and tumbling satellites. They showed that two different spacecraft with the same moment of inertia ratios (MoIRs) can maintain the same rotation pattern without any control torque. Given an inertia tensor $\mathbf{I}$ and the corresponding principal moments of inertia I_1, I_2 and I_3 such that $I_1 \geq I_2 \geq I_3$, the MoIRs are defined as:

$$I_{max} \equiv I_1/I_2 \tag{1.1}$$

$$I_{min} \equiv I_3/I_2 \tag{1.2}$$

Figure 1-8 illustrates the proposed strategy for docking and capture. A chaser (the servicing satellite) is composed of long adjustable booms. After estimating the MoIRs and the center of gravity (CG) of the target, it surrounds it in order for its CG to overlap with that of the target, before it matches its MoIRs by adjusting the length and orientation of the adjustable booms. The next step is to synchronize with the

Figure 1-8: A proposed strategy for docking to a spinning target.

motion of the target. Once the motion is synchronized, the same rotation pattern as the target is maintained with virtually no control torque required. Finally, the target is captured, with no relative motion, using a robotic arm. Multiple estimation and optimal control techniques have been proposed by Tsuda et al. [120] to estimate and match the motion parameters of the target. However, the manufacturing, launch and control of a chaser with long flexible booms remain a challenge that is yet to be overcome.

Recent studies suggest the use of the orbital dynamics to naturally approach a tumbling target with a safe fly-by trajectory [81, 80]. The approach can be synchronized such that a grapple fixture on the target is grabbed by a robotic arm at the closest approach to the target during the fly-by, resulting in a capture. However, it is not clear if this technique will work for complex tumbling patterns (including nutation) or when there are appendages around the target. Moreover, it does not guarantee a direct line-of-sight with the target's docking port.

Finally, different methods have been recently developed to compute safe and fuel-efficient trajectories for docking with a tumbling target [14, 15, 63]. These techniques involve the computation of fuel-efficient trajectories to the target and the use of constraints or goals to prevent collisions and plume impingement. All of the strategies proposed in this section for docking to a tumbling target are theoretical and were not attempted on orbit, prior to the research presented in this thesis.

1.4 A previously proposed GN&C architecture

A GN&C architecture is defined in this thesis as *an abstract description of the entities of a GN&C system and the relationships between those entities.* This definition is adapted from a broader system architecture definition proposed by the MIT Engineering Systems Division Committee [26]. GN&C architectures found in the literature have common characteristics supporting the definition above [95, 44, 92, 41, 59, 54]:

- a decomposition in simple entities;
- a hierarchy between the entities;
- some interactions between the entities.

In his book entitled *Automated Rendezvous and Docking of Spacecraft* [40], Fehse presents a typical GN&C architecture traditionally used for automated docking (Fig. 1-9). The schematic he provides is simplified and shows only the levels of authority, not the actual functional relations within the system. Furthermore, he provides very few details on the different algorithms that can be used to populate the architecture. Fehse also gives an overview of how automated docking has been traditionally performed. He describes the different phases of a RV mission, the requirements for approach safety and collision avoidance, the drivers for the approach strategy, and a brief overview of the onboard RV control system.

The GN&C architecture provided by Fehse involves humans performing high level monitoring and making mission critical decisions, such as the triggering of a CAM when anomalies occur. However, when docking to a tumbling target, the short reaction time in case of anomalies requires an increased level of autonomy by the onboard computer, which is not supported by the GN&C architecture proposed by Fehse.

1.5 Problem statement

In Sections 1.2 to 1.4, it was shown that state-of-the-art docking operations not only require human monitoring to face anomalies, but are also performed with cooperative

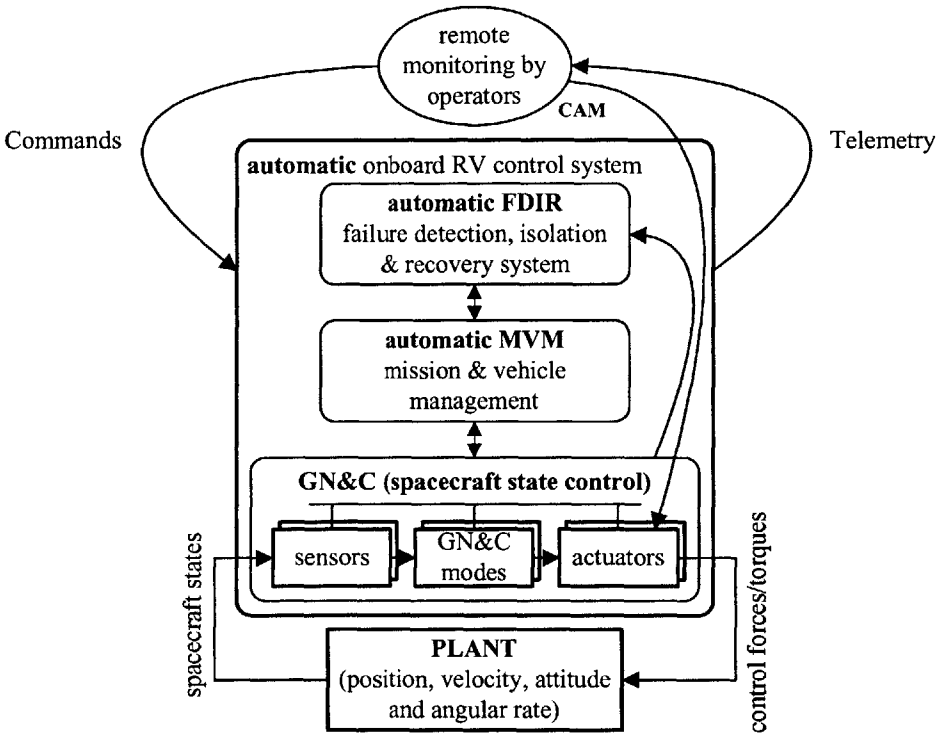


Figure 1-9: A typical control architecture used for automated docking [40].

targets (no tumbling targets). However, future space missions like satellite servicing will involve docking to a potentially tumbling target. These missions require computers to be granted more autonomy, in order to deal with anomalies and ensure the safety of the satellites involved in the mission. Therefore, there is a need to develop a system to support complex autonomous docking for future space missions, which is the subject of this thesis.

In this thesis, the chaser refers to the satellite performing the docking approach, while the target refers to the satellite that the chaser approaches. Four types of autonomous docking situations (ordered according to the level of cooperation between both satellites) have been identified:

1. *Target fully cooperating*: it is fully capable of communicating its states, controlling its attitude and displacements, and participating in the coordination of the docking maneuver (e.g., coordinated docking of two satellites of similar size).

2. *Target controlling its attitude only*: it is able to communicate its states, but

has control authority only over its attitude (e.g., docking with a fuel-depleted target using reaction wheels).

3. *Target drifting or tumbling*: it is able to communicate its states, but has no control authority on its displacements and attitude (e.g., docking with a fuel-depleted target with no reaction wheels).

4. *Target not cooperating*: there are two possible cases:

   - the target has no control authority, and no communication is occurring (e.g., docking with a dead satellite);
   - the target is actively trying to escape the chaser, and no communication is occurring (e.g., military applications).

The work presented in this thesis assumes knowledge of the states of the target. Because it is not the intention of the author to address advanced imaging techniques for state estimation, only the first three situations above are considered. Therefore, it is assumed that the target has the capability to sense its states and communicate them to the chaser whenever required.

More specifically, the objectives of this research are as follows:

- to develop a generic GN&C architecture that enables safe and fuel-efficient docking with a tumbling target, while maintaining a quick response to anomalies;
- to implement the GN&C architecture on hardware using GN&C algorithms relevant to autonomous docking;
- to demonstrate an autonomous docking maneuver with a tumbling target using the resulting GN&C software;
- to synthesize a generic process used to verify, prior to flight, the different components of the resulting GN&C software.

Docking itself can be divided into three stages: phasing (long-range); proximity operations; and terminal phase [97]. Phasing is the initial maneuver intended to match

orbital elements between the target and the chasing vehicles. Motion during this stage is governed by orbital mechanics, and can be planned offline. Proximity operations begin at about 40 km from the target. From this point, most of the maneuvers are calculated and executed onboard, either autonomously or manually by the flight crew. The terminal phase refers to the final hundred meters of the approach that ends in controlled physical contact. It is dominated by safety issues such as obstacle avoidance and plume impingement, rather than orbital mechanics. Most experts agree that the terminal phase is the most critical and challenging docking phase. Therefore, this thesis focuses on the terminal phase.

1.6 Thesis approach

In this thesis, the problem is approached at the system level. The approach consists to:

1. develop a modular GN&C software architecture enabling autonomous docking and formation flight on-orbit experiments;

2. populate it with existing algorithms spanning multiple areas of research like estimation, control, path planning, and fault detection, identification and recovery (FDIR);

3. refine it through hardware-in-the-loop testing;

4. extract a verification and validation (V&V) methodology that uses ground simulations and hardware-in-the-loop testing to increase the level of confidence prior to flight.

This approach is iterative and involves multiple series of experiments. The hardware used for the experimentation is the Synchronized Position Hold Engage and Reorient Experimental Satellites (SPHERES) facility, a series of micro-satellites flying *inside* the ISS to help engineers to develop and implement algorithms for autonomous docking and formation flight (Chapter 3).

The approach selected presents three key aspects: modularity in the development of the software, demonstration on hardware and a focus on the verification and the validation of the software prior to flight.

1.6.1 Software modularity

One of the key characteristics of the approach used in this research is modularity. A GN&C architecture is populated with multiple software modules. Each module contains an algorithm accomplishing a simple task, with its own inputs and outputs. Although they are designed to be independent, a hierarchy is permitted as one high level module can monitor many lower level modules. Standardization of the interfaces between the modules facilitates integration and reusability [113]. For example, most modules developed as part of this research were also used for formation flight experiments. Modularity provides the GN&C software with the capability to evolve and expand as new technologies become available.

The main advantages brought by modularity can be summarized as follows [13]:

- *Understandibility*: The behavior of every software module is understood without knowing details of other modules, which can allow different developers with different backgrounds to develop their own modules.

- *Maintainability*: Every software module is maintained or easily upgraded without affecting other modules.

- *Protection*: The consequences of a software error remain limited to the single module as long as the output of the module is not altered by the error. In general, this facilitates isolation and correction of the problem.

- *Reusability*: The same software module can be reused in different contexts or even in different systems, allowing the creation of a legacy of reliable test-proven modules.

1.6.2 Hardware-in-the-loop demonstrations

It is a common mistake to underestimate the time and effort required to migrate an algorithm from simulation to software executing in an embedded real-time computer. For the purpose of demonstrating an algorithm, engineers often prefer to use simulations coded in a software like MATLAB® [1]. These faster than real-time simulations allow quick and easy visualization of the results over a wide range of inputs.

However, the validity of a simulation's results is closely related to the validity of the various models embedded in the simulation. They can be over-simplistic, representing ideal situations. Therefore, some scientists claim that it is only through validation on realistic physical systems that a GN&C technology can go from *theoretical* to *applied* [82]. This thesis pursues that premise and presents the results of numerous experiments conducted on hardware in a variety of environments, including microgravity. These experiments validate the different modules used in the GN&C architecture and demonstrate autonomous docking maneuvers to both cooperative and uncooperative targets.

1.6.3 Verification & validation methodologies

One of the most difficult problems in the development, implementation and operation of a GN&C software architecture that enables autonomous docking is the development of sufficient confidence, prior to flight, that the system will perform in orbit as required by the mission objectives and as intended in the design [40]. Historically, V&V is a field of research that has been proven to be very useful for systems where testing on hardware is prohibitive in terms of cost. A good example is the field of computational fluid dynamics, where the use of simulations in the V&V process presents large cost benefits as opposed to expensive wind tunnel testing [88, 5].

Many techniques, processes and theories were developed over the years to validate computational models, in other fields of research, using limited experimentation [62, 21, 118]. Moreover, V&V is also often approached from a cost and risk management perspective [76, 75]. But for space missions, the V&V tools and techniques found in

the literature [36, 40] usually remain general and do not provide specific insights into the process. Therefore, this thesis places emphasis on the techniques used to qualify the software for autonomous docking prior to flight.

Definitions

The terminology used in the literature focusing on V&V differs slightly from one area of research to another. Therefore, it is necessary to clarify it to avoid confusion. The following definitions are used throughout this thesis [40, 5]:

Verification

- The proof that an item, function or process performs according to the specification, under which it has been developed.

Validation

- The proof that an item, function or process will behave as expected under real world conditions, or
- The proof that the description by mathematical modeling represents, to a sufficient level of accuracy, the behavior which an item, function or process would have under real world conditions.
- The level to which an item, function or process is validated corresponds to the Technology Readiness Levels (TRL) [50, 78].

Demonstration

- The operation of an item in front of witnesses, with the aim of providing evidence of proper function and performance.
- It differs from a verification as it is not a proof that the item fulfils all of the specifications.
- It differs from a validation as it is not a proof that the item functions and performs as required under all real world conditions.

Model

- A mathematical representation of a physical system or process intended to enhance the ability to understand, predict or control its behavior.

Simulation

- The exercise or use of a model (a model is used in a simulation).

Prediction

- The use of a model to foretell the state of a physical system under conditions for which the model has not been validated.

Calibration

- The process of adjusting numerical or physical modeling parameters in a computational model for the purpose of improving agreement with experimental data.

V&V platforms

Numerous platforms were built in the past for the verification or demonstration of various software or hardware components enabling autonomous docking. They can be classified in two categories. There are terrestrial platforms like flat floor vehicles [127, 67, 73, 108, 101], blimp units [90], underwater vehicles [8] and robotic facilities like the 12 degree-of-freedom (DOF) facility developed at the Naval Research Laboratory (NRL) [27] and the European EPOS simulator [55]. There are also space-operated platforms like the SPHERES facility [112, 57, 69, 68], the platforms developed through the University Nanosatellite Program [60], and also the former Mini AERCam [42, 43] and Personal Astronaut Assistant [34].

To date, terrestrial platforms do not provide a fully representative environment that would enable high fidelity 3-D simulation with hardware-in-the-loop testing. The facilities developed at the NRL allow the replication on the ground of six DOF motion of two spacecraft, but the quality of the dynamics being simulated is driven by the

quality of the models being used in the simulation. Because of its ability to operate both on the ground in a 2-D environment and inside the ISS in a 3-D environment, the SPHERES testbed was selected to implement and demonstrate the algorithms developed throughout this research.

1.7 Thesis roadmap

This chapter presented background information to motivate the need for a capability to perform an autonomous docking to a tumbling target. The problem is stated along with the approach selected to solve it.

In Chapter 2, existing GN&C algorithms relevant to autonomous docking are reviewed. They span areas of research like estimation, control, path planning, and FDIR.

In Chapter 3, the SPHERES testbed, the platform on which the experiments in this thesis are performed, is introduced along with different software tools to interface with it, or simulate its behavior. A GN&C architecture allowing autonomous docking to a tumbling target is proposed. The implementation of key GN&C algorithms is also discussed in detail.

In Chapter 4, the major difficulties encountered during the implementation process are discussed. The different GN&C software modules resulting from the implementation are then validated through a series of on-orbit experiments.

In Chapter 5, the integration of the different GN&C software modules, following the GN&C architecture proposed in Chapter 3, is covered in detail. Key experiments validating the integrated software are presented.

In Chapter 6, the most important docking experiment results are presented. These include autonomous docking to a cooperative target using straight and safe trajectories, as well as the very first autonomous docking to a tumbling target ever achieved in microgravity.

In Chapter 7, the process used to verify the GN&C software in the laboratory prior to flight is reviewed in detail. An extended process using newly available simulation

tools is also introduced.

Finally, in Chapter 8, a summary of the contributions made in this research is presented. Some recommendations for possible future work are also provided.

1.8 Main contributions

This section provides a list of the main contributions made throughout the research presented in this thesis. Each contribution is further explained in the following chapters.

- Developed a GN&C architecture for a nano-satellite, enabling on-orbit autonomous docking and formation flight experiments on a free flying platform in the ISS. Validated the GN&C architecture through 11 successful autonomous docking experiments and three successful formation flight experiments (Chapter 5 and 6)

- Developed a series of flight qualified GN&C modules to populate the architecture. Validated the modules through two experiments in the KC-135 and 23 in the ISS (Chapters 3 and 4)

- Created and implemented a process for the verification of GN&C software prior to flight. Validated the process through testing on an experimental free flying platform in the ISS (Chapter 7)

- Created simulation tools to support the implementation of GN&C algorithms for on-orbit formation flight and autonomous docking experiments (Chapter 3)

Two important space firsts were achieved in this research:

- First on-orbit autonomous docking to a tumbling target in microgravity (Chapter 6)

  - With both spacecraft having a similar mass

  - In spite of sensor errors that were autonomously detected and rejected online

- First autonomous formation flight performed inside of the ISS (Chapter 5)
  - The three spacecraft formation include two SPHERES satellites and the ISS (ultrasonic beacons were attached to the ISS)

Chapter 2

Review of GN&C algorithms for autonomous docking

The problem of autonomous docking is very complex. By itself, a docking maneuver involves navigating along an approach trajectory and capturing the target upon successful physical contact with it. This has been performed with humans-in-the-loop or solely through computers (Section 1.2). However, planning the maneuver and reacting to anomalies are traditionally taken care of by either the crew or the ground controllers that are monitoring the docking.

The purpose of the added autonomy when performing docking is to grant the onboard computer with the capability and the authority to replace the human presence in performing some or all of the high level functions where humans traditionally played a key role. Autonomy can be achieved at different levels. In addition to real-time navigation and control, it includes the capability to perform some or all of the following:

- plan a trajectory;
- detect, isolate and recover from failures in a timely manner;
- communicate relevant information with the other spacecraft involved in the docking;

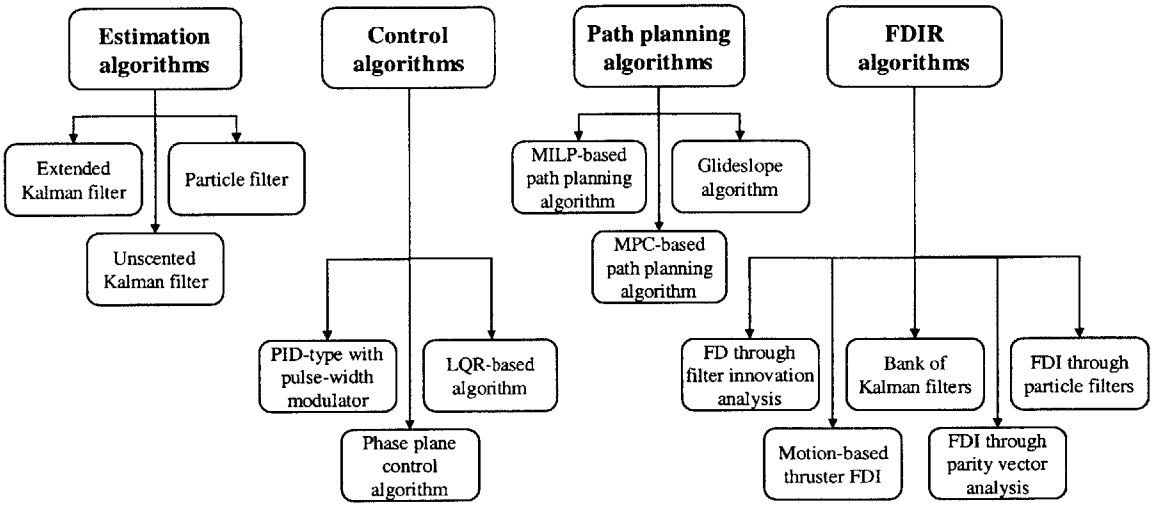


Figure 2-1: Overview of the algorithms presented in Chapter 2.

- schedule the different GN&C modes necessary for docking to occur.

Therefore, autonomous docking of two satellites involves many relevant areas of research, most of which are well established in the literature. A good understanding of the different available algorithms in each area of research is mandatory in order to select the right combination to integrate in a GN&C architecture.

This chapter covers the theory found in the literature behind common algorithms used when performing autonomous docking missions like the ones discussed in Section 1.2. Some relevant state-of-the-art algorithms that could be used in the near future are also provided. They are broken down into the following categories, as shown in Fig. 2-1, each forming a section in this chapter: state estimation, control, path planning and FDIR. A short description of the desirable features of algorithms covering each area is provided as well.

2.1 Review of estimation algorithms

State estimation algorithms process sensor data to provide state information (Fig. 2-2). It is highly desirable to use an algorithm that has guaranteed convergence within a reasonable time, and a guaranteed robustness and stability for the whole state space associated with the mission. However, for the case when the system is nonlinear (e.g.,

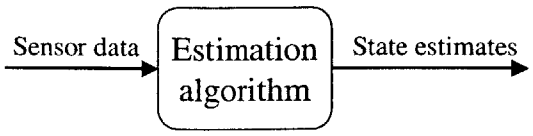


Figure 2-2: Input and output for a typical state estimation algorithm.

estimating the attitude of a satellite or processing time-of-flight measurements), this characteristic is very difficult if not impossible to prove mathematically. Extensive simulations are usually employed to validate the algorithm.

There exists a large number of state estimation methods that are suitable for estimating the various navigation states of a satellite, including the attitude states. Markley, Crassidis and Cheng [79] provided a thorough survey of the most relevant methods. Among the methods presented, three were further investigated in this thesis: the extended Kalman filter (EKF), the unscented Kalman filter (UKF) and the particle filter (PF). Each method has been used in the past for different applications. Convergence and robustness are not guaranteed for any of them when nonlinearities are present, but previous experience and simulations have shown that they work pretty well. They are covered in the following subsections.

2.1.1 Continuous-discrete extended Kalman filter (EKF)

The EKF [71, 16] forms the basis of most real-time spacecraft attitude estimation algorithms today. It has been used extensively for the last four decades. It uses the same mathematical scheme as the traditional Kalman Filter (KF), but is suited to nonlinear systems. Unfortunately, the guarantee of convergence and robustness usually associated with the KF does not hold for the EKF. Experience has shown that the EKF is appropriate for use in space missions provided that enough simulation is performed prior to the launch of the mission.

Over the years, many variants of the EKF have been developed, each with slightly different characteristics. However, all are constructed around the same foundation: a linearization of nonlinear equations that express the dynamics of the system and how the measurements relate to the states. This section introduces the theory from the

literature which is the basis of a continuous-discrete EKF.

System model

A system with continuous dynamics and discrete measurements can be modeled mathematically as [16]:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) + \boldsymbol{\varepsilon}(t) \qquad \boldsymbol{\varepsilon}(t) \sim \mathbf{N}[\mathbf{0}, \mathbf{Q}(t)] \tag{2.1}$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}(t_k), k) + \boldsymbol{\nu}_k \qquad \boldsymbol{\nu}_k \sim \mathbf{N}[\mathbf{0}, \mathbf{R}_k] \tag{2.2}$$

This model is valid for a time-varying system with nonlinear dynamics $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$, where $\mathbf{x}(t)$ is the state vector of dimension l and $\mathbf{u}(t)$ is the control input vector of dimension N. The measurements $\mathbf{z}_k$ made at discrete times t_k are related to the states by a nonlinear function $\mathbf{h}_k(\mathbf{x}(t_k), k)$. The vectors $\boldsymbol{\varepsilon}(t)$ and $\boldsymbol{\nu}_k$ are independent zero-mean white noise processes with time-varying spectral density $\mathbf{Q}(t)$ and covariance $\mathbf{R}_k$, respectively.

Like all Kalman filters, an EKF is a recursive process alternating between measurement updates and the propagation of the states using the dynamic model in between times when measurements are taken. General equations for both the measurement update phase and the propagation phase of the EKF are shown below.

Discrete measurement update equations

The measurements are processed using the standard EKF equations [16]. A Kalman gain $\mathbf{K}_k$ is first computed:

$$\mathbf{K}_k = \mathbf{P}_k^{(-)}\mathbf{H}_k^T\left(\hat{\mathbf{x}}_k^{(-)}, k\right)\left[\mathbf{H}_k\left(\hat{\mathbf{x}}_k^{(-)}, k\right)\mathbf{P}_k^{(-)}\mathbf{H}_k^T\left(\hat{\mathbf{x}}_k^{(-)}, k\right) + \mathbf{R}_k\right]^{-1} \tag{2.3}$$

where $^{(-)}$ means prior to the state update, $^{(+)}$ means after the state update and $\hat{}$ means an estimated value. The covariance matrix $\mathbf{P}_k$, computed in Eq. (2.7), represents the expected state error

$$\mathbf{P}_k = \mathbf{E}\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T\right] \tag{2.4}$$

and $\mathbf{H}_k\left(\hat{\mathbf{x}}_k^{(-)}, k\right)$ is defined as the Jacobian of $\mathbf{h}_k\left(\mathbf{x}\left(t_k\right), k\right)$:

$$\mathbf{H}_k\left(\hat{\mathbf{x}}_k^{(-)}, k\right) = \left.\frac{\partial \mathbf{h}_k\left(\mathbf{x}\left(t_k\right), k\right)}{\partial \mathbf{x}\left(t_k\right)}\right|_{\mathbf{x}(t_k)=\hat{\mathbf{x}}_k^{(-)}} \tag{2.5}$$

The Kalman gain is then used in both the measurement and the covariance update equations:

$$\hat{\mathbf{x}}_k^{(+)} = \hat{\mathbf{x}}_k^{(-)} + \mathbf{K}_k\left[\mathbf{z}_k - \mathbf{h}_k\left(\hat{\mathbf{x}}_k^{(-)}, k\right)\right] \tag{2.6}$$

$$\mathbf{P}_k^{(+)} = \left[\mathcal{I} - \mathbf{K}_k\mathbf{H}_k\left(\hat{\mathbf{x}}_k^{(-)}, k\right)\right]\mathbf{P}_k^{(-)} \tag{2.7}$$

where $\mathcal{I}$ is the identity matrix. The general nonlinear dynamics propagation equations are introduced next.

Continuous dynamics propagation equations

The following equations are the general propagation equations for a standard EKF [16]:

$$\dot{\hat{\mathbf{x}}} = \mathbf{f}\left(\hat{\mathbf{x}}\left(t\right), \mathbf{u}\left(t\right), t\right) \tag{2.8}$$

$$\dot{\mathbf{P}} = \mathbf{A}\left(\hat{\mathbf{x}}\left(t\right), t\right)\mathbf{P}\left(t\right) + \mathbf{P}\left(t\right)\mathbf{A}\left(\hat{\mathbf{x}}\left(t\right), t\right)^T + \mathbf{Q}(t) \tag{2.9}$$

where $\mathbf{A}\left(\hat{\mathbf{x}}\left(t\right), t\right)$ is the Jacobian of $\mathbf{f}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right), t\right)$:

$$\mathbf{A}\left(\hat{\mathbf{x}}\left(t\right), t\right) = \left.\frac{\partial \mathbf{f}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right), t\right)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(t)} \tag{2.10}$$

The equations presented in this section are very general and valid for a wide range of applications. This algorithm has been proven to be very valuable because of its suitability to nonlinear systems. However, for highly nonlinear systems, the linearization through the Jacobian might only be valid for time steps that are too small for the computation capability at hand. Moreover, for applications where the dynamics or measurement functions are non-differentiable, the Jacobian calculations are prob-

lematic. The technique presented in the next section is able of coping with such situations.

2.1.2 Discrete-time unscented Kalman filter (UKF)

The UKF [70, 74, 24, 25, 64] first appeared around the mid 1990's. Like an EKF, it is a recursive process that is based on the standard Kalman filter equations for the state update and propagation. However, it manipulates the covariance matrix differently. As opposed to linearizing a set of nonlinear equations like the EKF, it uses a fixed number of carefully selected parameters (sigma-points) that capture the first two moments (mean and covariance) of a Gaussian distribution. Each of the sigma points is propagated through the nonlinear function to yield a cloud of transformed points with transformed statistics, from which the mean and covariance are recalculated [74]. This technique presents several advantages over an EKF [79]:

- the expected error is lower;
- it can be used with non-differentiable functions;
- no Jacobian matrix calculations are required;
- it provides higher-order expansions;
- it generally converges faster;
- it is generally more tolerant to errors in the initial conditions.

This section introduces the general discrete-time equations used with the UKF, which are based on the same model as the one in Section 2.1.1. Both the propagation and state update equations are presented.

Dynamics propagation equations

Like the EKF, the states are propagated using Eq. (2.8) between time t_k and t_{k+1}. However, the propagation of the covariance is treated differently. A matrix composed

of sigma-point vectors $\boldsymbol{\chi}_k$ is formed from the $l \times l$ covariance matrix $\mathbf{P}_k^{(+)}$ as follows [79]:

$$\boldsymbol{\sigma}_k \leftarrow 2l \text{ columns from the matrix formed by } \left[\gamma\sqrt{\mathbf{P}_k^{(+)}}, \quad -\gamma\sqrt{\mathbf{P}_k^{(+)}}\right] \tag{2.11}$$

$$\boldsymbol{\chi}_k(0) = \hat{\mathbf{x}}_k^{(+)} \tag{2.12}$$

$$\boldsymbol{\chi}_k(i) = \boldsymbol{\sigma}_k(i) + \hat{\mathbf{x}}_k^{(+)} \tag{2.13}$$

where γ is a design parameter and $\sqrt{\mathbf{P}_k^{(+)}}$ is a shorthand notation for a matrix $\mathbf{Z}$ such that $\mathbf{Z}\mathbf{Z}^T = \mathbf{P}_k^{(+)}$ [25]. The selection of these sigma-point vectors keeps the first three moments the same as the original Gaussian distribution of the state estimates (the odd central moments are zero because of the symmetry of this set of points). The sigma-point vectors are then propagated through the nonlinear dynamics equation:

$$\boldsymbol{\chi}_{k+1}(i) = \mathbf{f}\left(\boldsymbol{\chi}_k(i), \mathbf{u}_k, k\right) \text{ for } i = 0, 1, ..., 2l \tag{2.14}$$

The propagated state estimates are obtained through the predicted mean at time t_{k+1} of the weighted sum of the sigma-point vectors:

$$\hat{\mathbf{x}}_{k+1}^{(-)} = \sum_{i=0}^{2l} W_i^{mean} \boldsymbol{\chi}_{k+1}(i) \tag{2.15}$$

where W_i^{mean} is a weighting parameter. For the purpose of clarity, the following substitution is made: $\hat{\mathbf{z}}_k^{(-)} \equiv \mathbf{h}_k\left(\hat{\mathbf{x}}_k^{(-)}, k\right)$. The predicted covariance ($\mathbf{P}_{k+1}^{(-)}$), output covariance ($\mathbf{P}_{k+1}^{zz}$), and cross-correlation ($\mathbf{P}_{k+1}^{xz}$) between $\hat{\mathbf{x}}_{k+1}^{(-)}$ and $\hat{\mathbf{z}}_{k+1}^{(-)}$ are calculated

by:

$$\mathbf{P}_{k+1}^{(-)} = \sum_{i=0}^{2l} W_i^{cov} \left[\boldsymbol{\chi}_{k+1}(i) - \hat{\mathbf{x}}_{k+1}^{(-)}\right] \left[\boldsymbol{\chi}_{k+1}(i) - \hat{\mathbf{x}}_{k+1}^{(-)}\right]^T + \mathbf{Q}(t)\left[t_{k+1} - t_k\right] \tag{2.16}$$

$$\mathbf{P}_{k+1}^{zz} = \sum_{i=0}^{2l} W_i^{cov} \left[\boldsymbol{\gamma}_{k+1}(i) - \hat{\mathbf{z}}_{k+1}^{(-)}\right] \left[\boldsymbol{\gamma}_{k+1}(i) - \hat{\mathbf{z}}_{k+1}^{(-)}\right]^T \tag{2.17}$$

$$\mathbf{P}_{k+1}^{xz} = \sum_{i=0}^{2l} W_i^{cov} \left[\boldsymbol{\chi}_{k+1}(i) - \hat{\mathbf{x}}_{k+1}^{(-)}\right] \left[\boldsymbol{\gamma}_{k+1}(i) - \hat{\mathbf{z}}_{k+1}^{(-)}\right]^T \tag{2.18}$$

where W_i^{cov} is another weighting parameter and $\boldsymbol{\gamma}_{k+1}(i)$ is defined as:

$$\boldsymbol{\gamma}_{k+1}(i) = \mathbf{h}\left(\boldsymbol{\chi}_{k+1}(i), k+1\right) \tag{2.19}$$

The weighting parameters W_i^{mean} and W_i^{cov} allow to tune the filter to account for the level of confidence that the Gaussian distribution is representative of the reality [25]. The reader should note that the discretization of the process noise matrix in Eq. (2.16) is a first order approximation and assumes that $(t_{k+1} - t_k) \rightarrow 0$. In fact, there exist numerous techniques to embed the process noise matrix in the equations for propagating the covariance [25, 122], each leading to a different level of accuracy. The technique presented in Eq. (2.16) is arguably the simplest and is valid when the process noise is purely additive in the model (Eq. (2.1)). The measurement update equations are introduced below.

Discrete measurement update equations

The first step in the measurement update process is to compute the innovations $\boldsymbol{\upsilon}_{k+1}$ and their associated covariance $\mathbf{P}_{k+1}^{\upsilon\upsilon}$ at time t_{k+1}:

$$\boldsymbol{\upsilon}_{k+1} = \mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1}^{(-)} \tag{2.20}$$

$$\mathbf{P}_{k+1}^{\upsilon\upsilon} = \mathbf{P}_{k+1}^{zz} + \mathbf{R}_{k+1} \tag{2.21}$$

The Kalman gain $\mathbf{K}_{k+1}$ is computed using:

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^{xz} \left(\mathbf{P}_{k+1}^{\upsilon\upsilon}\right)^{-1} \tag{2.22}$$

Finally, the updated states and covariance matrices are given by:

$$\hat{\mathbf{x}}_{k+1}^{(+)} = \hat{\mathbf{x}}_{k+1}^{(-)} + \mathbf{K}_{k+1}\boldsymbol{\upsilon}_{k+1} \tag{2.23}$$

$$\mathbf{P}_{k+1}^{(+)} = \mathbf{P}_{k+1}^{(-)} - \mathbf{K}_{k+1}\mathbf{P}_{k+1}^{\upsilon\upsilon}\mathbf{K}_{k+1}^{T} \tag{2.24}$$

The algorithm presented in this section has become increasingly popular largely because of the relative ease in generating sigma-points, as compared to computing the Jacobian functions in the EKF. However, it is generally accepted that it is not as computationally efficient as the EKF (some authors argue that the computational load is approximately twice as much as the EKF [25]).

For systems that are very nonlinear or present noise distributions that are not Gaussian, both the UKF and the EKF are likely to provide poor results. For these cases, the use of a particle filter might be the best approach.

2.1.3 Particle filter (PF)

A particle filter [79, 31, 99] is a suboptimal filter based on sequential Monte Carlo simulations. Instead of tracking a finite number of parameters describing a probability density function (PDF), like the mean and the covariance, the entire PDF is estimated through weighted particles (random samples) that are generated with pseudo-random number generators.

Particle filters are used when the dynamics of the system are highly nonlinear and where the PDF (after the propagation phase) is multi-peaked, heavily-tailed or skewed. Like the EKF and the UKF, it is composed of a propagation phase and an update phase. This section provides an overview of the general form of the PF algorithm.

System model

A PF allows for a more general system model than the EKF and the UKF. The following dynamics and measurement models for a discrete-time system are made without any assumption on the noise parameters $\boldsymbol{\varepsilon}_k$ and $\boldsymbol{\nu}_k$, other than being white noise processes:

$$\mathbf{x}_{k+1} = \mathbf{f}_k\left(\mathbf{x}_k, \mathbf{u}_k, k, \boldsymbol{\varepsilon}_k\right) \tag{2.25}$$

$$\mathbf{z}_k = \mathbf{h}_k\left(\mathbf{x}_k, k, \boldsymbol{\nu}_k\right) \tag{2.26}$$

Also, the three PDFs given by $\mathbf{p}\left(\mathbf{x}_0\right)$, $\mathbf{p}\left(\boldsymbol{\varepsilon}_k\right)$ and $\mathbf{p}\left(\boldsymbol{\nu}_k\right)$ are assumed to be known and independent. Therefore, the probabilities $\mathbf{p}\left(\mathbf{x}_{k+1} \mid \mathbf{x}_k\right)$ and $\mathbf{p}\left(\mathbf{y}_k \mid \mathbf{x}_k\right)$ can be derived from Eq. (2.25) and Eq. (2.26).

The measurement history $\boldsymbol{\mathcal{Z}}_k$ is defined as:

$$\boldsymbol{\mathcal{Z}}_k \equiv \left[\mathbf{z}_1^T, \mathbf{z}_2^T, ..., \mathbf{z}_k^T\right]^T \tag{2.27}$$

At each time step k, the state distribution $\mathbf{x}_k \mid \boldsymbol{\mathcal{Z}}_k$ is represented by a set of Π particles $\{\mathbf{X}_k(i)\}_{i=1}^{\Pi}$ with associated normalized weights $\{\boldsymbol{\mathcal{W}}_k(i)\}_{i=1}^{\Pi}$ satisfying:

$$\sum_{i=1}^{\Pi} \boldsymbol{\mathcal{W}}_k(i) = 1, \qquad \boldsymbol{\mathcal{W}}_k(i) \geq 0 \qquad \forall i \tag{2.28}$$

At any time, a state estimate can be computed, if necessary, through a weighted average of the particles $\{\mathbf{X}_k(i)\}_{i=1}^{\Pi}$:

$$\hat{\mathbf{x}}_k = \sum_{i=1}^{\Pi} \left[\mathbf{X}_k(i) \cdot \boldsymbol{\mathcal{W}}_k(i)\right] \tag{2.29}$$

State propagation process

The state propagation process simply consists of the propagation of each particle. Every time the particles are propagated, the process noise $\boldsymbol{\varepsilon}_k$ is sampled Π times from its distribution. Equation (2.25) is then used directly to propagate each particle.

Measurement update process

The measurement update process consists of two steps: updating the weight associated with each particle and resampling the particles. First, the weights are updated using the measurements as follows:

$$\mathcal{W}_{k+1}(i) \propto \mathbf{p}\left(\mathbf{Z}_{k+1} \mid \mathbf{X}_{k+1}(i)\right) \mathcal{W}_k(i) \tag{2.30}$$

The resulting weights are then normalized to satisfy Eq. (2.28).

To keep a sufficient diversity among the particles, and to avoid their tendency to degenerate because of the growing weight on only a fraction of the particles, a resampling is needed. Multiple resampling techniques exist [79, 31] including a commonly accepted one stating that the probability that a particle is resampled is proportional to its weighting parameter $\mathcal{W}_k(i)$ (also called the Roulette selection process). Using this technique, a particle can be selected more than once, and particles with low weighting factors are likely to disappear.

The number of particles plays an important role in the accuracy and the behavior of a PF. Unfortunately, the number of particles required increases exponentially with the number of elements in the state vector[1]. For a PF, the computation requirement does not depend on the complexity of the model, but rather on the number of particles made available to the algorithm. There is a tradeoff to be made between the computation time and the quality of the approximation. Multiple variants of the PF were developed to reduce the computational requirement while maintaining good performance [79, 99]. Some of them are designed such that their number of particles varies in time, allowing the algorithm to adapt online to changes in the required accuracy.

[1]To ensure convergence of the algorithm, a large number of particles is initially required to span the whole state space and increase the chance that a couple of particles will be in the vicinity of the correct solution.

Table 2.1: Brief comparison of estimation algorithms.*

	EKF	UKF	PF
Performance with nonlinear systems	Baseline	+	+
Speed of convergence		+	+
Computational requirement		-	-
Ease of implementation		+	-
Previous experience in space		-	-

*Comparison with the baseline in each category:
+: better performance
s: similar performance
-: lower performance

2.1.4 Summary of estimation algorithms

Three state estimation algorithms were investigated in this section: an EKF, a UKF and a PF. Table 2.1 summarizes the main characteristics of each algorithm. The EKF is the most widely used of the three and has been used extensively in previous space missions. It generally offers good performance at an acceptable computational cost, although the time it takes to converge is generally longer than for the others and the computation of Jacobians might complicate its implementation. The UKF presents better performance than the EKF for highly nonlinear systems, and is easier to implement. But the computational cost is approximately twice that of the EKF. The PF is usually the last resort for highly nonlinear systems when a UKF cannot meet the specifications. Its performance generally depends on the number of particles used, which also greatly affects the computational cost.

2.2 Review of control algorithms

In this thesis, control algorithms encompass the category of algorithms that process tracking errors in order to generate thruster commands (Fig. 2-3). A very large body of literature has been established on techniques for controlling a thruster-based spacecraft. A wide variety of control algorithms have been designed for both linear

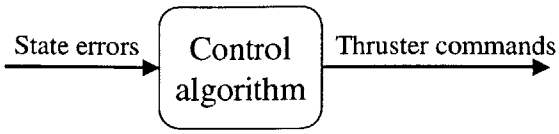


Figure 2-3: Input and output for a typical control algorithm.

(trajectory control) and nonlinear (attitude control) systems. Guaranteed stability, robustness and low fuel consumption, although very hard to obtain in practice, are highly desired characteristics of these algorithms.

Because of the type of hardware available for experimentation (Section 3.1), only algorithms suitable to ON/OFF thrusters are investigated. This category of thrusters involves thrusters that can either be turned ON or OFF, therefore producing a fixed thrust for a variable duration. For the purpose of this research, three algorithms were retained for their simplicity and wide use: proportional-integral-derivative (PID), phase plane and linear-quadratic regulator (LQR) controllers. These algorithms are described in more detail in the following sections.

2.2.1 PID-type control algorithm with a pulse-width modulator

PID controllers are the most commonly used form of dynamic compensation. They produce a command output c using an error input e:

$$c = K_p e + K_i \int e \, dt + K_d \dot{e} \tag{2.31}$$

where K_p, K_i and K_d are referred to as proportional, integral and derivative gains, respectively. Multiple well known techniques can be used to analyze the performance of this control law or to compute the appropriate gains to meet the specifications [126, 30]. These techniques are not covered here. A control law like the one in Eq. (2.31) can be developed independently for each DOF under control, resulting in a six-element commanded output, typically referred to as the commanded force-torque

vector:

$$\mathbf{C} = [\mathbf{F}_x \quad \mathbf{F}_y \quad \mathbf{F}_z \quad \mathbf{T}_x \quad \mathbf{T}_y \quad \mathbf{T}_z]^T \tag{2.32}$$

where $\mathbf{F}$ and $\mathbf{T}$ are vectors of commanded forces and torques expressed in the body frame. Because of the thruster arrangement around a spacecraft, a thruster management algorithm is needed to convert the commanded force-torque vector into proper thruster commands. The purpose of a thruster management algorithm is to select which thruster to turn ON and for how long to leave it ON to provide the required control output. Also, the ON/OFF nature of pulsed thrusters requires the thruster management algorithm to be compatible with this nonlinearity.

There exists a wide range of thruster management algorithms suitable for ON/OFF thrusters. Some have demonstrated the ability to minimize a certain cost function (either fuel consumption or deviation from the desired control output) [133]. Although such an algorithm presents certain advantages, such as being robust to uncertainty in actuator and sensor performance, it requires the implementation of a real-time linear program (LP) solver which can be difficult in practice. Other simpler techniques are designed to overcome the nonlinearity associated with ON/OFF commands through modulation.

Modulation techniques are based on the assumption that nonlinearities and discontinuities that occur on small time scales may be averaged out, leading to effectively linear, continuous actuation over longer time scales [57]. One of these techniques, a pulse-width modulator, involves pulses at a regular interval (fixed frequency), but of varying durations. This technique is a natural choice for most modern clocked digital computers because of their fixed-frequency nature. It was therefore selected, in this thesis, to interface with the PID controller.

With a zero-order hold, converting discrete thrust commands to the continuous time domain, the thrust duration can be derived through conservation of thrust impulse during each control period (also the pulse interval). In Fig. 2-4, the thrust impulse is represented by the shaded area under the curve. For each control period, a commanded impulse is derived from a commanded thrust. Knowing the actual

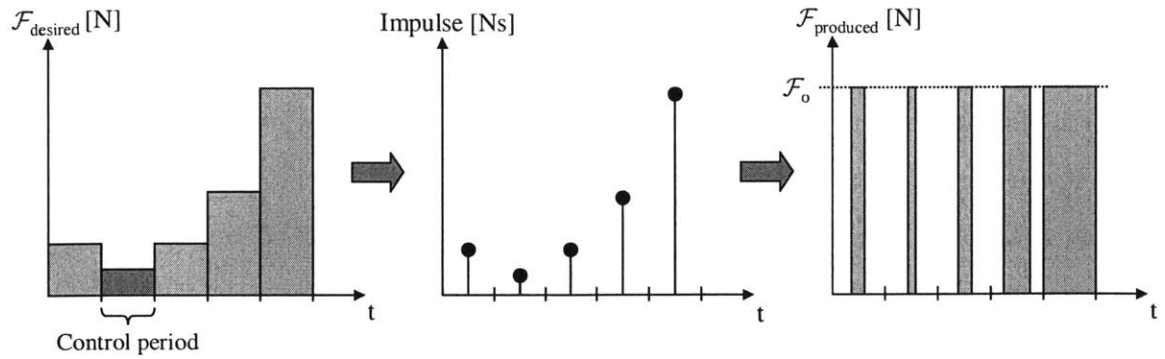


Figure 2-4: Conversion of the thrust commanded to thruster ON/OFF time with the use of the pulse-width modulator.

thrust produced by the thrusters, the commanded impulse can be converted to thrust duration.

The control input vector $\mathbf{u}$ can be defined as a vector where each element represents the fraction of the control period $\mathcal{P}$ for which the corresponding thruster is commanded ON. Neglecting any transient effects caused by the opening or the closing of the thrusters, it is computed from the commanded force-torque vector $\mathbf{C}$ as follows:

$$\mathbf{u} = \boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{F}}, \mathbf{D}, \mathbf{L}, B) \cdot \mathbf{C} \tag{2.33}$$

where $\boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{F}}, \mathbf{D}, \mathbf{L}, B)$ is the thrust mapping matrix that is a function of the nominal forces of thrusters $\boldsymbol{\mathcal{F}}$, their thrust direction $\mathbf{D}$, their location in the body frame $\mathbf{L}$ and the blowdown factor B associated with the number of thrusters opened simultaneously (opening multiple thrusters simultaneously causes a drop of pressure, which also causes a drop in the thrust produced by each thruster) [131, 12]. Multiplying the control input vector $\mathbf{u}$ by the control period $\mathcal{P}$ produces the ON-time commanded for each thruster. Multiple techniques exist to compute the thrust mapping matrix [57, 12], involving mostly offline computation. They are not covered here.

The control input vector $\mathbf{u}$, expressed in Eq. (2.33), can be converted into thruster ON-times simply by multiplying it with the control period. Also, the thruster pulse can be centered in the control period for increased accuracy when trying to follow a straight path using thrusters with different thrust. Because $\mathbf{C}$ is unconstrained, it

is possible that the control input vector $\mathbf{u}$, computed in Eq. (2.33), results in pulse durations longer than the control period or negative pulse durations [12]. Therefore, post-processing, like scaling of the control inputs, might be required to ensure a feasible solution. However, such a scaling leads to a loss of the impulse applied, which can negatively affect the performance of the controller.

There exist other algorithms that do not risk the problem of saturating the control inputs. The phase plane control algorithm presented in the next section is one of them.

2.2.2 Phase plane control algorithm

Phase plane control algorithms have been widely used in space applications since the beginning of the space age. The controller presented here is based on the work previously done by Draper Laboratory [109] and NASA Ames in their development of modern spacecraft controllers. It processes state errors using a phase plane logic to generate pre-determined thrust pulses, therefore avoiding thruster saturation unless specified by the user. Each DOF (attitude or position) is controlled independently and can have a different phase plane logic associated with it.

Figure 2-5 shows a typical phase plane logic used by a phase plane attitude controller (only the upper half is shown, the lower half being essentially the same but rotated 180 degrees around the origin). A similar phase plane logic can be developed for position control. The x-axis is the angle error (in degrees) and the y-axis is the rate error (in deg/sec). The phase plane is divided into five regions by switch curves (S1, S2, S3, S4, S5). S1 and S4 are determined by the angular acceleration capability associated with that particular axis and by the user-selected dead band. S3 limits the admissible rate error. S5 defines the drift channel (Region 4) and S2, the non-firing zone (Region 5). The locations of S2 and S5 depend on the noise level in the state estimates. They are adjusted to minimize propellant use and avoid jittering back and forth between two regions.

The controller works as follows. When state errors (e.g., angle and angular rate) are computed, they are translated to a phase point. Depending on the region where the point is located in the phase plane, and the logic associated with that region, the

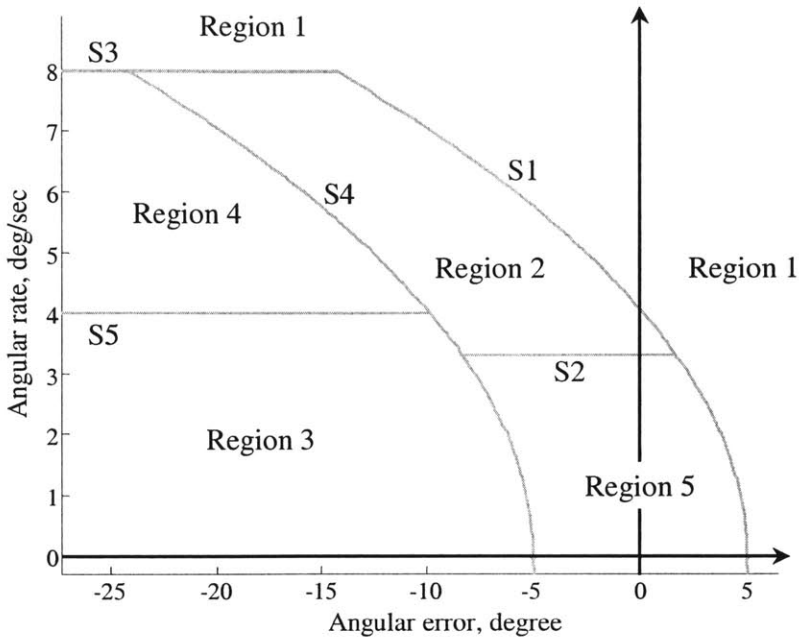


Figure 2-5: Phase plane logic for attitude control (upper half only).

thrusters acting around the appropriate axis may be actuated. When the phase point is in Region 1, the satellite is either too far from the desired location (in the positive direction), moving away from it or approaching it with too much velocity. Thrusting has to occur to reduce the velocity. In Region 2, thrusting occurs only when the phase point crosses S3 to get into it. On the other hand, if the phase point crosses S4 to get into Region 2, the satellite coasts until the phase point reaches S1. A phase point in Region 3 means that the satellite is either located too far in the negative direction or is approaching the desired location with too little velocity. Thrusting occurs to increase its velocity. When the phase point is in the drift channel (Region 4), the satellite is coasting toward the desired location (zero position error). The non-firing zone defined by the dead band of the system is the target zone toward which all trajectories converge. No firing occurs when the phase point is in Region 4 or 5. After the trajectory converges to Region 5, the resulting motion is a limit cycle around that region.

The continuous analog of the phase plane control algorithm presented in this section (when the sampling frequency grows to infinity and with 100% duty cycle) provides a Bang-OFF-Bang type control input that is a known solution to an opti-

mal control problem. Multiple control algorithms present an explicit solution to an optimization problem. The one presented in the next section is an example.

2.2.3 LQR control algorithm

LQR control algorithms [117] are very popular because of their simplicity, optimality, and inherent compatibility with state-space systems. They arguably represent the simplest form of optimal control algorithms. They are the algorithms of choice for state-space systems, and are often used as a reference for the comparison of different control algorithms.

The general form of a typical LQR control algorithm is now described in more detail. Given a discrete time-variant controllable system:

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k \tag{2.34}$$

where $\mathbf{x}_k$ is a state vector at time t_k, $\mathbf{A}_k$ is the dynamics matrix, $\mathbf{B}_k$ is the control effect matrix, $\mathbf{u}_k$ is the control input that is driving $\mathbf{x}$ to zero and $\mathbf{x}_0$ is a known initial condition. A quadratic cost function can be used to express a cost J associated with the state and the control inputs over a finite horizon of S steps:

$$J = \mathbf{x}_S^T \boldsymbol{\mathcal{Q}}_f \mathbf{x}_S + \sum_{i=0}^{S-1} \left(\mathbf{x}_i^T \boldsymbol{\mathcal{Q}} \mathbf{x}_i + \mathbf{u}_i^T \boldsymbol{\mathcal{R}} \mathbf{u}_i \right) \tag{2.35}$$

with given constant state and control input weighting matrices:

$$\boldsymbol{\mathcal{Q}} = \boldsymbol{\mathcal{Q}}^T \geq 0 \tag{2.36}$$

$$\boldsymbol{\mathcal{Q}}_f = \boldsymbol{\mathcal{Q}}_f^T \geq 0 \tag{2.37}$$

$$\boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{R}}^T > 0 \tag{2.38}$$

It can be shown that the optimal control input has a solution of the form

$$\mathbf{u}_k = \mathbf{K}_k \mathbf{x}_k \tag{2.39}$$

where the gain matrix $\mathbf{K}_k$ is computed using

$$\mathbf{K}_k = -\left(\boldsymbol{\mathcal{R}} + \mathbf{B}_k^T \mathbf{P}_{k+1} \mathbf{B}_k\right)^{-1} \mathbf{B}_k^T \mathbf{P}_{k+1} \mathbf{A}_k \tag{2.40}$$

$\mathbf{P}_k$ satisfies the algebraic Riccatti equation

$$\mathbf{P}_{k-1} = \boldsymbol{\mathcal{Q}} + \mathbf{A}_k^T \mathbf{P}_k \mathbf{A}_k - \mathbf{A}_k^T \mathbf{P}_k \mathbf{B}_k \left(\boldsymbol{\mathcal{R}} + \mathbf{B}_k^T \mathbf{P}_k \mathbf{B}_k\right)^{-1} \mathbf{B}_k^T \mathbf{P}_k \mathbf{A}_k \tag{2.41}$$

The recursion process for $\mathbf{P}$ is solved backward in time (from $k = S, ..., 1$) and is initiated with

$$\mathbf{P}_S = \boldsymbol{\mathcal{Q}}_f \tag{2.42}$$

For linear systems, since the computation of the gain matrix $\mathbf{K}_k$ in Eq. (2.40) does not depend on the states or the control inputs, the gain history can be computed offline, reducing the online computation to the matrix multiplication in Eq. (2.39).

The weighting matrices $\boldsymbol{\mathcal{Q}}, \boldsymbol{\mathcal{Q}}_f$ and $\boldsymbol{\mathcal{R}}$ are determined by the user and can be tuned to meet the desired specifications. A close analysis of Eq. (2.39) shows that an LQR control algorithm is in fact the equivalent of a proportional-derivative (PD) control algorithm with time-variable gains. Therefore, depending on the selection of the control input vector $\mathbf{u}$ and the matrix $\mathbf{B}$, an LQR control algorithm might have to be coupled with a pulse-width modulator like it is for the PID control algorithm in Section 2.2.1. Other versions of this algorithm exist. Among them are a version allowing for a receding horizon and one for an infinite horizon.

2.2.4 Summary of control algorithms

Three control algorithms were introduced in this section: a PID control algorithm coupled with a pulse-width modulator, a phase plane algorithm and an LQR control algorithm. Table 2.2 summarizes their main features. All three algorithms were used extensively in the past and involve approximately the same low level of online computation. The PID control algorithm can potentially provide higher accuracy because of both the variable duration of the generated pulses and the integral term.

Table 2.2: Brief comparison of control algorithms.*

	PID	Phase plane	LQR
Potential accuracy	Baseline	-	-
Overall robustness		+	S
Fuel efficiency		+	+
Optimized		S	+
Online computational requirement		S	S
Ease of implementation		-	-
Previous experience in space		S	-

*Comparison with the baseline in each category:
+: better performance
s: similar performance
-: lower performance

It is also the easiest to implement because it does not have to deal with switch curves or the solution of an optimal problem, although it still has to be analyzed offline. On the other hand, experience has shown that the generation of pulses with fixed duration by the phase plane control algorithm makes it is less prone to diverge in case of temporary errors in the generation of the state estimates. In terms of fuel efficiency, the LQR algorithm is the only algorithm directly derived to optimize a cost function, making it the most fuel efficient algorithm among the three presented in this section. However, the control input history produced by the phase plane control algorithm is similar to Bang-OFF-Bang control which is a known solution to an optimal fuel efficient control problem, making the phase plane control algorithm also very fuel efficient. The next section covers path planning algorithms, which are used to generate desired trajectories that a control algorithm tracks.

2.3 Review of path planning algorithms

The role of a path planning algorithm is to provide trajectories, for both the position and the attitude of a satellite, from the actual location to the desired target location (Fig. 2-6). In order to conserve fuel, which is a determining factor in the total lifetime

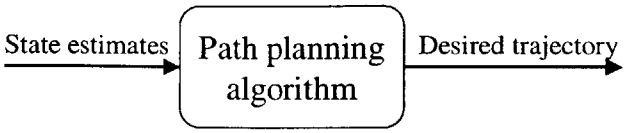


Figure 2-6: Input and output for a typical path planning algorithm.

of a satellite, fuel-optimal trajectories are highly desirable. Consideration not only for obstacle avoidance but also for plume impingement and line-of-sight are other desirable features. Robustness to perturbations caused by modeling errors is also important. Finally, to be of practical interest, a path planning algorithm has to be manageable by an onboard computer and require low computation, once the tracking of the plan is initiated, in order to quickly respond to unpredicted events.

This section describes three algorithms that were selected. The first generates fuel-optimal trajectories, using mixed-integer linear programming (MILP), to account for constraints like collision avoidance, plume impingement and even passive safety towards the end of the trajectory. The second uses parametric programming and model predictive control (MPC), which solves for the optimal control as a piecewise affine function of the states. Finally, the third algorithm under investigation is a simple but robust trajectory planning algorithm called the glideslope algorithm.

2.3.1 Optimal path planning using MILP

Early development of a MILP-based path planning algorithm at the MIT Aerospace Controls Laboratory (ACL) was performed by Richards et al. [102, 103]. The following formulation is based on the recent development by Breger [15, 14], who extended the work by Richards to linear time-variant systems. Given an initial state vector $\mathbf{x}_0$ and a discrete linear time-invariant dynamics model (for the sake of simplicity in the formulation):

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \tag{2.43}$$

the state at any future step k is described by:

$$\mathbf{x}_k = \mathbf{A}^k\mathbf{x}_0 + \begin{bmatrix}\mathbf{A}^{k-1}\mathbf{B} & \mathbf{A}^{k-2}\mathbf{B} & \dots & \mathbf{AB} & \mathbf{B}\end{bmatrix}\begin{bmatrix}\mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{k-1}\end{bmatrix} \tag{2.44}$$

Using this formulation, an optimization problem can be formed that optimizes and constrains the control inputs, and also constrains the states of the system. The cost function for this problem over S steps is:

$$J = \min_{\mathbf{u}_0,\dots,\mathbf{u}_{S-1}} \sum_{i=0}^{S-1} || \mathbf{u}_i ||_1 \tag{2.45}$$

where $|| \cdot ||_1$ is the 1-norm that captures the fuel expenditure. At each step k, the control inputs can be directly constrained using:

$$\mathbf{u}_{min_k} \leq \mathbf{u}_k \leq \mathbf{u}_{max_k} \tag{2.46}$$

The states can be constrained using a set of p constraint functions. Two formulations can be used to express the constraints. If the states are constrained to lie within a convex region, as it is the case for the final states in a docking problem, the constraint functions can be formulated as:

$$\mathbf{G}_k\mathbf{x}_k \leq \mathbf{c}_k \tag{2.47}$$

where $\mathbf{G}_k$ is a set of constraint functions and $\mathbf{c}_k$ is a vector of bounds for these functions at time t_k.

A problem, with constraints formulated as in Eq. (2.47), can be solved using a LP solver. If the states are constrained to remain outside of a region to avoid a collision, the constraint functions can be formulated using binary variables as [102]:

$$\mathbf{G}_k\mathbf{x}_k \leq \mathbf{c}_k + M\mathbf{y}_k \tag{2.48}$$

$$|| \mathbf{y}_k ||_1 \leq p - 1 \tag{2.49}$$

where M is a large number as compared to the elements in the state vector and $\mathbf{y}$ is a vector of binary variables. When set to one, a binary variable permits relaxation of its corresponding constraint by guaranteeing that the inequality is satisfied. The idea is if any of the inequality functions in Eq. (2.48) is satisfied without relaxing the constraint, then the states lie outside of the constrained region. This is guaranteed by Eq. (2.49), which ensures that at least one constraint is not relaxed. Richards et al. extended this formulation to account for plume impingement with obstacles and other vehicles [102], while Breger added passive safety abort, using safety constraints, toward the end of the trajectory [15].

A constraint formulation with binary variables requires the use of a MILP solver. It is computationally intensive, making its implementation in a real-time system very challenging. Therefore, this technique is suitable for computing a trajectory offline, and having a control algorithm tracking the trajectory online. If a problem occurs along the trajectory, other techniques are better suited to re-plan a trajectory online, such as the one shown in the next section.

2.3.2 Robust path planning through MPC

For discrete linear time-invariant systems, with constraints on the states and the control inputs, a technique has been developed by Bemporad et al. to determine explicitly the solution to an optimal control problem that can be formulated using LP techniques [11]. The resulting control profile is a piecewise affine and continuous function of the initial state, which can be computed offline. The online computation is reduced to a simple function evaluation, making this technique very attractive for real-time systems, with limited onboard computational power. The solution found offline is an exact solution, the equivalent of solving an open-loop optimal control problem over a finite horizon at each time step.

Because of the complex nature of this technique, the details behind the problem statement and the computation of optimal control profiles are omitted here, and can be found in Ref. [11, 104]. However, the use of the control profile online is illustrated in Fig. 2-7. For every control loop, the MPC algorithm computes the optimal control

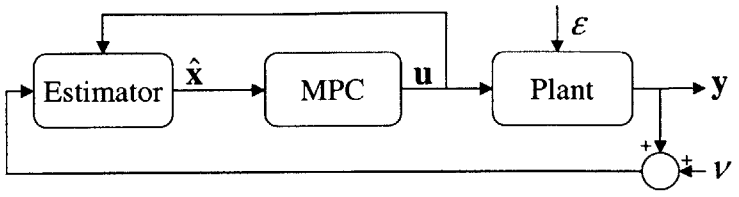


Figure 2-7: Typical control loop using the MPC algorithm with an explicit control solution [104].

input from the actual states and the control profile kept in memory. This calculation involves a simple table lookup. The control input is then applied over the next time step, and the process is repeated. Richards has extended this technique to include a MILP formulation, robustness through constraint tightening techniques, adaptation to uncertainty from online measurements and a decentralized formulation [104].

Although very promising, the main limitation behind this technique is that it faces the curse of dimensionality. The computation requirement increases exponentially with the number of states, especially when a MILP formulation is used and the number of constraints is high. Moreover, the optimal control profile, that is computed offline, also increases in size exponentially with the number of states. Therefore, a simpler algorithm that can be handled more easily by an onboard computer is needed. The algorithm introduced in the next section possesses these characteristics.

2.3.3 The glideslope algorithm

The glideslope algorithm [51, 94] is a common and widely used algorithm (Apollo, Shuttle) for trajectory planning in real time. It is mainly used when a straight line approach to the target is required, often because of line-of-sight constraints with docking navigation sensors. It is an algorithm that prescribes a proper approach velocity pattern to follow in the phase plane using a pre-determined number of thruster pulses, equally spaced in time (Fig. 2-8). The approach velocity ($\dot{\rho}$) is commanded to decrease linearly with the distance-to-go (ρ). The maneuver has to occur in a period $\mathcal{T}$ with a safe commanded arrival velocity ($\dot{\rho}_T$) and using a pre-determined number of thruster firings (Fig. 2-8). This simple algorithm can be described by the following

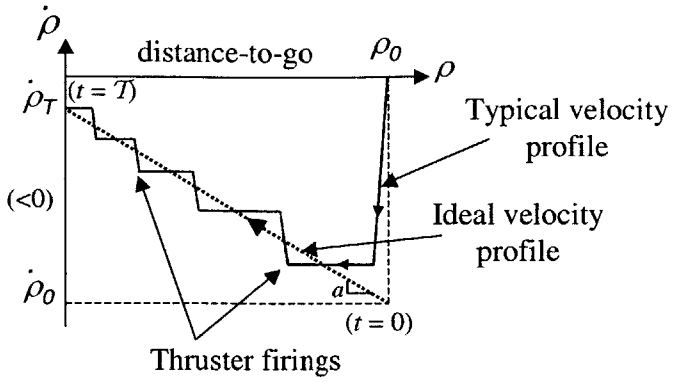


Figure 2-8: Typical approach velocity profile computed by the glideslope algorithm (system is initially at rest, five thruster pulses are allowed).

equation,

$$\dot{\rho} = a\rho + \dot{\rho}_T \tag{2.50}$$

where a is the glideslope (<0). The solution of that differential equation is

$$\rho(t) = \rho_0 e^{at} + \frac{\dot{\rho}_T}{a}(e^{at} - 1) \tag{2.51}$$

while the maneuver period $\mathcal{T}$ is

$$\mathcal{T} = \frac{1}{a} \ln\left(\frac{\dot{\rho}_T}{\dot{\rho}_0}\right) \tag{2.52}$$

A glideslope algorithm is a hybrid between a path planning algorithm and a velocity control algorithm. At the appropriate time to apply a velocity correction, the algorithm determines the magnitude of the velocity correction based on the velocity required to reach the next waypoint by the time the next velocity correction is applied (Eq. (2.51)). The result is an actual trajectory, in the phase plane, that oscillates around the desired velocity pattern as shown in Fig. 2-8. When performing a straight approach, such an algorithm is appropriate to plan and control the trajectory along the docking axis[2], while a different control algorithm can be used to maintain alignment transverse to the docking axis.

The glideslope algorithm can easily be adapted to evasive type maneuvers by

[2]The docking axis is defined as the axis normal to the docking port of the target.

Table 2.3: Brief comparison of path planning algorithms.*

	Glideslope	MILP	MPC
Obstacle / plume avoidance capability	Baseline	+	+
Optimized		+	+
Robustness		s	+
Offline computational requirement		-	-
Online computational requirement		+	s
Ease of implementation (offline part)		-	-
Ease of implementation (online part)		+	s
Previous experience in space		-	-

*Comparison with the baseline in each category:
+: better performance
s: similar performance
-: lower performance

selecting different initial and final commanded velocities. It is very simple, robust, easy to implement and requires low computational power. However, optimality is not guaranteed and the presence of obstacles is not taken into account. Nevertheless, it has the nice property of reducing the ΔV induced at each burn toward the end of the maneuver, which in some cases can help to reduce the effects of plume impingement.

2.3.4 Summary of path planning algorithms

Three path planning algorithms were presented in this section. Table 2.3 summarizes the main characteristics of each. In terms of the capability to avoid obstacles and plume impingement, the MILP-based algorithm performs better than the MPC-based one because of its capability for use with time-varying systems. However, the MPC-based algorithm is the only one that directly accounts for uncertainty in the model, making it the only truly robust algorithm among the three. Among all three algorithms, only the glideslope algorithm does not involve the solution to an optimization problem. The algorithm that requires the most offline computation is the MPC-based path planning algorithm. As for online computation, because the plan is entirely computed offline, the MILP-based algorithm performs the best since its out-

put is directly provided to the control algorithm. Although the MPC-based and the glideslope algorithms integrate the functionality of both a path planning and a control algorithm, using simple functions, they still require slightly more online computation time than the MILP-based algorithm. The MPC-based algorithm is the hardest to implement offline because of its complexity, while the MILP-based algorithm only requires the solution of a MILP problem. Since the path to follow is provided at each time step, the MILP-based algorithm requires only minimal effort to implement online, while the others require only the implementation of simple functions. Finally, only the glideslope algorithm has been used on past missions.

2.4 Review of FDIR algorithms

FDIR is a very active area of research [31, 49, 131, 128, 28]. FDI algorithms are very well established and were successfully implemented on various military and civil aircraft as well as on the Space Shuttle. An FDIR algorithm also has the recovery capability which allows for a greater level of autonomy. It must be able to isolate a failure and maintain the functionality of the system. If recovery is not possible, it must trigger an abort. Failures must be detected at all levels: from system level to lower levels in the GN&C software functions, and in the sensor and the actuator hardware. It is also desirable to have the capability to detect and isolate two or more simultaneous failures, whenever possible.

Five algorithms were investigated with the capability to detect, isolate and sometimes recover from sensor and actuator failures. The first algorithm uses the innovation in the computation of the measurement update in the EKF or the UKF to detect measurement errors. The second is a motion-based maximum-likelihood thruster FDI algorithm used to detect and isolate stuck-ON and stuck-OFF thruster failures. The third consists of a bank of KFs: one for each failure mode and one for the case of no failure. The fourth uses parity vectors in the parity space to detect and isolate sensor failures when redundant sensors are available. Finally, the last is an extension of the PF that accounts for different failure modes.

2.4.1 FD through filter innovation analysis

Like other Kalman filter techniques, the EKF and the UKF have built-in fault detection capability when the sensor measurements $\mathbf{z}_k$ are compared to the expected measurements given the state estimates $\mathbf{h}_k\left(\hat{\mathbf{x}}_k^{(-)}\right)$, as in Eqs. (2.6) and (2.20). After convergence of the state estimates (which can be based on criteria determined through simulation), the result of this comparison (often called residual or filter innovation) can be used as a verification that the measurements are coherent with the state estimates [40]. Non-coherence is a sign of a problem that can be related to the sensor used when taking the measurements, to the actuators or to numerical problems with the algorithm itself.

The technique is simple. A filter innovation threshold is determined offline through simulation. This threshold is usually a function of sensor noise as well as desired rates of false positive and false negative diagnosis. The required online computation consists of comparing the computed filter innovation with the threshold. An innovation above the threshold is a sign of non-coherence and the measurement is rejected. The threshold is adjusted such that a certain fault pattern is likely to be the main trigger of the alarm, while other types of faults are more likely to be picked up by other algorithms.

Because the measurement update process of the EKF and the UKF already involve the computation of an innovation, the added computation required by this technique is negligible, which is a great advantage. Although the filter innovation provides fault detection capability, it does not necessarily provide fault isolation capability unless the failure to be isolated has a clear signature through the filter innovation.

2.4.2 Motion-based thruster FDI

A thruster FDI algorithm was developed by Deyst and Deckart [33] for the detection and isolation of thruster failures on the Space Shuttle. It has recently been refined by Wilson et al. [131, 128] to detect and isolate thruster failures on the X-38 escape vehicle using inertial navigation sensors (INS).

The concept behind this algorithm follows. The algorithm is divided into three main segments. The first segment consists of the computation of a residual acceleration vector as follows. Every time the inertial navigation sensors are sampled, the data is filtered to produce a vector of actual angular and linear accelerations. At the same time, a detailed dynamics model of the satellite is used to compute an expected linear and angular acceleration vector given the actual commanded thrusters. The subtraction of the actual from the expected accelerations produces the residual acceleration vector.

In the second segment, the residual acceleration vector is compared with a catalog of pre-computed residual accelerations corresponding to each possible thruster failure mode, for both stuck-ON or stuck-OFF failures. Before detection occurs, only the active fault modes are retained for comparison.[3] A maximum likelihood function identifies which active fault modes (including the case of no fault) is likely to correspond to the actual residual accelerations. If only the case of no fault is retained, no fault detection occurs. As soon as one or multiple fault modes are retained, a fault is detected and the algorithm switches to isolation mode (most of the time, more than one fault mode is initially retained).

The third segment consists of isolating the failure by process of elimination. In isolation mode, both active and inactive fault modes are used in the comparison of the residual accelerations in an attempt to exonerate fault modes among the ones that were retained earlier. A fault mode is exonerated, whether it is active or not, when its corresponding residual acceleration does not match anymore the actual residual acceleration. If all faults are exonerated prior to having only one remaining, a false alarm is declared. Once all but one fault mode have been exonerated, the fault is successfully isolated, and corresponds to the remaining fault mode.

For increased accuracy in the calculation of the residual accelerations, this algorithm can be used in parallel to an online mass and thruster property identification algorithm [130, 129]. The residual accelerations can also be computed from a KF,

[3]The notion of active fault mode is related to its observability at any time. Stuck-ON failures are only observable when the thrusters are not commanded, while stuck-OFF failures are only observable when the thrusters are commanded.

which requires more computation. A nice property of this algorithm is its capability to reconfigure itself and remain functional, after the isolation of a failure, by simply switching a flag allowing the model of the system to take the failure into account when computing subsequent expected accelerations.

This technique assumes that the INS used in the calculation of the residual accelerations is healthy. Because it is run independently of the state estimation algorithm, it is insensitive to errors in the state estimates. However, since it is called every time the INS is sampled (which usually is at a high frequency), it can be computationally intensive. Also, it can only detect and isolate faults that are catalogued, which is reasonable given the small number of thrusters on modern satellites. Recent progress by Wilson [131] suggests that this technique can also be applied to the occurrence of more than one fault simultaneously, as long as the signature is properly catalogued.

2.4.3 Bank of Kalman filters

To detect sensor and actuator faults, the use of a bank of KF has been a method of choice almost since the appearance of the KF itself [77, 10, 49]. The technique consists of running in parallel a series of Kalman filters, each tuned to a particular fault mode through its embedded model, including one for the case of no fault. The filter innovation (Section 2.4.1) for each KF is monitored. Among all of the innovations, at any point in time, only one is consistently near zero mean, when the fault mode it represents is active, which corresponds to the current failure mode.

This method can be quite computationally intensive and hard to implement because of the use of multiple online KFs. It does, however, have the capability to recover given a failure. Once the FDI problem is solved, the KF providing the near zero innovation is the one to be trusted. The information it provides can then be used by the controllers.

2.4.4 FDI through parity vector analysis

A FDI algorithm was developed in the 1970's for the detection and isolation of sensor faults. This algorithm, often referred to as FDI through parity vector analysis, is especially well suited for systems with redundant INS oriented in different directions [53].

The concept behind the algorithm is now described. A parity vector is defined in the parity space formed by the sensing directions of the redundant sensors. When a failure occurs, the parity vector grows in a direction corresponding to the failed sensor. The details behind the algorithm are omitted here and can be found in Ref. [52].

To detect and isolate a failure, at least two measurements more than the number of states being sensed are required. For example, when the x-, y- and z-rates are being sensed, five gyroscope measurements are required to allow the detection and the isolation of the failure of a single gyroscope. The orientation of each sensor can be selected to equalize the estimation error variance in all directions [96], while providing uniform detectability of failures [28].

The main advantage of this method is its insensitivity to actuator failures or failures of other types of sensors since gyroscope measurements are compared to each other. Once a failure has been isolated, recovery occurs by discarding the data from the failed sensor. The algorithm then remains fully functional, provided that there are enough redundant sensors remaining. However, if redundant sensors are oriented in the same direction, voting techniques have to be used, based on a direct comparison of the sensor outputs, to isolate a failure.[4]

2.4.5 FDI through particle filters

The last FDI algorithm covered is an extension of the PF and is often referred to in the literature as the hybrid PF [31]. Like the traditional PF, it is a relatively new concept. It can be used to detect and isolate actuator failures.

[4]Voting techniques only work if at least three redundant INS oriented in the same direction are available.

A hybrid PF is a PF that is augmented with a discrete state variable Y representing, at any time, the actual mode that the system is in. Given a set of m possible modes

$$Y = y_1, ..., y_m \tag{2.53}$$

there is a transition function representing the probability that the variable transitions to a new mode y_j at time t_k:

$$\mathbf{p}\left(Y_k \mid Y_{k-1}, \mathbf{x}_{k-1}\right) \tag{2.54}$$

Each mode is characterized by a dynamics model, similar to the one in Eq. (2.25), reflecting the dynamics of the system given the state of fault or no fault.

The algorithm is now described in more detail. A set of Π particles $\{\mathbf{X}_k(i)\}_{i=1}^{\Pi}$ is initially obtained from the sampling of Eq. (2.54) Π times and is approximately evenly distributed in the state space spanned by the continuous state vector. The state propagation process is initiated by sampling a new mode for each particle through Eq. (2.54). Then, the corresponding process noise $\boldsymbol{\varepsilon}_{k_j}$ associated with the new mode y_j for that particle is also sampled. The particle is thereafter propagated according to the dynamics equation corresponding to the mode y_j.

The state update phase proceeds as for the standard PF, except that the weighting function also takes into account the mode after the propagation:

$$\boldsymbol{\mathcal{W}}_{k+1}(i) \propto \mathbf{p}\left(\mathbf{Z}_{k+1} \mid Y_{k+1}(i), \mathbf{X}_{k+1}(i)\right) \boldsymbol{\mathcal{W}}_k(i) \tag{2.55}$$

Resampling of the particles also occurs as for the standard PF. At any time, the state estimate $\hat{\mathbf{x}}_k$ is obtained through the weighted average expressed in Eq. (2.29), while the probability of being in a certain mode y_j can be approximated by the sum of the weights of the particles in that mode:

$$p\left(Y_k = y_j\right) \approx \sum_i \boldsymbol{\mathcal{W}}_k(i)|_{Y_k(i)=y_j} \tag{2.56}$$

Whenever that probability increases above a predetermined threshold, a fault corresponding to the fault mode y_j is successfully detected and isolated. The filter remains fully functional after the failure, allowing for a recovery.

Like any other PF, the hybrid PF has the advantage of being able to cope with highly nonlinear systems. The hybrid PF presented in this section adds FDI capability to a standard PF. The main difficulty with such a technique is the enormous number of particles required to avoid sample impoverishment, where fault modes with non-zero probability of being the actual state of the system contain no sample. This makes it very challenging to embed a hybrid PF in a real-time system. Nevertheless, some techniques exist to reduce the number of particles while keeping the capability to avoid false negatives [31].

2.5 Summary of FDIR algorithms

This section presented five FDIR algorithms: FD through filter innovation analysis, motion-based thruster FDI, bank of Kalman filters, FDI through parity vector analysis, and FDI through particle filters. Table 2.4 summaries the main characteristics for each of them. FD through filter innovation analysis and FDI through parity vector analysis are two techniques able to detect sensor failures. Motion-based thruster FDI and FDI through particle filters allow for thruster failure detection. The only algorithm allowing detection of both sensor and thruster failures is the bank of Kalman filters. All techniques presented in this section, except FD through filter innovation analysis, provide isolation and recovery capability. But they all involve more computation and are more difficult to implement. Finally, all techniques have acquired experience in space missions or in the aviation industry, except for the motion-based thruster FDI (the version introduced by Wilson) and FDI through particle filters, which are fairly new techniques.[5]

[5]It is important to mention here that although the motion-based thruster FDI algorithm introduced by Wilson has not been used in a space mission to date, it has been successfully demonstrated on the SPHERES facility during a series of experiments in the ISS.

Table 2.4: Brief comparison of FDIR algorithms.*

	FD through navigation filter analysis	Motion-based thruster FDI	Bank of Kalman Filters	FDI through parity vector analysis	FDI through particle filters
Capability to cope with sensor failures	Baseline	-	S	S	-
Capability to cope with thruster failures		+	+	-	+
Capability to isolate a failure		+	+	+	+
Capability to recover after a failure		+	+	+	+
Computational requirement		-	-	-	-
Ease of implementation		-	-	-	-
Previous experience		-	S	S	-

*Comparison with the baseline in each category:
+: better performance
s: similar performance
-: lower performance

2.6 Summary

This chapter described the main characteristics of multiple estimation, control, path planning and FDIR algorithms that can be integrated into a GN&C architecture for performing autonomous docking. The theory behind most algorithms was provided. Finally, a brief comparison of the algorithms in each category was performed based on a literature review and on the experience of the author with each of them. The next step is to develop a GN&C architecture compatible with these algorithms, and to implement some of these algorithms on hardware.

Chapter 3

Implementation of GN&C algorithms

The first step in developing and populating a GN&C architecture for real-time, hardware-in-the-loop implementation of autonomous docking was to implement key algorithms in an embedded computer, and make them available as software modules ready to be integrated into an architecture. Migrating algorithms from equations to embedded software requires time and effort. The development of a representative hardware setup is a major part of that effort. Accessibility to such a testbed, especially for space applications, is often very limited and costly. This is a reason why engineers traditionally evaluate GN&C algorithms through simulation rather than actual testing. Therefore, the body of literature covering the experimental implementation of GN&C algorithms for space applications is limited.

Through the development of SPHERES, a satellite testbed designed to fly inside the ISS, the MIT Space Systems Laboratory (SSL) acquired relevant experience in the hardware implementation of GN&C algorithms. In fact, the availability of a series of GN&C software modules that were flight qualified through on-orbit testing is an important outcome and contribution of this research.

This chapter addresses the implementation of key algorithms spanning the areas of estimation, control, path planning and FDIR. It is intended to provide clear instructions on the adaptation of algorithms for hardware-in-the-loop testing. This chapter

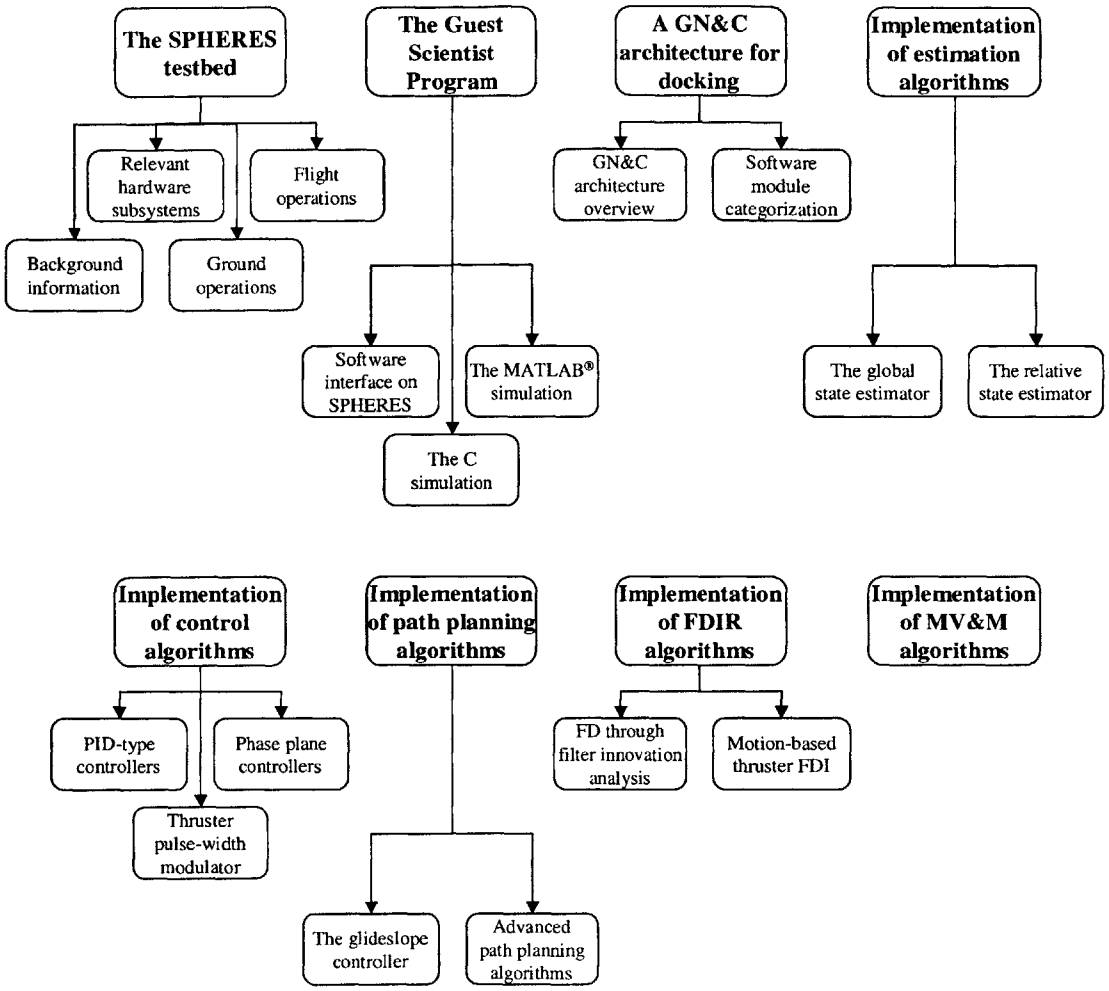


Figure 3-1: Overview of Chapter 3.

is divided as shown in Fig. 3-1. The SPHERES testbed is first introduced, followed by the Guest Scientist Program (GSP): the software interface developed by the MIT SSL. The GN&C architecture used for autonomous docking is then presented. The next two sections cover the implementation of the navigation and control algorithms. Finally, the current implementation of path planning, FDIR, and mission and vehicle management (MVM) algorithms is discussed.

3.1 The SPHERES testbed

The MIT SSL has developed the SPHERES testbed to provide researchers with an experimental laboratory to develop and test formation flight and autonomous docking

algorithms [112, 113]. Due to the limited experience and testing of these algorithms in a representative microgravity environment, formation flight and docking technologies are considered high risk. However, these technologies can provide significant benefits as they can enable coordinated motion of multiple satellites that perform missions such as sparse aperture telescopes, interferometry, re-supply, assembly and satellite servicing.

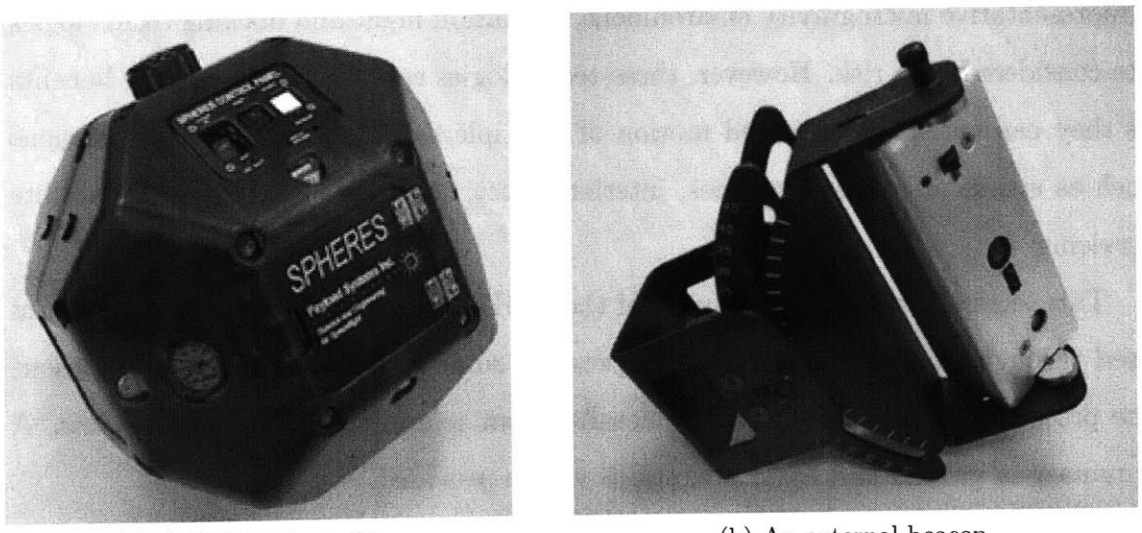
This section provides an overview of the SPHERES testbed, the testing platform used throughout this research. It first presents some background information about the project and then provides more details about relevant hardware subsystems. A summary of ground and flight operations is also provided.

3.1.1 Background information

The SPHERES testbed consists of multiple nano-satellites (Fig. 3-2a) which can control their relative positions and orientations in a six DOF environment. The testbed is primarily designed to operate inside the ISS, but it can also operate onboard NASA's Reduced Gravity Aircraft (both the C-9B and the former KC-135) [6], as well as in a 2-D environment (three DOF motion) on a flat floor (like the one at the NASA Marshall Space Flight Center (MSFC) Flight Robotics Laboratory [119]) or on a laboratory air table (Fig. 3-3). A total of six satellites have been built for two long-term facilities: a ground laboratory (three satellites) and the ISS laboratory (three satellites).

Typically, test platforms designed to operate in space, such as the Japanese ETS-VII satellites [89, 65], DART [110, 7] or Orbital Express [115], are exposed to the risk of unrecoverable failures if an anomaly occurs in the GN&C software. By operating inside the ISS, SPHERES exploits the microgravity environment to represent the dynamics of distributed satellite and docking missions, while preventing the testbed from experiencing unrecoverable failures when real or simulated GN&C failures occur. It is a testbed designed specifically to perform flight research and to exercise the flight software within a short timeframe[1] in a microgravity environment at a small

[1]The time between the design of an algorithm and the analysis of the flight data.



(a) A SPHERE satellite.

(b) An external beacon.

Figure 3-2: SPHERES hardware.

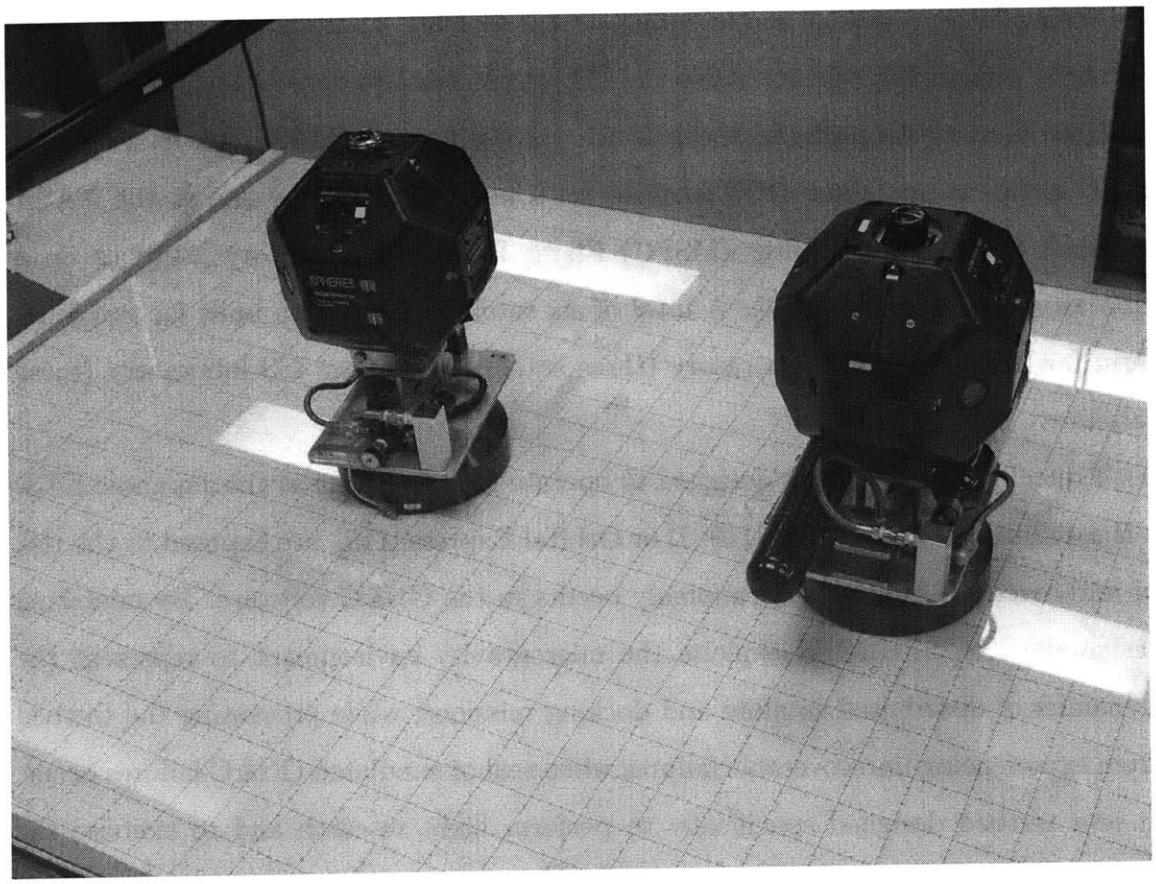


Figure 3-3: MIT SSL 2-D air table.

fraction of the cost of conventional flight demonstrations. It provides a cost-effective, long duration, replenishable, risk tolerant, and easily reconfigurable platform for the execution and testing of algorithms in microgravity.

An interesting analogy to formation flight at close proximity can be made with people driving their car on a highway. Surprisingly few collisions occur. However, one does not toss the keys of an expensive car to a teenager and tell to learn to drive in this environment. Instead, learning generally occurs in a less expensive car, in a less risky yet increasingly challenging environment, until the handling of nominal and many off-nominal situations becomes second nature. SPHERES provides exactly that environment for distributed satellite GN&C. It is commonly used to demonstrate high-risk relative position and attitude control algorithms [86], collision avoidance algorithms, fault detection, isolation and recovery algorithms [131], docking control architectures [87], TPF-like formation flight [68, 69], and even tethered formations [23, 22]. These high risk algorithms can be developed and tested in a low risk - long duration, shirtsleeve microgravity environment. Therefore, SPHERES allows engineers to push the algorithms to their limits in various realistic mission scenarios, learning about both their theoretical and physical limitations, without concern for the loss of the satellites.

To describe the maturity of new technologies, like the ones enumerated above, NASA and the Department of Defense (DoD) use the TRL metric [50, 78]. A detailed description of the TRLs is provided in Appendix A. Figure 3-4 illustrates how ISS microgravity experimentation with a testbed like SPHERES can increase the maturity of the software modules, with key characteristics and the corresponding TRL at each step [113]. Because of a lack of accessibility to a microgravity testing platform, engineers typically have to resort to ground-based facilities to test their GN&C software [40]. These facilities involve integrated tests, with a mix of hardware-in-the-loop and software-in-the-loop testing. They can bring the TRL of GN&C software up to TRL-5, and sometimes TRL-6 when the simulation has been calibrated using flight data. SPHERES provide accessibility to a microgravity environment to further increase the fidelity of the testing and therefore the maturity of the algorithm. It can

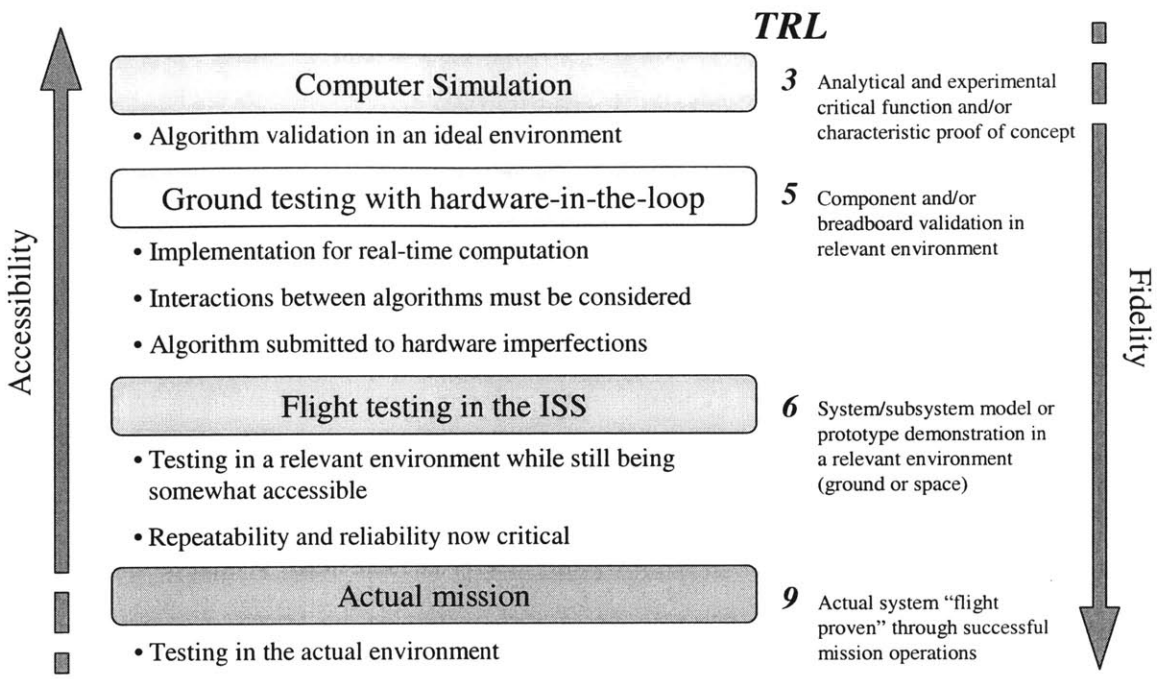


Figure 3-4: ISS experimentation to increase the TRL of docking algorithms.

bring GN&C software modules and the GN&C architecture up to TRL-6 through microgravity hardware-in-the-loop testing. It can also be used to validate simulation models through experimentation in microgravity. These models can then be used to verify different phases of an actual mission, prior to flight, through simulation.

Another feature of the SPHERES testbed is the flexibility of the interface with the hardware. This was necessary to permit testing of algorithms that arise from a variety of research areas ranging from autonomous docking to formation flight and FDIR. SPHERES can also simulate different levels of cooperation between space vehicles. Full knowledge of the satellite's own position as well as the position and attitude of one or more additional satellites can be systematically degraded to realistically simulate various failure scenarios.

SPHERES is composed of the following [113, 57, 58, 12, 19]. A NASA laptop computer onboard the ISS is used as a ground station to transmit software and commands to the satellites and record telemetry. The ISS crew members use the laptop to start tests or change test configurations. The test configuration involves the satel-

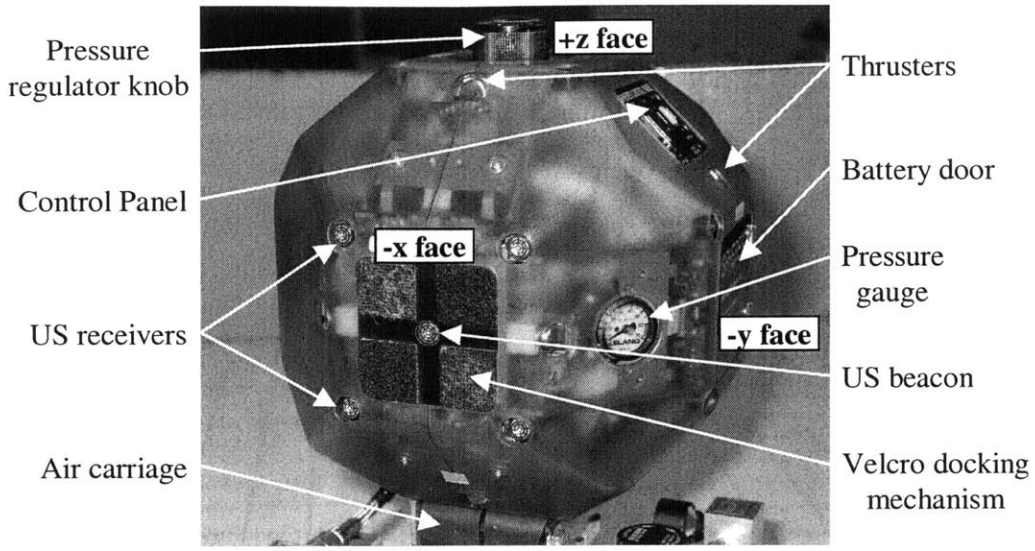


Figure 3-5: Visible components around the satellite and body axes.

lites being used, the location of the external beacons and the program loaded in each satellite. Each satellite is self-contained, with all the necessary subsystems to operate autonomously. Figure 3-5 illustrates different components of the SPHERES satellites, along with the body axis corresponding to each face.[2] Power is provided by 16 AA batteries. The propulsion system uses pressurized CO_2, miniature nozzles, and 12 electronically actuated micro-solenoid thruster valves. The satellite's position and attitude are determined using INS, which consist of three single axis accelerometers and three single axis rate gyroscopes, as well as 24 microphones that detect the arrival time of ultrasonic (U/S) pings from up to nine transmitting beacons. Five of these beacons (Fig. 3-2b) are mounted on the walls of the laboratory at MIT and in the U.S. Laboratory on ISS. They define the test volume and relate the attitude and position of each satellite to the ISS body frame. A single beacon is mounted on the surface of each satellite, bringing the total number of beacons in the ISS up to eight (one of the beacon address is not being used in the ISS). Appendix B provides detailed hardware specifications for the gyroscopes, accelerometers and U/S sensors. Two radio frequency (RF) channels provide independent inter-satellite communica-

[2]The origin of the satellite's body frame is at its geometric center. The body axes are perpendicular to each square face on the satellite. They are oriented such that the x-, y- and z-body axes point in the direction of the +x, +y and +z face, respectively.

tions and satellite-to-laptop communications. The main microprocessor is a Texas Instruments C6701 DSP processor. Each satellite is also equipped with an expansion port to interface with external payloads (e.g., an external docking mechanism [106]). Now, the subsystems of particular relevance to the GN&C algorithms are described.

3.1.2 Relevant hardware subsystems

Among all SPHERES subsystems, three are especially relevant to the implementation of GN&C algorithms on hardware for docking: the navigation, the propulsion and the docking subsystems. They provide the satellites with the means to sense their states throughout the operational volume, as well as to move in all six DOF and capture another satellite. This section describes them in detail.

The navigation subsystem

An innovative position and attitude determination system, based on U/S transmission, has been developed for the SPHERES testbed. The purpose of the system is to provide each satellite with an estimation of its position, velocity, attitude and angular rate. Like GPS [121], this system uses time-of-flight (TOF) data of signals emitted from transmitters at known locations to receivers on the satellites to estimate that satellite's states (Fig. 3-6). More precisely, the basic measurement is the time that a U/S signal takes to travel from a beacon (transmitter, Fig. 3-2b) mounted at a known location on the laboratory wall, to a microphone (receiver) located on the satellite (Fig. 3-5).[3] Given that there are five beacons mounted on the walls and 24 microphones on each satellite, there is a potential of 120 TOF measurements per satellite per measurement event. This data, combined with data from three gyroscopes, is processed using the navigation software module to compute a six DOF state solution. Many other spacecraft, including ETS-VII [65], DART [111] and the Mini AERCam [43], used a similar approach to compute their position and attitude. The SPHERES U/S-based system provides a very precise location (on the order of a few millimeters)

[3]This TOF data is sometime referred to as *the distance matrix* or ranging data (Fig. 3-6), since it can be easily converted to range measurements using the speed of sound, as shown in Eq. (3.3).

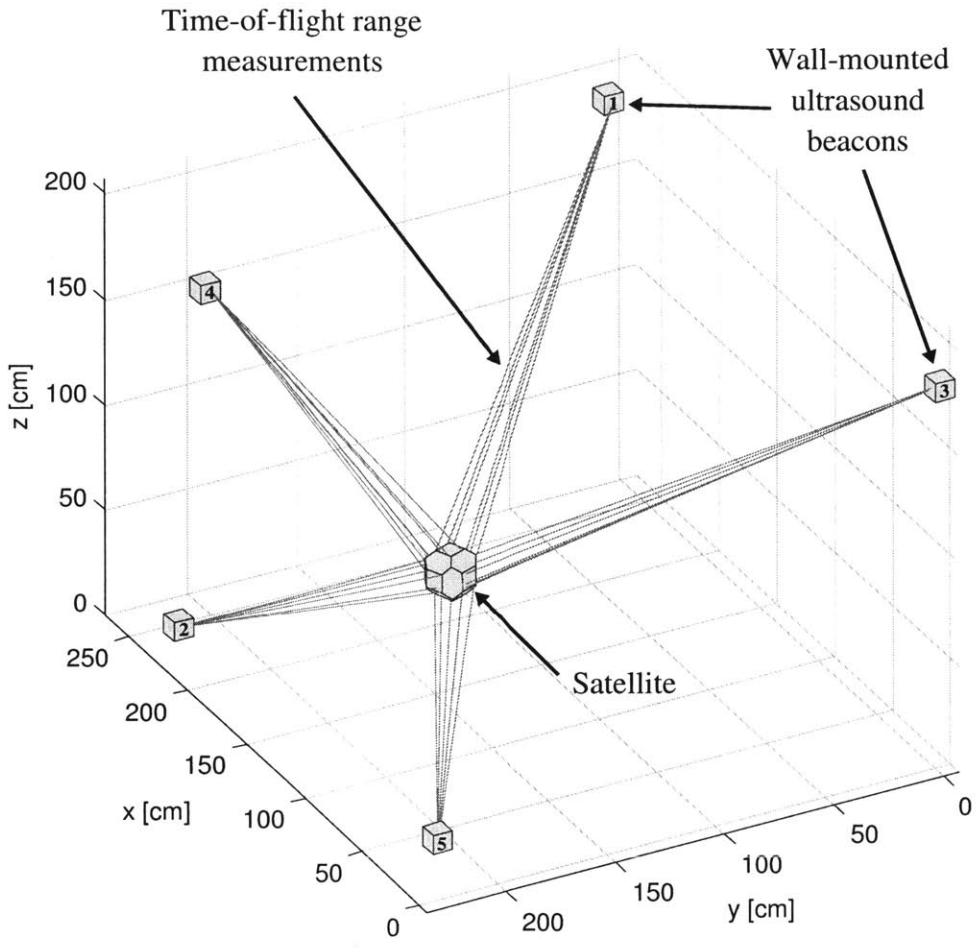


Figure 3-6: Data collected by the U/S sensor system during one sampling cycle [57].

and orientation (1-2 degrees) at a maximum update rate of 5 Hz. This is comparable to differential GPS.

When SPHERES was originally designed by a class of seniors at MIT [20, 112], a trade study was performed to analyze the most appropriate hardware combination for state determination aboard the ISS. A combination of U/S beacons (transmitters) and microphones (receivers) along with INS (gyroscopes and accelerometers) was selected because of:

- its low cost;
- its simplicity;
- its traceability to existing sensor suites (bandwidth and resolution similar to

differential GPS);

- its compactness relative to GPS receivers;
- previous experiment with U/S systems;
- the concern that actual GPS signals would not be detectable inside the ISS;
- the desire to minimize electromagnetic interference with other ISS equipment;
- the ability to make all relevant states observable;
- the presence of an Earth-like atmospheric condition inside the ISS.

These sensors are used to simulate a combined GPS-INS sensor suite providing position and attitude estimates, similar to commercially available sensor suites [46]. Moreover, a mix of absolute (U/S navigation system) and inertial sensors allows for absolute measurements (relative to the ISS body frame), long term accuracy and high bandwidth.

The method for estimating absolute position and attitude using the external beacons is as follows. Up to a total of nine beacons can be used at one time in the system. Table 3.1 and Fig. 3-7 illustrates the timing associated with the data collected following a beacon ping. When a satellite (the *leader*) requests an absolute state update, it emits an omnidirectional infrared (IR) flash at time 0 msec which is received by the surrounding satellites (the *followers*) and the external beacons. All of the satellites zero the timers on the receivers and listen for a U/S ping using each of their 24 U/S receivers located on their surface (Fig. 3-5). Since the IR flash travels at the speed of light, its contribution to the TOF measurement is essentially zero. The external beacons each emit a U/S ping in sequence (20 msec apart). The sequence is determined by the address of each beacon. In Table 3.1, the first column corresponds to an address that is entered into a beacon. No two beacons can share the same address. The second column corresponds to the time that a beacon, with that address, emits a U/S ping following the arrival of an IR flash. For example, Beacon #1 has a 10 msec delay between reception of the IR flash and transmission of its U/S ping.

Table 3.1: Timing corresponding to each beacon address

Beacon address	Pinging time* (msec)	TOF passed to the computer* (msec)
1	10	20
2	30	40
3	50	60
4	70	80
5	90	100
6	110	120
7	130	140
8	150	160
9	170	180

* with reference to the time the IR flash is transmitted

The delay is 30 msec for Beacon #2, etc. The times in the second and third column define a window, which corresponds to the time over which the microphones on the satellites listen for a U/S ping from a beacon with that address (Fig. 3-7). Once the time in the third column is reached, the system sends to the computer the arrival times of any U/S pings, or an array of zeros if nothing was heard. The 10 msec delay between a time in the third column and a time in the next row of the second column allows echos to dissipate. Therefore, the satellite uses the time delay after the IR flash to distinguish the beacon at the source of the ping, according to the recording windows illustrated in (Fig. 3-7). For example, a U/S signal received at time 75 msec is determined to have been transmitted by Beacon #4, which pings at 70 msec. The difference between the time delay and the pinging time of the corresponding beacon is the TOF data passed to that satellite's computer (5 msec in the example above).

In situations where the global estimator tends to be unstable and multi-path is suspected, the time delay allowed for the signal to dissipate can be increased to 30 msec by using every odd beacon addresses on the five external beacons (1, 3, 5, 7 and 9) as illustrated in Fig. 3-8. Any signal received in windows corresponding to even beacon addresses (2, 4, 6 and 8) is simply ignored by the software (illustrated by the grey lines in Fig. 3-8). This limits the total number of beacons to five instead of nine, preventing the use of the onboard beacons simultaneously with the external

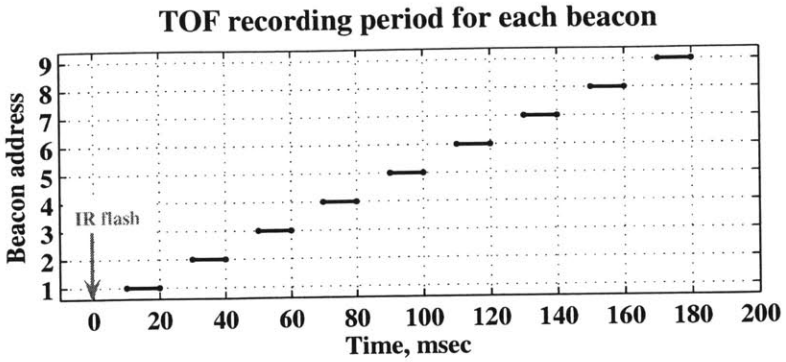


Figure 3-7: TOF recording periods for each of the nine beacons.

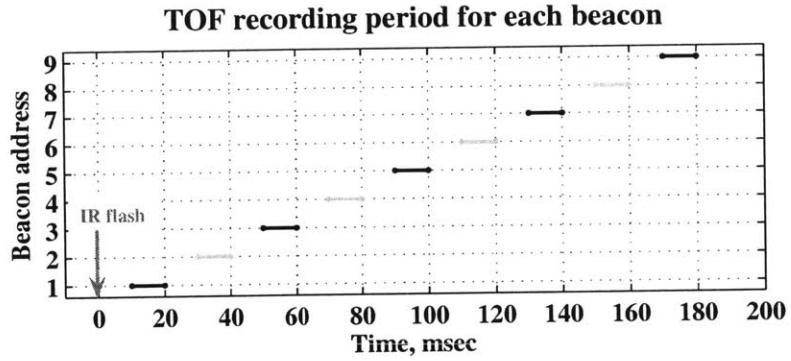


Figure 3-8: TOF recording periods with a reduced number of beacons.

beacons. This technique has proven to be effective in the laboratory when large metallic objects (e.g., cabinets) are surrounding the test area.

A satellite must receive at least three good ranges each from at least three beacons in order for the estimation algorithm to be able to update all of the state estimates (three positions, three velocities, four coupled quaternions and three angular rates). Because it can take up to 200 msec to cycle through all the beacons, a hardware limitation on the maximum global update frequency is set at 5 Hz. In addition to the five external beacons, each satellite is equipped with a beacon located on its docking face (Fig. 3-5). This beacon works exactly the same way as the external beacons. It can be used separately or in addition to the external beacons. Since the microphones are grouped in sets of four at the corners of each of the six square faces on the satellite, the four microphones facing a beacon on a neighboring satellite measures four TOF measurements that can be used to calculate two relative bearing angles and one range

to the beacon. Unfortunately, they do not provide a reasonable estimate of relative roll about the range vector.

Due to cross talk, a constraint has to be imposed on when the U/S system can be used. It was found that the noise produced by a thruster firing has a U/S component. Therefore, the programmer has to ensure that no thrusters are firing, when the U/S system is sampled, in order to avoid interference with the ping produced by the beacons. Figure 3-9 illustrates a typical thrust and estimation pattern. The control period is divided into a thrust window and a beacon sampling window. A thrust window is defined as a period of time when the thrusters are allowed to fire. Its duration is adjustable by the programmer. For the example shown in Fig. 3-9, because the thrust window covers only 20% of the control period, the maximum duty cycle is set at 20%. This is plenty for most maneuvers due to the relatively large thrust produced by the thrusters (0.09 N per thruster for a 4.3 kg satellite with a total of 12 thrusters). The satellite requests beacon pings during the beacon sampling windows (between thrust windows), with 200 msec between each consecutive ping cycle. Each vertical arrow in Fig. 3-9 represents the sampling of all beacons (up to nine). Therefore, this timeline shows four complete sampling cycles between each thrust window. If one constrains each sampling cycle to last 200 msec and conducts four such samplings between each thrust window, then the control period that allocates 20% of its time to the thrust window is one second in duration. Of course, this period is user-selectable, subject to these constraints. To speed up the control rate, one could interrogate fewer beacons, do so less often, and allocate less time for thrusting.

The INS onboard each satellite consist of three gyroscopes and three accelerometers, aligned with each body axis (Fig. 3-5). Their specifications are listed in Appendix B. They can be sampled at a much higher frequency than the U/S system (up to 1 kHz). Due to the presence of an internal resonance at 338 Hz, the gyroscopes are sampled at 1 kHz and pre-filtered using a low-pass discrete filter with a cutoff frequency set at 50 Hz (Appendix B). The navigation modules can then sub-sample the pre-filtered gyroscope data without aliasing the 338 Hz resonance into the lower

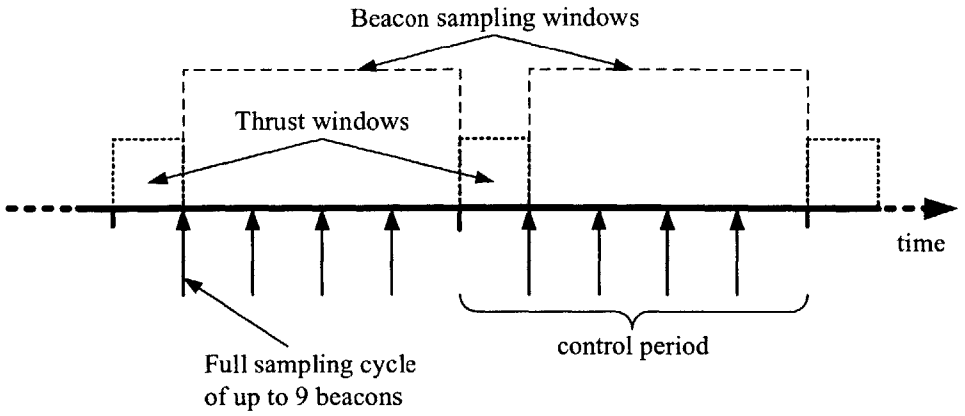


Figure 3-9: Typical thrust and estimation pattern used on the SPHERES testbed.

frequency data.

In the presence of gravity, a one degree inclination between the sensing axis and the horizontal axis is enough to saturate the DC, low-pass accelerometers. This makes them difficult to use in 1-g, as they are very sensitive to any tilt the satellite might have on the air bearing used at the MIT SSL 2-D laboratory. Therefore, they are yet to be integrated into the navigation modules covered in this section.

The propulsion subsystem

SPHERES uses a cold gas propulsion subsystem. The main reasons for the selection of CO_2 as a propellant are the possibility to store it in liquid form at room temperature at relatively low pressure (860 psi), and the fact that it is not toxic in small concentrations. Figure 3-10 shows the components of the propulsion subsystem. A regulator, accessible to the user, lowers the pressure to between 0 and 35 psi.[4] To maximize efficiency, the pressure has to be maintained higher than $\approx$12 psi so that the CO_2 flow chokes at the nozzle exit, the smallest cross section it encounters throughout the system. A capacitor ensures that the liquid CO_2 fully expands prior to entering the smaller distribution tubing, and helps reduce pressure fluctuations when opening a thruster. The ON/OFF valves allow the CO_2 to escape through the nozzles for a

[4]A hard stop is set at 35 psi for the satellites flying in ISS, although the regulator is originally designed to bring the pressure up to 55 psi.

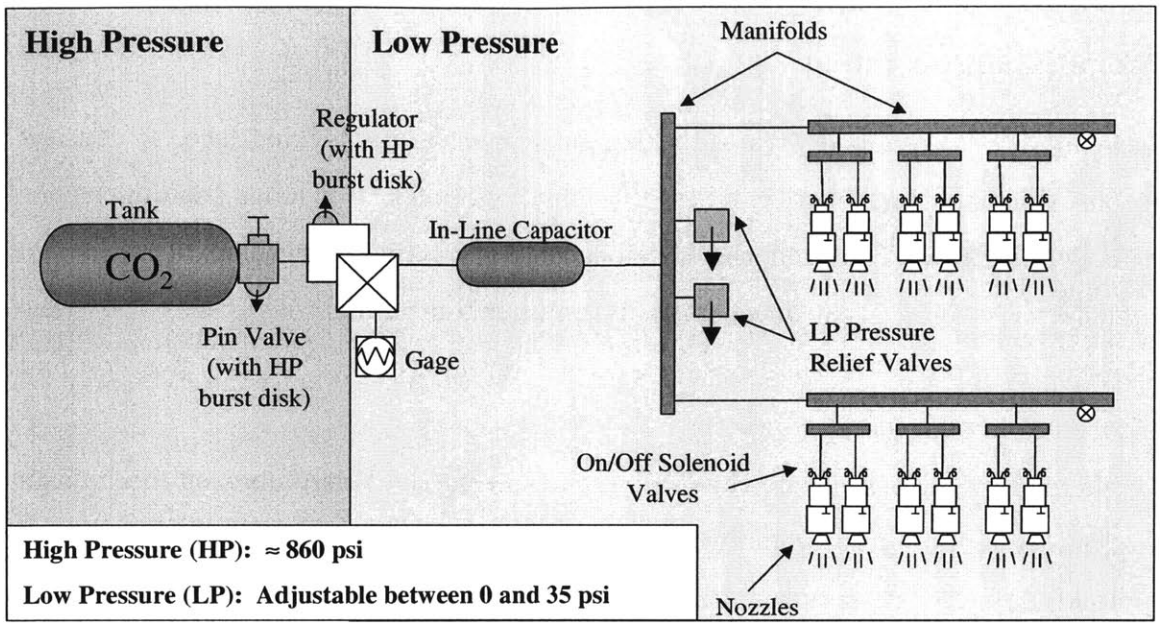


Figure 3-10: The SPHERES propulsion subsystem.

duration specified by the control law. Therefore, the thrust produced when a valve is opened is not adjustable. Appendix C provides the theoretical performance of the propulsion subsystem for typical pressure settings of 25, 35 and 55 psi.

The SPHERES thrusters are controlled using pulse-width modulation. They are commanded via a function linked to a high priority hardware interrupt. The user specifies the actual delays before sending the opening and closing signals (e.g., Thruster #1 opens in 100 msec and closes in 200 msec). A counter decrements these delays using a 1 msec clock. When the delay reaches 0, the corresponding signal is sent to open or close the thruster. Therefore, if a command is sent to open a thruster, the user can expect the signal to be sent within 1 msec and the thruster to be physically opened within 5-10 msec (typical opening delay). A lot of effort has been devoted into characterizing the SPHERES propulsion system [19, 12]. The results of that characterization have included simulation models and an extension of the pulse-width modulator presented in Section 3.5.2.

The docking subsystem

Each satellite is equipped with a docking subsystem on its docking face. It consists in four Velcro patches (two patches of hooks, two patches of loops) of dimensions 1 in $\times$ 1 in, located around the U/S beacon for relative navigation between two satellites (Fig. 3-5). The mechanism is universal, meaning that it has the same Velcro pattern on all the satellites. It tolerates errors up to ± 1.5 cm in position and ± 5 degrees in attitude at contact.

Microgravity experience in the ISS has shown that the Velcro docking mechanism tend not to stick properly when two satellites are making an aligned contact at a small approach velocity. Also, automatic undocking is not possible, as the crew is needed to separate the satellites once the Velcro has latched. The addition of an external docking mechanism using the expansion port connector on each satellite (located on the face opposite to the docking face in Fig. 3-5) could increase the reliability and the level of realism of the capture sequence at the end of the docking, and also enable an automatic release sequence [105, 107, 106].

3.1.3 Ground operations

Table 3.2 shows the development and ground operation schedule of the SPHERES testbed. The initial development occurred during a senior undergraduate class at MIT in 1999 [112]. Two prototype versions of the satellites were flown in the KC-135 between 2000 and 2002. The flight version of the satellites became available in 2002. After 2002, the flight satellites have been tested at the MIT SSL, in the KC-135 and on NASA MSFC flat floor.

3.1.4 Flight operations

Table 3.3 shows a summary of the ISS operations of the SPHERES testbed. The initial March 2003 launch of SPHERES to ISS was manifested on STS-116. After the Columbia accident in February 2003, the launch slipped. The next opportunity to launch a satellite was in November 2003 on Progress 13P, but the launch was

Table 3.2: SPHERES ground operations time table

Date	**Location**	**Description**
Fall '99	MIT SSL	Start of system development followed by the construction of two prototype satellites
Feb. '00	KC-135	First μ-g test with two prototype satellites (attitude tracking)
Mar. '00	KC-135	Second μ-g test with two prototype satellites
Nov. '01	KC-135	Third μ-g test with two prototype satellites
Spring '02	Payload Systems	Development of five flight satellites
Aug. '02	KC-135	First μ-g test with two flight satellites; μ-g docking tests
Fall '02	MIT-SSL	Development of the Guest Scientist Program (GSP)
Feb. '03	KC-135	Second μ-g test with two flight satellites; test of mass identification algorithms
Spring '03	MIT SSL	Development of the follower and docking control algorithms
Nov. '03	KC-135	Third μ-g test with two flight satellites; 3-D attitude control, beacon search pattern, docking tests
Jun. '04	NASA MSFC	Flat floor demonstration of an autonomous docking maneuver with a rotating target
Dec. '04	NASA MSFC	Flat floor demonstration of tethered formation flight
Spring '05	MIT SSL	Development of the Universal Docking Port
Summer '05	MIT SSL	Development of the Miniature Video Docking Sensor
Jan. '06	C-9B	μ-g test of the MOSR experiment
Jul. '06	NASA MSFC	Flat floor demonstration of spacecraft assembly and reconfiguration for the SWARM project
Sep. '06	NASA MSFC	Flat floor demonstration of tethered formation flight and space assembly for SWARM

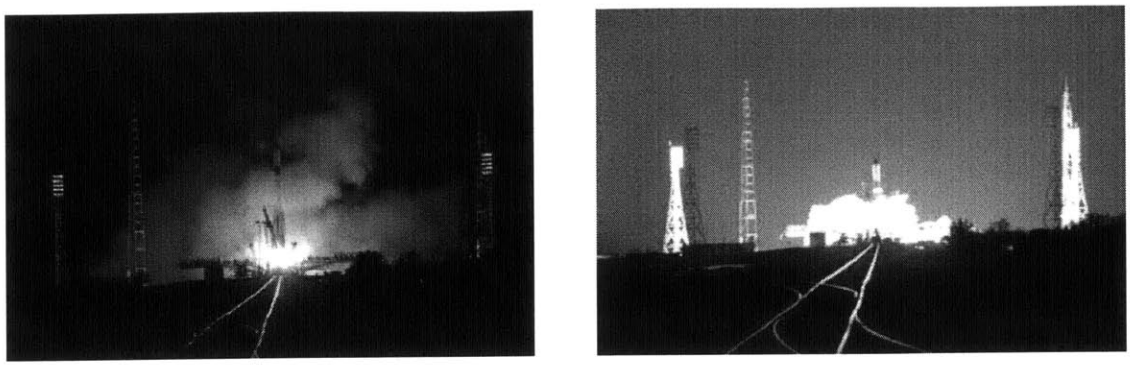


Figure 3-11: Launch of Progress 21P, marking the launch of the first SPHERES satellite.

postponed. After being manifested numerous times on different Progress flights, the first satellite was finally launched on April 24, 2006 on Progress 21P (Fig. 3-11). The second satellite was launched on July 4th, 2006 via the Space Shuttle on mission STS-121. The third satellite joined the fleet on December 9, 2006 on STS-116.

A brief description regarding test session preparation and operation follows. The MIT SPHERES team gets notified of an upcoming ISS test session typically two weeks beforehand. The software, which has already been elaborated, is integrated and verified on the SPHERES tesbed on the MIT SSL 2-D air table. It is sent to the DoD Space Test Programs office one week before the test session, for upload to the ISS. During the test session, the MIT SPHERES team monitors the experiments in real time (video and audio), using a high bandwidth internet connection with NASA. Figure 3-12 shows the typical testing environment in the ISS. Testing generally occurs in the U.S. Laboratory segment close to the hatch. The beacons are mounted on the walls around the test volume. The view displayed in Figure 3-12 is representative of the view sent to MIT via the internet video feed. During live operations, instructions are communicated, upon request, to the astronaut performing the experiments. Telemetry is downloaded and stored on the ISS laptop. Approximately three days after the test session, the telemetry files are emailed to the MIT SPHERES team for analysis. The video recording of the experiments follows a few weeks later. Such test sessions are conducted approximately every month (Table 3.3), thus giving the MIT SPHERES team the opportunity to refine their algorithms prior to the next one.

Table 3.3: SPHERES ISS operations time table

Date	Location	Description
Aug. '03	Progress 12P	Launch of test equipment (one beacon, one beacon-tester)
Aug. '03	JSC	Delivery of two flight satellites, five beacons and consumables for up to two months
Aug. '04	ISS	Performed beacon and beacon-tester experiments (Astronaut Mike Foale, Expedition 8)
Apr. '06	Progress 21P	Launch of one satellite, one beacon, one laptop transmitter, seven tanks, 10 battery packs
May '06	ISS	ISS Test Sessions 01 and 02. Single satellite operations. Hardware checkout, open- and closed-loop maneuvering, FDI tests (Astronaut Jeff Williams, Expedition 13)
Jul. '06	STS-121	Launch of one satellite, four beacons, one laptop transmitter, five tanks, six battery packs
Aug. '06	ISS	ISS Test Sessions 03 and 04. One and two satellite operations. Complex closed-loop tracking, docking, mass properties identification (Astronaut Jeff Williams, Expedition 13)
Nov. '06	ISS	ISS Test Session 05. One and two satellite operations. Mass properties identification, safe docking, docking to a tumbling target (Astronaut Michael Lopez-Alegria, Expedition 14)
Dec. '06	STS-116	Launch of one satellite and the remaining batteries and tanks (for a total of 48 batteries and 96 tanks on orbit)
Mar. '07	ISS	ISS Test Sessions 06 and 07. One, two and three satellite operations. Docking with faults, safe docking to a tumbling target, mass property identification, circular formation flight (Astronaut Sunita Williams, Expedition 14)
Apr. '07	ISS	ISS Test Session 08. Two and three satellite operations. Space assembly, reconfiguration, circular formation flight, collision avoidance (Astronaut Sunita Williams, Expedition 14)

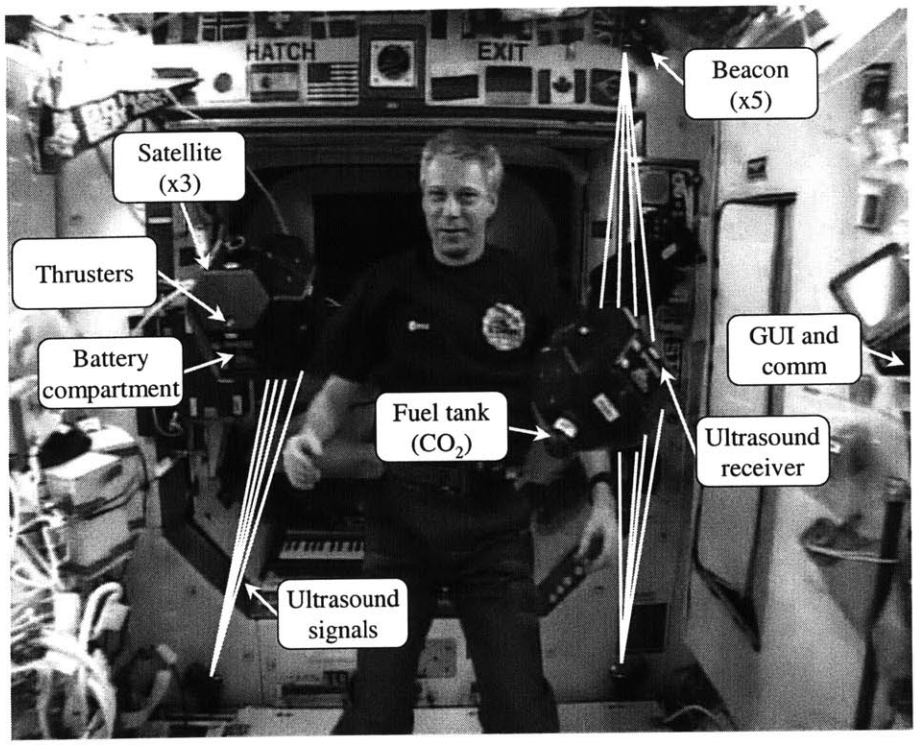


Figure 3-12: Typical SPHERES setup in the U.S. Laboratory on ISS (courtesy of ESA).

Having provided an overview of the SPHERES testbed, the next section focuses on software tools developed by the MIT SSL to support the implementation of the algorithms.

3.2 The Guest Scientist Program

A main feature of SPHERES is the capability to conduct a variety of different research efforts simultaneously. By allowing multiple guest scientists to access the facility, the cost per scientist can be greatly reduced. The SPHERES project has a Guest Scientist Program (GSP) that provides a software interface with the hardware and a series of standard library functions for direct access to both the sensors and actuators onboard the satellites. It also encompasses the process by which the guest scientists implement their software and interact with MIT to verify it prior to flight.

Figure 3-13 presents a program overview. The key elements of the GSP are the

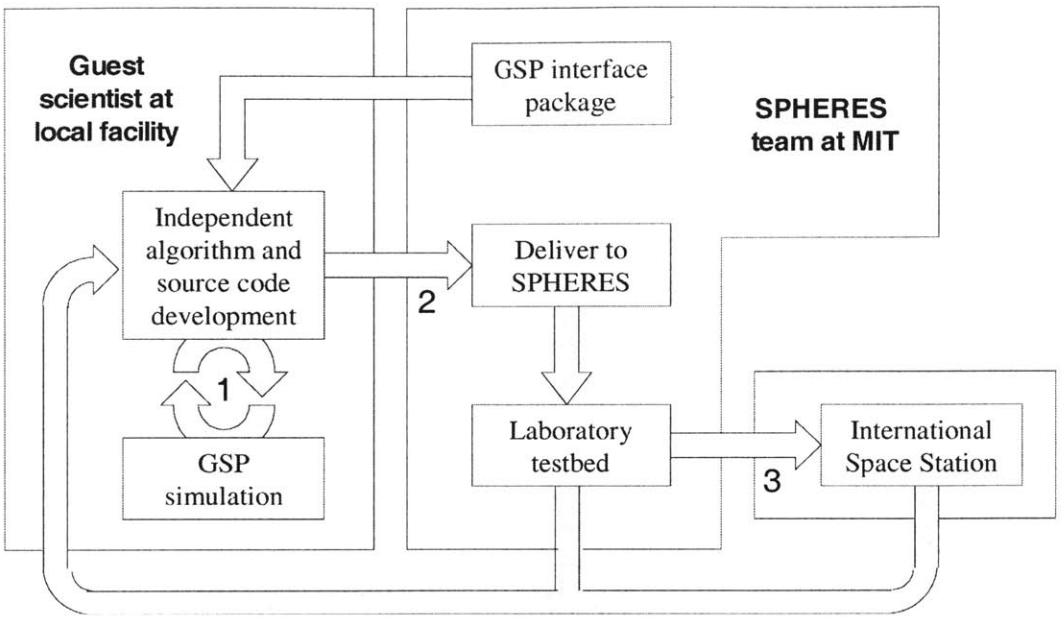


Figure 3-13: GSP information flow diagram [38].

SPHERES simulators, the 2-D ground laboratory located at the MIT SSL, and the 3-D laboratory onboard the ISS [38, 57, 58]. The GSP program consists of three development steps (Fig. 3-13):

1. The GSP simulator provides an initial software tool to help in the development of algorithms and to simulate the real-time software that runs on the SPHERES testbed.

2. After the software is successfully verified on the simulator, the scientists provide the code to the MIT SSL for testing on the 2-D ground laboratory.

3. Once the basic operation of the code has been successfully demonstrated, it is then qualified for testing in the ISS.

The following subsections provide more details about the different segments of the GSP.

3.2.1 Software interface on SPHERES

Each satellite is equipped with a digital signal processor for processing various commands and algorithms. The real-time operating system (OS) executes different segments of code during discrete time intervals. The processing of a particular segment is triggered by an interrupt, a signal to the core of the OS that an event has occurred. These interrupts can be hardware-based (e.g., clock ticks) or software-based (e.g., triggered by a program). SPHERES is composed of many interrupts with a pre-defined level of priority to resolve conflicts [113].

The software interface with the SPHERES hardware is composed of two main parts: a library of standard interface functions (often called the *SpheresCore* library) and a series of primary interface functions accessible by the user (*gspIdentitySet, gspInitProgram, gspInitTest, gspInitTask, gspPadsInertial, gspPadsGlobal, gspControl, gspTaskRun*). More details on these functions can be found in Ref. [58]. Each of these primary interface functions is directly linked to a programmable interrupt on the satellite, providing flexibility for the implementation of different GN&C architectures. A user can directly link software modules to each primary interface function, allowing them to be executed on the real-time OS. For example, the function *gspControl* is linked to an interrupt that occurs typically at the control period, which is defined by the user. A control algorithm inserted in that function would therefore be executed at the control period.

The different types of generic processes allowed by this software interface are shown in Fig. 3-14. Typically, initialization processes are the ones that occur once, either at bootup, when starting an experiment, or when switching between GN&C modes (Section 3.3.2). Other processes can be run at regular intervals or after an event. High priority processes are usually tied to a hardware interrupt. However, to avoid processor overload, high priority processes are restricted to 1 msec of computation time on SPHERES. Therefore, they can be run up to 1 kHz (e.g., sampling of the gyroscopes). Low priority processes are given more processor time. They are usually tied to software interrupts. However, these processes are interrupted by higher priority

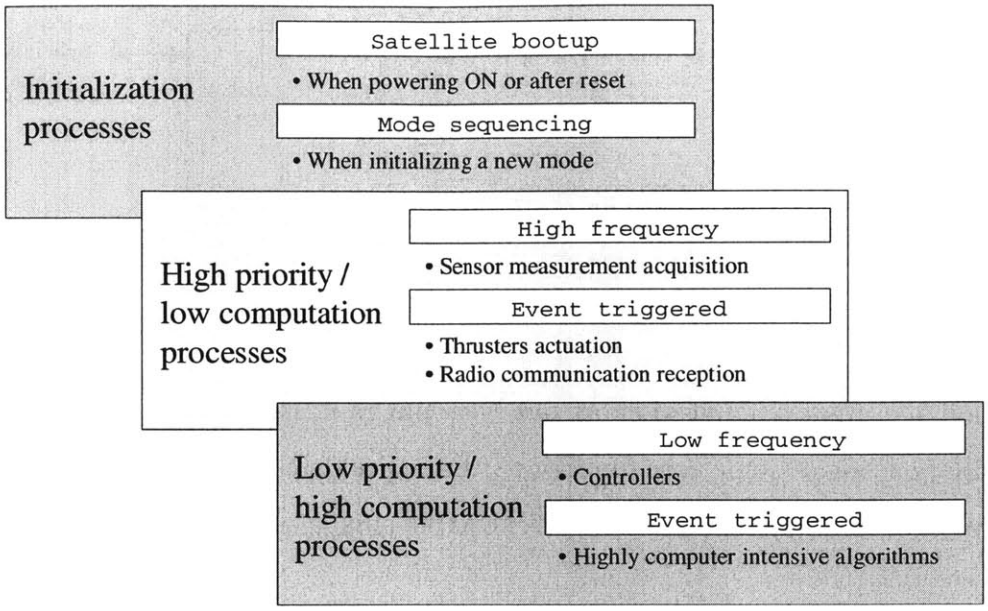


Figure 3-14: Different levels of computational processes.

processes whenever they occur. Typically, they run at a rate of 20 Hz and below (e.g., the controllers).

The SPHERES computer is mainly programmed in the C programming language, although C++ code can also be compiled. There exist numerous books providing direct C translation of mathematical algorithms [98]. However, software development is usually performed in a language that offers a wide range of visualization tools (e.g., MATLAB®). Therefore, two simulators are provided as part of the GSP: a C simulation and a MATLAB® simulation.

3.2.2 The C simulation

The C simulation was designed to emulate, on a desktop computer, operation of the entire SPHERES testbed. The idea behind it was to allow a user to compile a script written in C, run it on the simulation, compile the exact same script for SPHERES, run it on the testbed and compare the results. A graphical user interface (GUI) allows instantiation of different parameters, like the presence or absence of gravity, the number of satellites in the experiment, the beacon locations and the initial location of each satellite. This simulation is intended to facilitate software development by

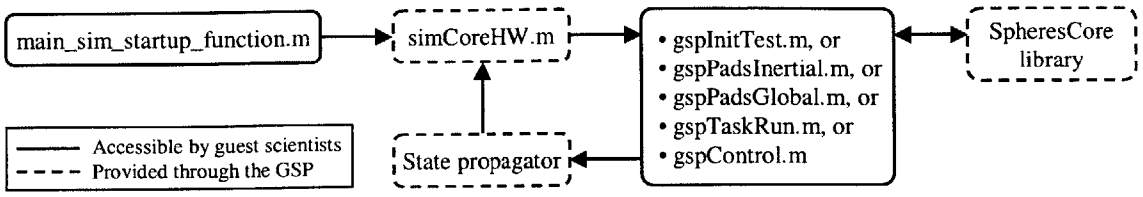


Figure 3-15: Information flow in the MATLAB® simulation.

guest scientists who do not have direct access to the satellites.

Although a great amount of effort has been put in it, the C simulation has never been fully functional (some core functions still require debugging). While waiting for its completion, a simple but realistic MATLAB® simulation has been developed.

3.2.3 The MATLAB® simulation

One of the contributions of this research has been the creation of simulation tools to support the implementation of GN&C algorithms for on-orbit formation flight and autonomous docking experiments. These simulation tools are embedded in a MATLAB® simulation provided as part of the GSP.

The SPHERES MATLAB® simulation was formed from the combination of the following:

- a MATLAB® translation of the standard GN&C algorithm library in *SpheresCore*;
- a simple state propagator;
- a series of calibrated actuator and sensor models;
- a high level software architecture similar to the one used on SPHERES.

Figure 3-15 shows the flow of information within the MATLAB® simulation. A simulation engine (*simCoreHW.m*) replicates the interrupts that occur in the hardware, executes the different primary interface functions in the same order as they occur on the satellites, and propagates the states of the satellites between the interrupts. This creates a realistic simulation of the dynamic response of the system.

The hardware models used in the simulation are based on data acquired from calibration of the SPHERES hardware (e.g., thruster strength, sensor resolution, inertia properties). The sensor models include realistic sensor noise for the gyroscopes and U/S sensors, and calibrated biases and blackouts[5] for the U/S sensors. The thruster models [19, 12] include thruster strength for each thruster, reduction of thrust when multiple thrusters are opened simultaneously, calibrated thrust at different pressure levels and realistic thruster actuation noise (5% of the thruster strength). The thruster opening and closing delays[6] are not currently taken into account.

The sequence of primary interface functions, processed by the simulation engine, is based on the IR flash pattern programmed by the user. As part of the function initializing the simulation (*main_sim_startup_function.m*), the user specifies each different thrust-estimation pattern used in the experiment (e.g., Fig. 3-9). The simulation engine determines a timeline for the execution of each primary interface function based on that information. Other initialization parameters include the test number to be run, the total number of satellites, a simulation timeout, the serial number of each satellite, the true initial states of each satellite and other global variables proper to the experiment. The *true states* of each satellite are propagated between the execution of each primary interface function. Communication between the satellites is performed via the help of global variables (the communication delays and losses[7] are not currently modeled).

To facilitate the development of the simulation, some key assumptions had to be made:

- A control cycle is divided in two segments:
  - a thrust window that occurs first;
  - a beacon sampling window that follows.

[5]Approximately 3% of the time, the U/S sensors do not record an incoming U/S pulse or return bad data, possibly the result of echoes.

[6]The typical thruster opening delay is on the order of 5 msec, while its closing delay is on the order of 2 msec [12].

[7]It is known that the performance of the communication system varies between tests because of a glitch in the initialization process, and because of the relative orientation between the antennas in the satellites and the one on the ISS laptop.

- The only event that can occur in a thrust window is the reading of the gyroscopes (which is automatically generated);
  - IR flashes and the execution of event triggered processes[8] cannot occur in a thrust window, although no warning messages are generated if the user still does so.
- The same schedule of events is followed by all the satellites (e.g., IR flashes, tasks and sensor readings). An example of a sequence of sensor reading events is shown in Fig. 3-17;
  - tasks are also currently time-based, not event-based.[9]
- Computation delays are not currently modeled.
- The termination of the simulation is triggered by setting the global variable *ctrlTestTerminate* on Satellite-1 (leader) to one.

These assumptions might in some cases affect the desired flexibility of the simulation. Modifications are expected to occur as more complex simulations are developed.

This concludes the section on the GSP and the different software tools made available for the implementation of GN&C algorithms on SPHERES. The next sections cover the architecture, the main subsystems and the implementation of the actual GN&C algorithms.

3.3 A GN&C architecture for autonomous docking

This section gives a high-level overview of the GN&C architecture used on the SPHERES testbed [87]. This architecture forms the basis for a multitude of autonomous docking and formation flight experiments. The different GN&C modules comprising the architecture are briefly introduced as well. These modules

[8]Event triggered processes include any process that is run in *gspTaskRun*, with the exception of the state estimator.

[9]The current implementation of the MATLAB® simulation requires the timing of the events to be explicitly specified by the user.

are designed to be integrated into a software architecture that enables autonomous docking on the SPHERES testbed. The following sections describe in more detail the software modules that comprise this architecture.

3.3.1 GN&C architecture overview

One of the key characteristics of SPHERES is its modularity in the algorithm development. Many functions are needed to operate SPHERES. By modularizing the software, the user can focus on the development of one function, while borrowing existing modules for the other functions [38, 113]. Figure 3-16 shows a hierarchical view of the architecture. It is based on Fehse's review of automated docking methodologies (Section 1.4). In Fig. 3-16, the components common to Fehse's architecture (Fig. 1-9) are greyed. The box labeled *autonomous onboard docking control systems* contains the software modules that run on a satellite. Each software module is an implementation of an algorithm accomplishing a certain task. External interfaces include the plant (satellite), the operators and the other satellite. The software modules are grouped into FDIR, solver, MVM and GN&C. To date, the SPHERES team has focused on the implementation of GN&C modules [86, 87]. However, the team is now working on the implementation of MVM, FDIR and solver modules. Some of them are discussed in this chapter.

Since methodologies in Ref. [40] assume some level of real-time human intervention, some extensions (shown in black in Fig. 3-16) were necessary to enable autonomous docking. The extended architecture accounts for the idea that communications with human operators may be intermittent or non-existent (illustrated by dashed lines), forcing the designers to integrate fault detection, isolation and recovery (FDIR) capability at all levels [37, 61]. In this extended model, the onboard computer is granted the authority, through the main FDIR module, to trigger a collision avoidance maneuver (CAM) if a problem is detected. Furthermore, a solver module allows for online planning and scheduling. It is also monitored by the FDIR algorithm to prevent infeasible solutions or other failures. Since knowledge of the target's states is required, remote sensing and/or communications between the satellites are needed

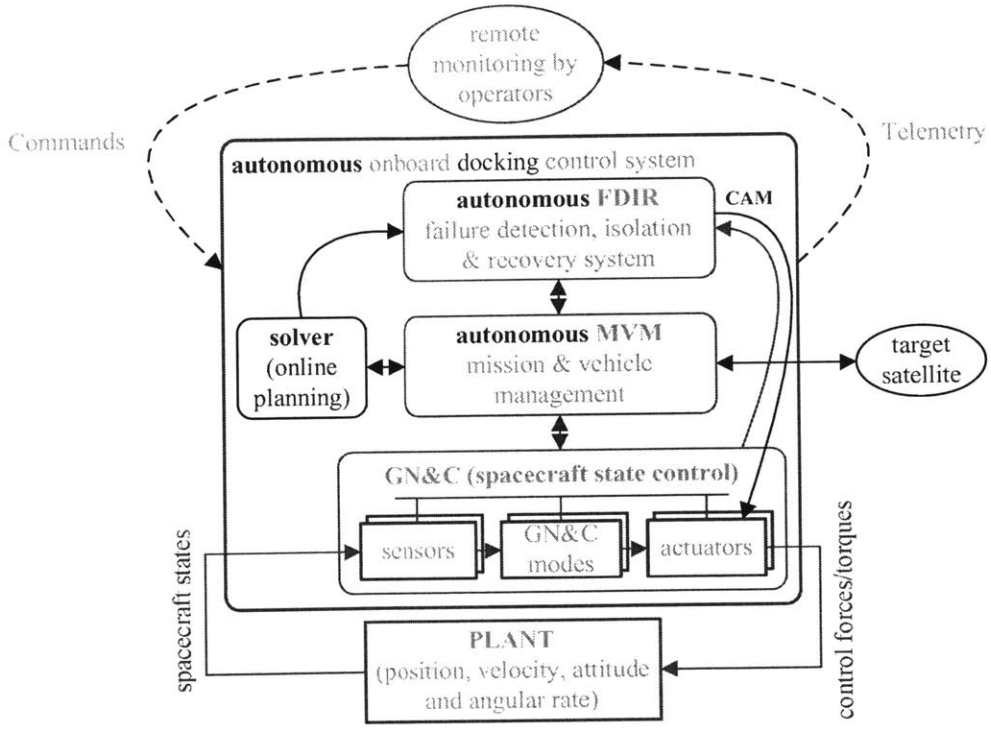


Figure 3-16: A hierarchical view of the GN&C architecture used on the SPHERES testbed [87].

(Section 1.5).

3.3.2 Software module categorization

The following subsections briefly describe the different categories of software modules that currently exist. Most of the modules are at the GN&C level, although few are at the MVM and FDIR level.

Navigation modules

The most complex modules currently implemented are the navigation modules that allow each satellite to estimate its position, velocity, attitude and angular rate [85]. The inputs to the modules are the times that U/S pings take to travel from beacons, at known locations, to receivers on each satellite (Fig. 3-12). The navigation modules combine that information with data from onboard gyroscopes to compute the state

vector via the use of an EKF (Section 2.1.1).

Control modules

The control modules use the estimated state vector to compute thruster pulses that allow the satellite to maneuver [86]. Two sub-modules are typically used. The first implements the algorithm which uses the estimated state vector to compute forces and torques to be applied to the satellite. The second converts these forces and torques to thruster pulse durations using a pulse-width modulation scheme. Examples of the first sub-module include PID-type and phase plane controllers (Section 2.2). PID-type controllers, combined with pulse-width modulators, are used to regulate the satellite or to closely track a trajectory (both position and attitude). Phase plane controllers are available when large displacements or angular changes are necessary.

Path planning modules

The path planning modules are used to provide the trajectory to follow to accomplish the mission objectives. Only the glideslope controller, which is a hybrid between a path planning and a control module, is currently available for real-time operation. It is implemented at the GN&C level in Fig. 3-16. It is used to plan and control an approach trajectory for docking maneuvers (Section 2.3.3). Because of the lack of a solver module, advanced path planning algorithms are currently executed offline. The trajectory they output is hard coded in the software uploaded to the satellite.

MVM module

To perform a complex maneuver autonomously, it is highly desirable to divide it into small tractable steps to achieve the mission objectives. Numerous methodologies have been developed to autonomously progress through these steps [61]. On SPHERES, the MVM module (*gspControl*) is the one that manages the sequence of steps to be performed. Each of these steps is characterized by a GN&C mode, which dictates the GN&C modules to use, their different sets of gains and parameters, and the objective to meet prior to move to the next step, until the overall mission objective is achieved.

FDIR modules

A robust GN&C architecture has to include FDIR capability at all levels, from the hardware level to the MVM level [37, 61]. Two FDIR modules have been implemented on SPHERES. A thruster fault identification module, using gyroscopes and accelerometers, was developed by Wilson [131]. Although tested as a stand-alone module on the hardware, it has not yet been integrated into the GN&C architecture. The second existing FDIR module detects U/S measurement errors caused by multi-path when two satellites are in close proximity [85].

The remaining sections of this chapter describe the implementation of modules in each of these categories. It is important to note that not all algorithms presented in Chapter 2 have been implemented for real-time operations. The main criteria behind the selection of which algorithms to implement were the computational requirement and the expected ease of implementation.

3.4 Implementation of estimation algorithms

Two navigation modules were initially implemented. Both consist of a pre-filter, which selects the best TOF measurements from the 120 available, and an EKF that processes the selected measurements to provide state estimates. The global estimator provides estimates relative to the laboratory frame, using the external beacons, while the relative estimator provides relative estimates, using the single beacon onboard a neighboring satellite.

Because of the unavailability of part of the SPHERES hardware for the first three test sessions in the ISS, another variant of the relative estimator (the single beacon estimator) was designed and implemented. It allowed incremental tests and a valuable gain of microgravity test experience in preparation for the arrival of the remaining SPHERES hardware to the ISS. It also uses data from only one beacon to compute an approximation of the state vector. It integrates gyroscope data to compute attitude changes, and samples a single U/S beacon to obtain displacement and velocity estimates.

The following subsections present the navigation software modules developed to compute state estimates using different sensor suites. Two EKFs were implemented: one for the global estimator and one for the relative estimator. Both are covered in detail in this section.

3.4.1 The global state estimator

The global state estimator was the first navigation module to be developed. Its purpose is to provide the satellite with an estimate of its absolute navigation states (position, velocity, attitude and angular rate) with respect to a coordinate frame attached to the ISS. The inputs to the global estimator are the signals from the five external U/S beacons as well as the three gyroscopes. To save computation time, the EKF updates the state estimates every time a set of measurements is taken (either an array of TOF data corresponding to a single beacon or an array of readings of the three gyroscopes) as opposed to the batch processing of the measurements from all five external beacons and three gyroscopes. It takes anywhere from 15 to 25 msec to process external beacon measurements and approximately 1 msec to process gyroscope measurements. Since both the propagation of the attitude states and the computation of an absolute position, based on TOF measurements, are nonlinear, nonlinear filtering methods have to be used [16].

The EKF state estimation process is illustrated in Fig. 3-17. This represents a close-up view of one of the full sampling cycles (one vertical arrow) shown in Fig. 3-9. The arrows below the timeline illustrate when the U/S beacons ping, as well as when the sensor measurements (both U/S TOF and gyroscope data) are received by the computer. The arrows above the timeline show the state propagation and the measurement update phases of the EKF alternating, whenever measurements are received. It can be seen that the time interval Δt between two measurement updates (also the propagation time) is not constant.

The six DOF motion of each satellite is described in a state vector containing the three translational DOFs (position $\mathbf{r}$ and velocity $\mathbf{v}$) as well as the three rotational

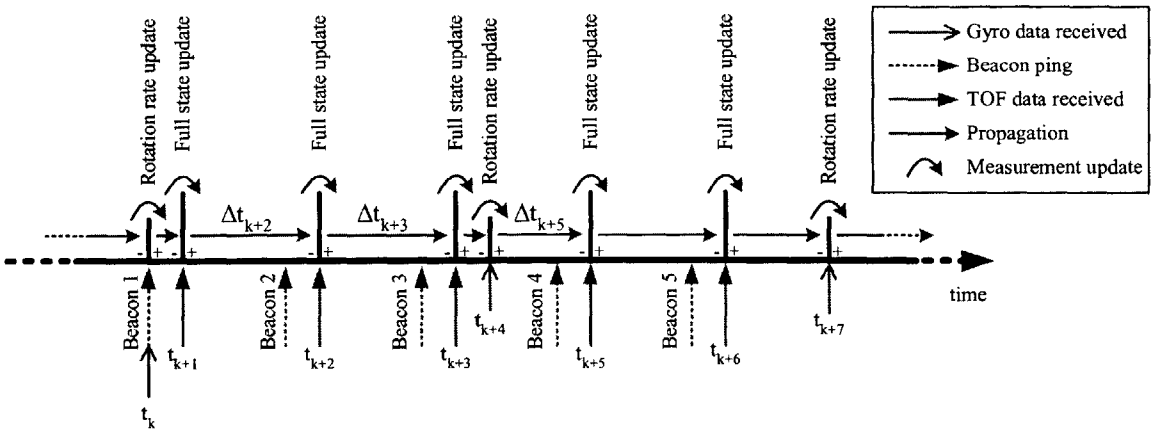


Figure 3-17: Measurement updates during a full sampling cycle of the 5 beacons.

DOFs (four element quaternions $\mathbf{q}$ and angular rate $\boldsymbol{\omega}$):

$$\mathbf{x} = [r_x \;\; r_y \;\; r_z \;\; v_x \;\; v_y \;\; v_z \;\; q_1 \;\; q_2 \;\; q_3 \;\; q_4 \;\; \omega_x \;\; \omega_y \;\; \omega_z]^T \tag{3.1}$$

The state vector used in Section 3.4.1 has $l = 13$ elements. The position and velocity components are expressed in a coordinate frame attached to the ISS or the ground laboratory (global frame), while the attitude and the angular rates are expressed in the satellite's body frame (Fig. 3-5).

The following convention is used to express the attitude using the quaternions throughout this thesis. The attitude of a satellite is defined as a rotation from a reference frame (the global frame in this section) to the satellite's body frame. This rotation is fully described by a rotation (of an amount θ_e) around a unit vector $[n_x \;\; n_y \;\; n_z]$ fixed to the satellite's body frame and stationary in the reference frame (Euler axis) [126]. Therefore, the attitude is described in terms of the quaternions as:

$$\begin{aligned}\mathbf{q} &= [q_1 \;\; q_2 \;\; q_3 \;\; q_4]^T \\ &= \left[n_x \sin\left(\frac{\theta_e}{2}\right) \quad n_y \sin\left(\frac{\theta_e}{2}\right) \quad n_z \sin\left(\frac{\theta_e}{2}\right) \quad \cos\left(\frac{\theta_e}{2}\right)\right]^T\end{aligned} \tag{3.2}$$

This four-element attitude quaternion is non-singular, contains only one redundant parameter, is easily normalized, and has simple rules for successive rotations, therefore

making it a good choice to express the attitude of the SPHERES satellites [57]. The remaining of this subsection shows the implementation of the general EKF equations shown in Section 2.1.1 for the global estimator.

Measurement update equations

This subsection derives the equations necessary to update the state estimates when the U/S beacons are sampled (the sampling of the gyroscopes is treated later in this section). The expected measurement vector $\mathbf{h}_k$, based on the state estimates just prior to the update ($\hat{\mathbf{x}}_k^{(-)}$), is derived below. Knowing the temperature T inside the ISS (which directly provides the speed of sound), the TOF measurement vector $\boldsymbol{\tau}_k$, collected after each beacon ping, can be easily converted to a vector of range measurements $\mathbf{z}_k$, also referred to as *the distance matrix*:

$$\mathbf{z}_k = \sqrt{\alpha R T} \cdot \boldsymbol{\tau}_k \tag{3.3}$$

with the adiabatic index α=1.402 and the ideal gas constant R=287.05 J/(kg·K) for air, as well as a room temperature of T=295 K. The vector $\mathbf{h}_k$ represents the expected distance corresponding to the TOF data collected by each of the 24 U/S receivers given the state estimates. For each measurement i, the corresponding component of the $\mathbf{h}_k$ vector can be expressed as the following L2-norm:

$$[\mathbf{h}_k]_i = \left|\left|\boldsymbol{\rho}_i^{ISS} - \boldsymbol{\varrho}_i^{ISS}\right|\right| \tag{3.4}$$

where $\boldsymbol{\rho}_i^{ISS}$ is the corresponding receiver (microphone) position vector and $\boldsymbol{\varrho}_i^{ISS}$ is the corresponding transmitter (beacon) position vector relative to the origin of the global frame defined in Fig. 3-6. The position of a receiver can be converted from the body frame of the satellite to the global frame using the following:

$$\boldsymbol{\rho}_i^{ISS} = \boldsymbol{\theta}_k^{(-)^T} \boldsymbol{\rho}_i^{body} + \hat{\mathbf{r}}_k^{(-)} \tag{3.5}$$

where $\hat{\mathbf{r}}_k^{(-)}$ is the position component of the state vector and the rotation matrix $\boldsymbol{\theta}_k^{(-)^T} = \boldsymbol{\theta}^T(\mathbf{q})\Big|_{\mathbf{q}=\hat{\mathbf{q}}_k^{(-)}}$ is the transpose of the attitude matrix $\boldsymbol{\theta}(\mathbf{q})$, which takes the following general form: [126]

$$\boldsymbol{\theta}(\mathbf{q})^{ISS\rightarrow body} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{3.6}$$

using the quaternion components of the state vector prior to the update.

From the combination Eqs. (3.4) to (3.6), the following analytical expression for Eq. (2.5) at time t_k for the i^{th} measurement can be derived:

$$[\mathbf{H}_k]_{ij} = \begin{cases} \dfrac{\left(\left[\boldsymbol{\theta}_k^{(-)^T}\boldsymbol{\rho}_i^{body}\right]_j + [\hat{\mathbf{r}}_k]_j^{(-)} - [\boldsymbol{\varrho}]_j\right)}{[\mathbf{h}_k]_i} & , j = 1:3 \\ \dfrac{\left(\boldsymbol{\theta}_k^{(-)^T}\boldsymbol{\rho}_i^{body} + \hat{\mathbf{r}}_k^{(-)} - \boldsymbol{\varrho}\right)^T \cdot \left[\frac{\partial \boldsymbol{\theta}_k^{(-)^T}}{\partial [\mathbf{q}]_{(j-6)}} \cdot \boldsymbol{\rho}_i^{body}\right]}{[\mathbf{h}_k]_i} & , j = 7:10 \\ 0 & , \text{otherwise} \end{cases} \tag{3.7}$$

where i is varying from 1 to the number of measurements n_k, j is varying from 1 to l and the matrix $\mathbf{H}_k$ has dimensions $n_k \times l$. In Eq. (3.7), the case where $j = 1:3$ represents the sensitivity of the measurements to variations in the position states, while the case where $j = 7:10$ represents the sensitivity to variations in the attitude states.

At this point in the derivation, it is important to introduce a technique that increases the robustness of the algorithm by significantly reducing the chance that the covariance matrix $\mathbf{P}$ becomes non-positive definite. As with any Kalman filter, it is necessary that the covariance matrix $\mathbf{P}$ remains positive definite. However, although the quaternion vector is composed of four elements, only three are independent. The fourth element can be determined at any time from the other three using the unity constraint on the quaternion vector. Hence, one eigenvalue of the $\mathbf{P}$ matrix is always identically zero and numerical errors can make it negative.[10]

[10]The covariance associated with a deterministic variable is zero, justifying the singularity in the

To solve this problem, Lefferts, Markley and Shuster [71] proposed a reduced form of the covariance matrix that takes into account the unity constraint. This technique ensures that no negative eigenvalue is developed as a consequence of one of the quaternions being a deterministic variable. For this estimator, the result is a 12×12 covariance matrix instead of 13×13. The covariance update equations remain unchanged, other than using the reduced size covariance matrix. Equation (2.3) can be rewritten as:

$$\tilde{\mathbf{K}}_k = \tilde{\mathbf{P}}_k^{(-)} \tilde{\mathbf{H}}_k^T \left(\hat{\mathbf{x}}_k^{(-)} \right) \left[\tilde{\mathbf{H}}_k \left(\hat{\mathbf{x}}_k^{(-)} \right) \tilde{\mathbf{P}}_k^{(-)} \tilde{\mathbf{H}}_k^T \left(\hat{\mathbf{x}}_k^{(-)} \right) + \mathbf{R} \right]^{-1} \tag{3.8}$$

where $\tilde{}$ indicates the reduced form and the measurement noise matrix $\mathbf{R}$ is assumed to be constant. Equation (2.7) can also be rewritten in a similar way:

$$\tilde{\mathbf{P}}_k^{(+)} = \left[\mathcal{I} - \tilde{\mathbf{K}}_k \tilde{\mathbf{H}}_k \left(\hat{\mathbf{x}}_k^{(-)} \right) \right] \tilde{\mathbf{P}}_k^{(-)} \tag{3.9}$$

The following equations are used to transition between the reduced and the expanded form:

$$\mathbf{K}_k = \mathbf{S} \left(\hat{\mathbf{q}}_k^{(-)} \right) \tilde{\mathbf{K}}_k \tag{3.10}$$

$$\tilde{\mathbf{H}}_k = \mathbf{H}_k \mathbf{S} \left(\hat{\mathbf{q}}_k^{(-)} \right) \tag{3.11}$$

where $\mathbf{S} \left(\hat{\mathbf{q}}_k^{(-)} \right)$ is composed of:

$$\mathbf{S} \left(\hat{\mathbf{q}}_k^{(-)} \right) = \begin{bmatrix} \mathcal{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathcal{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} & \Xi(\mathbf{q}) & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathcal{I}_{3\times3} \end{bmatrix}_{\mathbf{q}=\hat{\mathbf{q}}_k^{(-)}} \tag{3.12}$$

covariance matrix.

with $\Xi(\mathbf{q})$ taking the following general form [71]:

$$\Xi(\mathbf{q}) = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \tag{3.13}$$

The derivation of $\Xi(\mathbf{q})$ is omitted here and can be found in Ref. [71]. Equation (3.10) is used to compute the expanded Kalman gain matrix $\mathbf{K}_k$ needed in Eq. (2.6), whereas Eq. (3.11) is used to derive the reduced Jacobian matrix $\tilde{\mathbf{H}}_k$ from Eq. (2.5).

With the derivation of the measurement update equations now complete, the next subsection covers the derivation of the propagation equations, which is the second part of the EKF for the global estimator.

Dynamics propagation equations

Neglecting the rotation of the ISS with respect to the inertial frame,[11] the following equation can be used to estimate the dynamics expressed in Eq. (2.8):

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}(\hat{\mathbf{x}}(t))\,\hat{\mathbf{x}}(t) + \mathbf{B}(\hat{\mathbf{x}}(t))\,\mathbf{u}(t) \tag{3.14}$$

To facilitate the implementation of the algorithm in a digital computer, the forward Euler integration approximation is used to express continuous dynamics as discrete in time:

$$\hat{\mathbf{x}}_{k+1}^{(-)} = \hat{\mathbf{x}}_k^{(+)} + \dot{\hat{\mathbf{x}}}_k^{(+)} \Delta t \tag{3.15}$$

Experimentation has shown that this simple method provides good results, as long as the propagation period Δt is small. The propagation period Δt is defined as the time between two updates (Fig. 3-17):

$$\Delta t = t_{k+1} - t_k \tag{3.16}$$

[11] The rotation of the ISS on itself (approximately one rotation every 90 minutes) is barely perceived by the SPHERES gyroscopes (on the order of two counts out of 4096).

Equation (3.14) can be used to rewrite Eq. (3.15) (to first order):

$$\hat{\mathbf{x}}_{k+1}^{(-)} = (\boldsymbol{\mathcal{I}} + \mathbf{A}_k \Delta t)\, \hat{\mathbf{x}}_k^{(+)} + (\mathbf{B}_k \Delta t)\mathbf{u}_k \tag{3.17}$$

The right side of Eq. (3.17) is composed of two terms: the propagated states (to first order) with consideration only for the dynamics of the system $(\boldsymbol{\mathcal{I}} + \mathbf{A}_k \Delta t)\, \hat{\mathbf{x}}_k^{(+)}$ and the state variations caused by firing the thrusters $(\mathbf{B}_k \Delta t)\mathbf{u}_k$. The dynamics matrix $\mathbf{A}_k$ is composed of:

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{0}_{3\times3} & \boldsymbol{\mathcal{I}}_{3\times3} & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{4\times3} & \mathbf{0}_{4\times3} & \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}) & \mathbf{0}_{4\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times4} & \mathbf{0}_{3\times3} \end{bmatrix}_{\boldsymbol{\omega}=\hat{\boldsymbol{\omega}}_k^{(+)}} \tag{3.18}$$

where $\boldsymbol{\omega}$ is the angular rate component of the state vector and $\boldsymbol{\Omega}(\boldsymbol{\omega})$ has the following general form [126]:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) \equiv \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{3.19}$$

The control input vector elements are defined as the fraction of the propagation period (number between 0 and 1) for which each thruster was turned ON and delivered a constant thrust:

$$\mathbf{u}_k = [u_1 \;\; u_2 \;\; \cdots \;\; u_N]^T \tag{3.20}$$

The matrix $\mathbf{B}_k$ is composed of:

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{0}_{3\times12} \\ \frac{1}{m}\boldsymbol{\theta}^T(\mathbf{q})\mathbf{D}[diag(\boldsymbol{\mathcal{F}})] \\ \mathbf{0}_{4\times12} \\ \mathbf{0}_{3\times12} \end{bmatrix}_{\mathbf{q}=\hat{\mathbf{q}}_k^{(+)}} \tag{3.21}$$

where m is the mass of the satellite, $[diag(\mathcal{F})]$ is a $N \times N$ diagonal matrix with the vector of thruster strength $\mathcal{F}$ as its diagonal, N the number of thrusters and $\boldsymbol{\theta}^T(\mathbf{q})$ the transpose of the attitude matrix defined in Eq. (3.6). Because of the availability of direct, high bandwidth measurements of angular rates, and because of the slow angular rates that each satellite experiences, the angular acceleration produced by the thrusters are not included in the state propagation equation. This is indicated by the zeros in the bottom half of the matrix $\mathbf{B}_k$. Although not currently implemented, more accurate thruster models [131, 130] can be integrated as part of the matrix $\mathbf{B}_k$, if necessary.

As for the covariance matrix, a discretized propagation equation, based on a reduced state transition matrix and a reduced covariance matrix, has been developed in Ref. [71]:

$$\tilde{\mathbf{P}}_{k+1}^{(-)} = \tilde{\boldsymbol{\Phi}}_k^{(+)} \tilde{\mathbf{P}}_k^{(+)} \tilde{\boldsymbol{\Phi}}_k^{(+)^T} + \tilde{\mathbf{Q}} \Delta t \tag{3.22}$$

where the process noise matrix $\mathbf{Q}$ is assumed to be constant. The reduced state transition matrix $\tilde{\boldsymbol{\Phi}}_k^{(+)}$ is composed of:

$$\tilde{\boldsymbol{\Phi}}_k^{(+)} = \begin{bmatrix} \mathcal{I}_{3\times3} & \mathcal{I}_{3\times3}\Delta t & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathcal{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{\Lambda}_k^{(+)} & \boldsymbol{\aleph}_k^{(+)} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathcal{I}_{3\times3} \end{bmatrix} \tag{3.23}$$

with $\boldsymbol{\Lambda}(\mathbf{q})$ and $\boldsymbol{\aleph}(\mathbf{q})$ being expressed as [71]:

$$\boldsymbol{\Lambda}_k^{(+)} = \boldsymbol{\theta}\left(\mathbf{q}_k^{(+)}\right) \cdot \boldsymbol{\theta}^T\left(\mathbf{q}_{k+1}^{(-)}\right) \tag{3.24}$$

$$\boldsymbol{\aleph}_k^{(+)} = -\frac{1}{2}\boldsymbol{\Lambda}_k \Delta t \tag{3.25}$$

This reduced state transition matrix was developed for a state vector that includes gyroscope biases instead of angular rates. The model used when deriving Eqs. (3.23) to (3.25) is compatible with having angular rates in the state vector instead of gyroscope biases, as long as the propagation period is small. Also, the increase in covariance due

to the uncertainty associated with the thruster firings is currently not implemented in Eq. (3.22).

At this point, all the necessary EKF equations have been derived for the use of the reduced representation of the covariance matrix. The process can be summarized as follows. Starting from a plausible guess of the state estimates with its associated reduced covariance, a set of TOF data is converted to range measurements (Eq. (3.3)) as soon as it is collected. The expected measurement vector, as well as its Jacobian, are then computed using Eq. (3.4) and Eq. (3.7), respectively. The Jacobian is reduced using Eq. (3.11) and the reduced Kalman gain is computed using Eq. (3.8). The reduced Kalman gain is used to update the reduced covariance using Eq. (3.9). The expanded gain can be recovered using Eq. (3.10), which then allows the states to be updated using Eq. (2.6). These states are propagated using Eq. (3.17), while the reduced covariance matrix is propagated using Eq. (3.22). This process is repeated after the next set of TOF data is collected. The integration of the gyroscope measurements is now covered.

Integration of gyroscope measurements

The gyroscopes provide a direct measurement of the angular rates of the satellite. In general, they can be sampled at a very fast rate, greatly improving the attitude estimation bandwidth, especially when thruster commands are not made available to the EKF. The gyroscope data can be integrated into an EKF in many ways. Since gyroscopes do not provide any information about the absolute position of the satellite, it is not necessary to proceed with a full state update every time they are sampled.

Gyroscope measurements are integrated using Eqs. (3.17) to (3.19). The three pre-filtered angular rate measurements (using the anti-aliasing filter described in Appendix B) are used to directly determine the angular rate components ω_x, ω_y and ω_z of the state vector. These components allow to compute $\boldsymbol{\Omega}(\boldsymbol{\omega})$, which is then used in the propagation equation through the matrix $\mathbf{A}$. The states are propagated until either a new set of U/S measurements is available or a new angular rate vector is measured from the gyroscopes. Since the gyroscope measurements can be acquired

at a much faster rate than the U/S measurements, this process has the advantage of improving the bandwidth of the attitude estimation as well as the first order approximation made when deriving Eq. (3.17). Also, for the sake of computational savings, only the angular rate components of the state vector are updated when sampling the gyroscopes, not the covariance matrix. Testing has shown that the performance of the EKF is barely affected by not updating the covariance matrix when integrating gyroscope measurements (the process noise matrix $\mathbf{Q}$ handles the problem sufficiently).

Another method for integrating gyroscope measurements is by replacing the angular rate components of the state vector in Eq. (3.1) by a gyroscope bias vector $\mathbf{b}$ and by using the attitude measurements to estimate the gyroscope biases. The angular rates are obtained by subtracting the estimated bias vector from the gyroscope reading vector $\boldsymbol{\mu}$ using a simple equation [71]:

$$\hat{\boldsymbol{\omega}} = \boldsymbol{\mu} - \hat{\mathbf{b}} \tag{3.26}$$

This equation was derived from a gyroscope model proposed by Farrenkopf [39]. The same EKF process, with the exact same equations, can still be used, since they were originally derived for a state vector included gyroscope biases instead of angular rates. However, the process noise matrix $\mathbf{Q}$ needs to be tuned because the gyroscope biases are expected to change at a much slower rate than the angular rates themselves.

To increase the convergence rate of the EKF when estimating gyroscope biases, it is possible to use the angular rate in the state vector until the attitude solution converges, and then to replace them by the gyroscope biases. One has to also consider the reliability of the attitude measurement system prior to including the gyroscope biases in the state vector. Experimentation has shown that an error (even temporary) in the attitude measurements can corrupt the gyroscope biases. Consequently the angular rates derived from the gyroscope measurements, which depend on accurate biases, would also be corrupted. Because of the small process noise typically used in the dynamics model of the gyroscope biases, this problem would take a long time to correct itself, with respect to the typical duration of an experiment on SPHERES.

For experiments where estimates of the relative states between two satellites are required, a user has the option to subtract the global estimates of two satellites, or to use either variants of the relative state estimator, including the single beacon state estimator. The next subsection presents the relative state estimator and all its variants, which are able to provide up to six DOF relative state estimates using a single external beacon and onboard gyroscopes.

3.4.2 The relative state estimator

The role of the relative state estimator is to provide relative state estimates between two satellites using the onboard beacons. This estimator is used in situations where global positioning is not made available (e.g., to simulate a docking maneuver around Mars). Nonlinear filtering methods are also used to obtain the relative state estimates. This section describes, in detail, the EKF used to generate relative state estimates. Three software modules were derived from this EKF, all of which are covered in this subsection.

State vector for relative state estimates

When only one external beacon is used (either attached on the wall or on another satellite), only six states are observable. They are expressed in the following state vector, representing the relative range and velocity of the external beacon in the satellite's body frame (defined in Fig. 3-5):

$$\mathbf{x} = [r_x \;\; r_y \;\; r_z \;\; v_x \;\; v_y \;\; v_z]^T \tag{3.27}$$

Unless otherwise specified, the state vector used in Section 3.4.2 is expressed in the body coordinate frame of the satellite and has $l = 6$ elements. The measurement update equations, as well as the dynamics propagation equations, are now presented.

Measurement update equations

This subsection presents the equations used in updating the state estimates when U/S measurements are collected. The process is simpler than in Section 3.4.1 since no quaternion terms are used in the state vector, removing the need for a reduced covariance matrix. Starting from a plausible guess of the state estimates, with their associated covariance, a measurement is processed using Eq. (3.3) as soon as it is collected. The expected measurement vector, as well as its Jacobian, are then computed using the following equations:

$$[\mathbf{h}_k]_i = \left|\left|\hat{\mathbf{r}}_k - \boldsymbol{\rho}_i^{body}\right|\right| \tag{3.28}$$

$$[\mathbf{H}_k]_{ij} = \begin{cases} \frac{\left[\hat{\mathbf{r}}_k - \boldsymbol{\rho}_i^{body}\right]_j}{[\mathbf{h}_k]_i} & , j = 1:3 \\ 0 & , \text{otherwise} \end{cases} \tag{3.29}$$

The Kalman gain is directly computed using Eq. (2.3). The Kalman gain allows an update of the covariance matrix using Eq. (2.7), as well as the propagation of the state estimates using Eq. (2.6). The equations used in the state propagation phase are now presented.

Dynamics propagation equations

The states are propagated using Eq. (3.17) with the matrices $\mathbf{A}_k$ and $\mathbf{B}_k$ being composed of:

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathcal{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{3.30}$$

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{0}_{3\times12} \\ -\frac{1}{m}\mathbf{D}[diag(\boldsymbol{\mathcal{F}})] \end{bmatrix} \tag{3.31}$$

The minus sign in Eq. (3.31) is necessary to indicate that an acceleration of the satellite in one direction is in fact interpreted as the beacon accelerating in the opposite direction. The derivative of the covariance is computed using Eqs. (2.9) and (3.30),

with $t = t_k^{(+)}$. The following approximation is used to propagate the covariance:

$$\mathbf{P}_{k+1}^{(-)} = \mathbf{P}_k^{(+)} + \dot{\mathbf{P}}_k^{(+)} \Delta t \tag{3.32}$$

Like with the global estimator, the increase in covariance due to thruster firings is currently not modeled.

This completes the necessary equations for the EKF of the relative state estimator. They all occur in the order presented. This recursive process is repeated every time a new U/S measurement is collected.

Estimating the relative angular rate

One of the limitations of the EKF described above is the lack of a direct measurement of the angular rate between the two satellites. This is especially true when docking to a rotating target, when the chaser needs to constantly accelerate to stay in front of the target. The angular rates provided by Eqs. (3.38) and (3.39) describe the rate at which the line-of-sight to the beacon of the other satellite changes direction. This is different than the relative angular rate.

The onboard gyroscopes can be used to solve this problem. When the target satellite is holding its attitude, the measurements of the chaser's gyroscopes can be used to directly approximate the relative angular rate. If the target is rotating, it needs to transmit its own angular rate to the chaser. The difference between the angular rates registered by both the chaser and the target is the relative angular rate. If the angular rate of the target remains approximately constant, the target does not need to communicate its rate to the chaser very often. An advantage of using gyroscopes is their high sampling rate, which helps reduce the lag in the state estimates.

The EKF-based relative state estimator has been successfully used on the SPHERES testbed. The following subsections describe three GN&C software modules that manipulates differently the relative state estimates it provides.

Module providing range and bearing angles

As shown in Section 1.3, early docking navigation sensors provided only range and bearing information to the chaser. This GN&C module for relative state estimates is the SPHERES equivalent of early docking navigation sensors. It takes as input the state vector in Eq. (3.27) and expresses it in terms of the range and bearing angles to the beacon (spherical coordinates):

$$\mathbf{x} = \begin{bmatrix} \Gamma & \dot{\Gamma} & \theta_{pitch} & \theta_{yaw} & \omega_{pitch} & \omega_{yaw} \end{bmatrix}^T \tag{3.33}$$

where Γ is the magnitude of the range vector $\mathbf{r}$, and θ_{pitch} and θ_{yaw} are the bearing angles to the beacon (Fig. 3-18). The states in Eq. (3.27) are transformed to those in Eq. (3.33) using the following equations. The range Γ is expressed as:

$$\Gamma = \sqrt{r_x^2 + r_y^2 + r_z^2} \tag{3.34}$$

The range rate is determined by differentiating Eq. (3.34):

$$\dot{\Gamma} = \begin{cases} \frac{1}{\Gamma}\left(r_x \cdot v_x + r_y \cdot v_y + r_z \cdot v_z\right) & , \Gamma \neq 0 \\ 0 & , \Gamma = 0 \end{cases} \tag{3.35}$$

The second case in Eq. (3.35) is not physically possible because the range computed from the center of one satellite to the beacon of another satellite should never be less than the radius of a satellite. However, in practice, since the algorithm is unconstrained, nothing mathematically prevents Γ from becoming small during the convergence process. Directly assigning $\dot{\Gamma} = 0$ when $\Gamma = 0$ prevents a division by zero. Although currently not implemented, the condition $\Gamma = 0$ can be modified to $\Gamma < \epsilon$, where ϵ is some tolerance around the origin. This would prevent arbitrarily large $\dot{\Gamma}$ from being computed when Γ gets close to zero, and would result in a more numerically stable algorithm.

The bearing to the beacon consists of two angles, similar to the elevation and az-

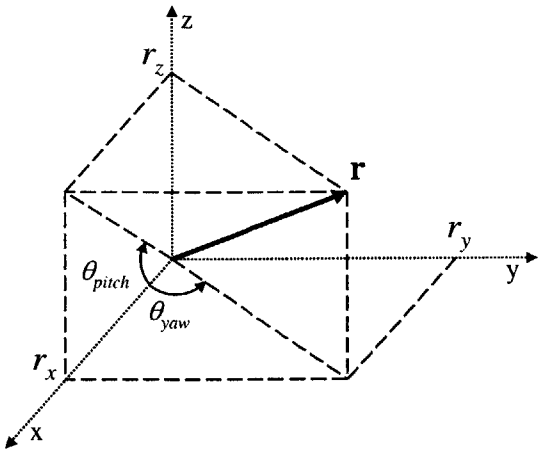


Figure 3-18: Beacon location (**r**) in the satellite body frame (x, y, z).

imuth angles. The pitch (θ_{pitch}) and yaw (θ_{yaw}) bearing angles, illustrated in Fig. 3-18, define the rotation a satellite must perform around its y- and z-body axes, respectively, to point its expansion port face (+x face, opposite to the -x face shown in Fig. 3-5) directly at the beacon, whose location is defined in the satellite's body frame by **r**. These angles are related to the states in Eq. (3.27) by:

$$\theta_{pitch} = \begin{cases} \arctan\left(-r_z/r_x\right) & , r_x \neq 0 \\ -1 \cdot sgn\left(r_z\right) \cdot \frac{\pi}{2} & , r_x = 0 \end{cases} \tag{3.36}$$

$$\theta_{yaw} = \begin{cases} \arctan\left(r_y/r_x\right) & , r_x \neq 0 \\ sgn\left(r_y\right) \cdot \frac{\pi}{2} & , r_x = 0 \end{cases} \tag{3.37}$$

where $sgn()$ refers to the signum function. Equations (3.36) and (3.37) are differentiated to obtain the angular rates ω_{pitch} and ω_{yaw}:

$$\omega_{pitch} = \begin{cases} \left(r_z \cdot v_x - r_x \cdot v_z\right) / \left(r_x^2 + r_z^2\right) & , r_x \neq 0 \text{ or } r_z \neq 0 \\ 0 & , r_x = 0 \text{ and } r_z = 0 \end{cases} \tag{3.38}$$

$$\omega_{yaw} = \begin{cases} \left(r_x \cdot v_y - r_y \cdot v_x\right) / \left(r_x^2 + r_y^2\right) & , r_x \neq 0 \text{ or } r_y \neq 0 \\ 0 & , r_x = 0 \text{ and } r_y = 0 \end{cases} \tag{3.39}$$

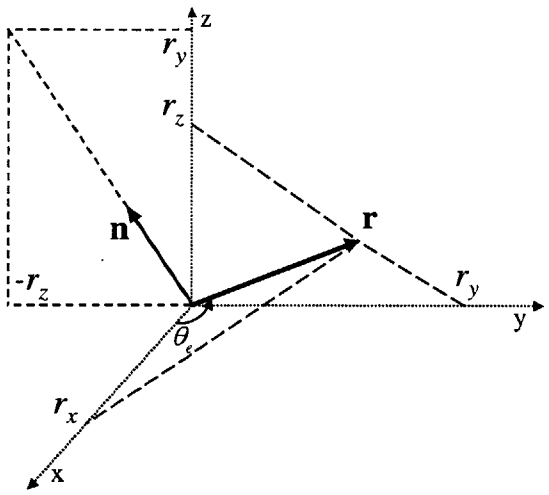


Figure 3-19: Attitude angular error around the Euler axis (**n**) shown in the satellite body frame (x,y,z).

Equations (3.36) to (3.39) are written such that ω_{pitch} is positive when ω_y is positive, and ω_{yaw} is positive when ω_z is positive.

It is also possible to express the bearing angles in terms of quaternions. Figure 3-19 illustrates the approach taken. The attitude angular error θ_e around the Euler axis **n** is found using:

$$\theta_e = \begin{cases} \arccos\left(r_x/\Gamma\right) & , \Gamma \neq 0 \\ 0 & , \Gamma = 0 \end{cases} \tag{3.40}$$

The Euler axis, expressed by the unit vector $\mathbf{n} = [n_x \;\; n_y \;\; n_z]$, is found to be:

$$[n_x \;\; n_y \;\; n_z]^T = \begin{cases} \left[0 \;\; \frac{-r_z}{\sqrt{r_y^2+r_z^2}} \;\; \frac{r_y}{\sqrt{r_y^2+r_z^2}}\right]^T & , r_y \neq 0 \text{ or } r_z \neq 0 \\ [1 \;\; 0 \;\; 0]^T & , r_y = 0 \text{ and } r_z = 0 \end{cases} \tag{3.41}$$

Once the attitude angular error θ_e[12] and the Euler axis **n** are known, they are combined using Eq. (3.2) to form the quaternion vector **q** expressing the bearing angles. It is interesting to note that Eqs. (3.40) and (3.41) are implicitly constraining $q_1 = 0$, as no roll information can be extracted using this method. Therefore, the

[12]Here, θ_e is defined by the angular offset about the Euler axis between the normal to the docking face and the beacon.

Euler axis, determined in Eq. (3.41), is constrained in the y-z plane.

A similar process can be followed for the cases where the bearing angles are expressed with respect to a different face of the satellite. The following equations apply for the bearing angles with respect to the docking face (-x face in Fig. 3-5). They replace Eqs. (3.37), (3.41) and (3.2), respectively.

$$\theta_{yaw} = \begin{cases} \arctan\left(-r_y/r_x\right) & , r_x \neq 0 \\ -1 \cdot sgn\left(r_y\right) \cdot \frac{\pi}{2} & , r_x = 0 \end{cases} \tag{3.42}$$

$$[n_x \;\; n_y \;\; n_z]^T = \begin{cases} \left[0 \;\; \frac{r_z}{\sqrt{r_y^2+r_z^2}} \;\; \frac{-r_y}{\sqrt{r_y^2+r_z^2}}\right]^T & , r_y \neq 0 \text{ or } r_z \neq 0 \\ [1 \;\; 0 \;\; 0]^T & , r_y = 0 \text{ and } r_z = 0 \end{cases} \tag{3.43}$$

$$\begin{aligned} \mathbf{q} &= \left[n_x \sin\left(\frac{\pi-\theta_e}{2}\right) \;\; n_y \sin\left(\frac{\pi-\theta_e}{2}\right) \;\; n_z \sin\left(\frac{\pi-\theta_e}{2}\right) \;\; \cos\left(\frac{\pi-\theta_e}{2}\right)\right]^T \\ &= \left[n_x \cos\left(\frac{\theta_e}{2}\right) \;\; n_y \cos\left(\frac{\theta_e}{2}\right) \;\; n_z \cos\left(\frac{\theta_e}{2}\right) \;\; \sin\left(\frac{\theta_e}{2}\right)\right]^T \end{aligned} \tag{3.44}$$

Two different state vectors have been presented so far to express the relative states between two satellites. The communication of various elements of these state vectors between two satellites can increase the number of observable states, as described in the next subsection.

Module combining the states of two cooperative satellites

This module extends the sensing capabilities of one satellite (typically the chaser) by using state information from a second satellite (the target). It inputs the relative state estimates computed by both satellites and outputs an extended state vector for the chaser. In the case where the docking faces are approximately pointed to each other, up to ten states can be observed solely using the onboard beacons of the satellites: the range (Γ), the horizontal and vertical tangential displacements (δ_y and δ_z), the range rate ($\dot{\Gamma}$), the tangential velocities ($\dot{\delta}_y$ and $\dot{\delta}_z$), the pitch and yaw angles

(θ_{pitch} and θ_{yaw}), and the pitch and yaw angular rates (ω_{pitch} and ω_{yaw}).

Upon reception of the U/S ping emitted by the target, the chaser processes the information and computes its relative states with respect to the target's beacon as expressed in Eq. (3.33) and shown Fig. 3-20a for the 2-D problem (for the purpose of clarity, the states in the third dimension, θ_{pitch} and ω_{pitch}, are omitted). The same process occurs when the chaser emits its U/S ping, but this time the target computes only the relative states shown in Eq. (3.27) (Fig. 3-20b). It then communicates them to the chaser. The tangential displacements and velocities (Fig. 3-20c) can be taken directly from the r_y, r_z, v_y and v_z components of the target's relative state vector when the chaser is approximately aligned in front of the target's docking face ($\theta_{yaw,t} \rightarrow 0$) and it points its docking face towards the target ($\theta_{yaw,c} \approx 0$):

$$\delta_y \approx r_{y,t} \tag{3.45}$$

$$\delta_z \approx r_{z,t} \tag{3.46}$$

The single beacon state estimator module

For the first three test sessions on ISS (May 18, May 20 and August 12, 2006), only one satellite was available and only one beacon was mounted to the wall. The main limitation, when using the GN&C module that determines the range and bearing angles to a beacon, is the lack of position information in a plane perpendicular to the line-of-sight to the beacon. Faced with the challenge of providing six DOF state estimation, while using only a single U/S beacon and three gyroscopes, a simple solution was developed. For the three minute duration of a test, the change in the attitude of the satellite, with respect to the initial attitude, can be adequately determined by integrating the angular rates provided by the gyroscopes, while the position and the velocity would be computed by repetitively determining the location of the external beacon in the satellite's body frame using the TOF data.

Although the resulting location of the external beacon does not yield enough

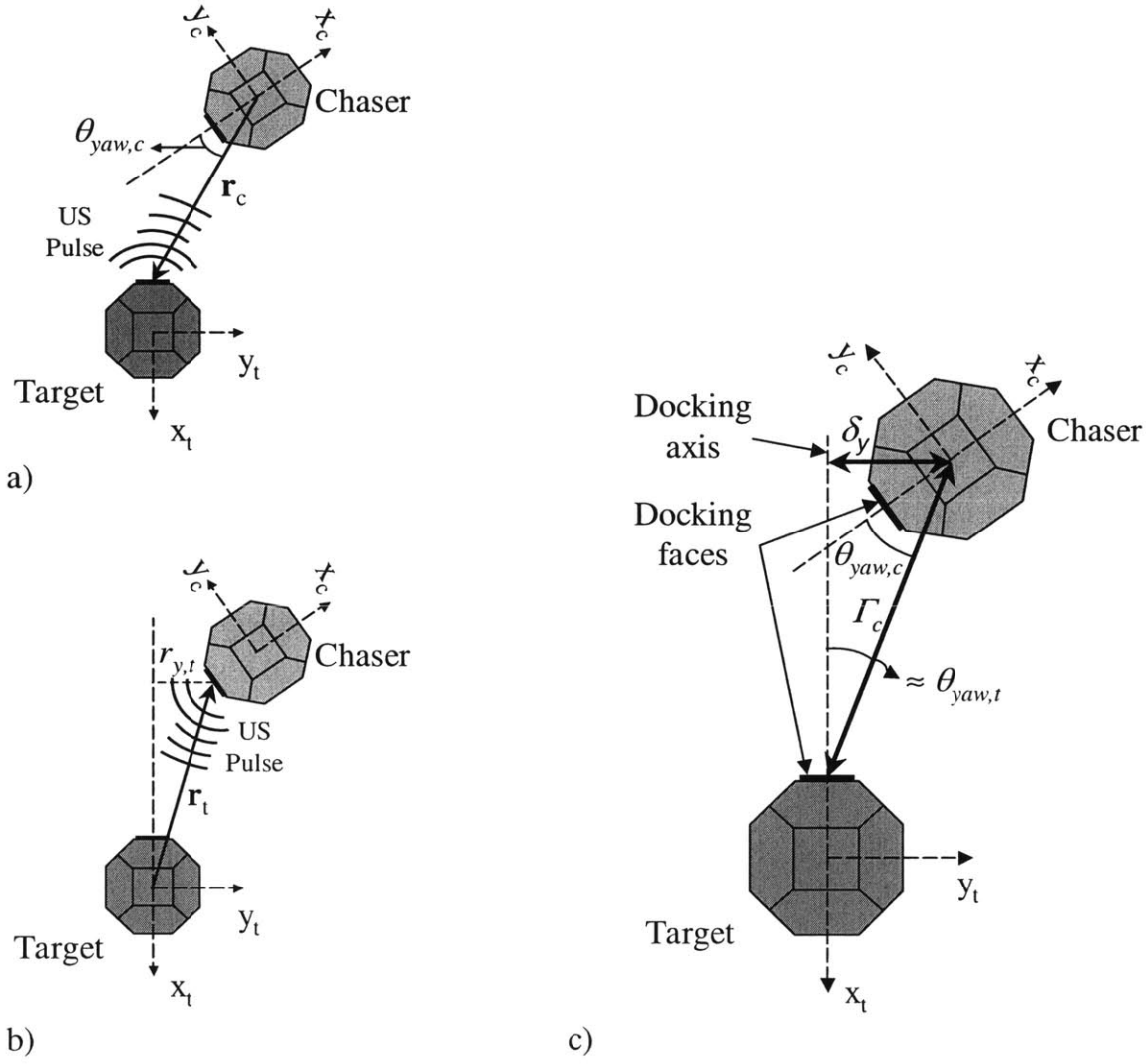


Figure 3-20: Relative state estimates (2-D problem) computed when a) the target is pinging its beacon and b) the chaser is pinging its beacon. The combined relative state estimates are shown in c).

information to get an absolute position, regular sampling of the beacon provides displacement information by subtracting the updated beacon location to the one initially recorded, assuming that the attitude of the satellite is maintained fixed during the experiment (using a high frequency attitude controller). By decoupling attitude and position measurements, the satellite can navigate in all six DOF for as long as the gyroscope drift remains small (approximately two minutes, enough for a typical SPHERES experiment). This section derives the equations used for the single beacon state estimator.

The three translational DOF (displacement $\mathbf{d}$ and velocity $\dot{\mathbf{d}}$) are determined using the information provided by the relative state estimator:

$$\mathbf{d}_k = \hat{\mathbf{r}}_0 - \hat{\mathbf{r}}_k \tag{3.47}$$

$$\dot{\mathbf{d}}_k = -\hat{\mathbf{v}}_k \tag{3.48}$$

where $\hat{\mathbf{r}}_0$ is the initial beacon location in the satellite's body frame. These equations are only true if the satellite holds its attitude. If needed, one can derive displacement equations accounting for the change of the location of the beacon caused by attitude slews.

The three rotational DOF (attitude $\mathbf{q}$ and angular rate $\boldsymbol{\omega}$) are determined independently of the translational DOF using measurements from the gyroscopes only. The angular rate is computed using Eq. (3.26) with fixed gyroscope biases. The attitude quaternions are directly computed using the forward Euler integration:

$$\hat{\mathbf{q}}_{k+1} = \hat{\mathbf{q}}_k + \frac{1}{2}\Omega\left(\hat{\boldsymbol{\omega}}_k\right)\hat{\mathbf{q}}_k \Delta t \tag{3.49}$$

With the attitude measurement provided by Eq. (3.49), a high bandwidth attitude controller maintains the attitude of the satellite fixed. Although the telemetry indicates a fixed attitude, the satellite is, in fact, slowly rotating, following any drift the gyroscopes might have since no external measurements are used to correct that drift. Any attitude change caused by the drift of the gyroscope also results in a displace-

ment that cannot be perceived through any of the state variables in Eqs. (3.47) and (3.48). This displacement is approximately proportional to the attitude error and the distance to the beacon. Experimentation has shown that integrating the gyroscope data using Eq. (3.49), without any attitude correction, can lead to an attitude drift of up to 5 deg/min. Although this seems an important drift at first glance, it still shows that decoupling displacement determination from attitude determination, using the method shown in Section 3.4.2, yields good results for the purpose of an experiment using the SPHERES testbed, which lasts on the order of a few minutes.

This completes the detailed description of the implementation of each navigation module. The next section focuses on the hardware and software modules allowing the satellites to maneuver in the test environment.

3.5 Implementation of control algorithms

Together with the navigation modules, the control modules form the lowest level of the SPHERES GN&C architecture presented in Fig. 3-16. They directly interface with the SPHERES propulsion subsystem. They are generic and can be used for a wide variety of experiments, including autonomous docking and formation flight experiments. The following subsections describe three control modules that were implemented as part of this research: a PID-type controller, a pulse-width modulator that is usually coupled with the PID-type controller, and a phase plane controller.

3.5.1 PID-type controllers

The first control modules to be implemented were standard PID-type controllers to control both the attitude and the position of the satellites [116, 126]. These controllers simply output force and torque commands proportional to state errors. Although the attitude dynamics using quaternions is nonlinear, the PID-type attitude controller uses the quaternions computed by the estimator to approximate angular errors. This approximation is accurate for small angular errors (<10 degrees). This allows each axis to be controlled independently. The result is an expression that is very similar

to a standard PID controller. The attitude control law is of the form:

$$\mathbf{T} = \begin{bmatrix} 2 \cdot K_p \cdot sgn(q_4) \cdot q_1 + 2 \cdot K_i \cdot \int (sgn(q_4) \cdot q_1)\, dt + K_d \cdot \omega_x \\ 2 \cdot K_p \cdot sgn(q_4) \cdot q_2 + 2 \cdot K_i \cdot \int (sgn(q_4) \cdot q_2)\, dt + K_d \cdot \omega_y \\ 2 \cdot K_p \cdot sgn(q_4) \cdot q_3 + 2 \cdot K_i \cdot \int (sgn(q_4) \cdot q_3)\, dt + K_d \cdot \omega_z \end{bmatrix} \tag{3.50}$$

Wie [124, 125] has shown that the PD version of that controller (with the integral gain set to zero) is globally asymptotically stable. Tests have also shown that this control law can provide enough accuracy for the purpose of an autonomous docking maneuver.

One difficulty in implementing the PID controller in the SPHERES testbed comes from the proper selection of the gains to achieve the desired performance, especially when performing experiments on the MIT SSL 2-D air table where there are substantial perturbations. The following gain model provides good performance. It is derived from the standard equations modeling a continuous-time second-order system with a natural frequency ω_n and a damping ratio ζ [126].

For a PID position controller:

$$K_P = m\left(\omega_n^2 + 2\zeta\omega_n/\tau\right) \tag{3.51}$$

$$K_I = m\left(\omega_n^2/\tau\right) \tag{3.52}$$

$$K_D = m\left(2\zeta\omega_n + 1/\tau\right) \tag{3.53}$$

For a PD position controller with K_I=0:

$$K_P = m\omega_n^2 \tag{3.54}$$

$$K_D = m\left(2\zeta\omega_n\right) \tag{3.55}$$

where m is the mass of the satellite (and its air carriage in 2-D ground experiments).

For the PID-type attitude controller shown in Eq. (3.50):

$$K_P = I\left(\omega_n^2 + 2\zeta\omega_n/\tau\right) \tag{3.56}$$

$$K_I = I\left(\omega_n^2/\tau\right) \tag{3.57}$$

$$K_D = 0.75 \cdot I\left(2\zeta\omega_n + 1/\tau\right) \tag{3.58}$$

For the PD-type attitude controller derived from Eq. (3.50) with K_I=0:

$$K_P = I\omega_n^2 \tag{3.59}$$

$$K_D = 0.75 \cdot I\left(2\zeta\omega_n\right) \tag{3.60}$$

where I is the inertia (scalar) around a body axis of the satellite (and its air carriage in 2-D ground experiments). The factor 0.75 used in Eqs. (3.58) and (3.60) was determined through multiple experiments involving PID-type position and attitude controllers used simultaneously. For the PID-type controllers presented above, the time constant τ, associated with the integral gain, is usually set between 10 and 20 seconds.

To improve the behavior of the system, and prevent large overshoots due to integral windup, a limit is placed on the maximum absolute value that the integral term can achieve. This proved to speed up the recovery from large unmodeled perturbations. At the time of writing of this thesis, only one PID controller for position and one for attitude can be used simultaneously. Setting the integral term as an argument of the function would solve that problem. Also, every time the integral gain is set to zero, the integral term is reset such that it increases only when a PID controller is used. Experimentation has shown that when the state errors are larger than 10 cm in position and 10 degrees in attitude, a PD controller is more appropriate to avoid overshoots. When the errors decrease below these values, a PID controller can be used for greater accuracy.

To transform forces and torques output by the controllers to thruster commands, a pulse-width modulator is needed. This is the subject of the next subsection.

3.5.2 Thruster pulse-width modulator

The thruster pulse-width modulator currently available on SPHERES uses a thrust mapping matrix $\mathcal{M}$ and conservation of impulse (Fig. 2-4) to convert forces and torques into thruster ON/OFF times. It consists of a direct implementation of the theory presented in Section 2.2.1. Because of hardware limitations, the minimum thruster opening time is set at 10 msec. It is possible, although fuel expensive, to generate smaller net thrust inputs (equivalent to a 1 msec thrust command) by turning ON two opposing thrusters, and turning one OFF slightly before the other. In practice, this technique might be problematic knowing the variations in thruster opening and closing delays. No investigation has been performed to validate this approach.

Since the control law does not place an upper bound on the commanded torque, it is also possible that a thruster is commanded to produce a thrust that exceeds its capability. When this occurs, the thrust durations for all the thrusters are normalized such that the maximum thrust duration is equal to the control period. This maintains the net actuation direction at the expense of the absolute thrust amplitude. It results in a net impulse loss. It is currently up to the user to tune the PID-type controllers properly to avoid this situation.

Early testing of the PID-type controllers, combined with the pulse-width modulator, showed adequate performance for as long as the state estimators remained stable. Jumps in the state estimates quickly drove the controller to saturate the thrusters for an extended period of time and often resulted in collisions. Moreover, large slews usually resulted in overshoots. Therefore, a controller with a better behavior in the presence of perturbations and large errors was needed. The resulting controller is presented in the following section.

3.5.3 Phase plane controllers

A phase plane controller, like the one shown in Fig. 2-5, has been developed to provide an efficient and accurate way to track a trajectory during different stages

of the autonomous docking maneuver. This approach is widely used for spacecraft attitude control, such as the Space Shuttle.

The phase plane controller is programmable. It can output either forces and torques or thruster ON/OFF commands. It can be used to control one DOF at a time, or a combination of multiple DOFs (e.g., x- and y-translations, and z-rotations). The user can specify the duty cycle for both attitude and position control, which determines the shape of the switch curves S1 and S4 in Fig. 2-5. The attitude and position deadband can also be specified, which sets the horizontal offset of the switch curves S1 and S4 from the origin.

However, the switch curves marking the rate limits of the drift channel (S3 and S5) are not directly programmable. Experimentation has shown that rate limits around 0.02 m/sec and 5 deg/sec are adequate for SPHERES; therefore, these values are currently hard coded in the software. It is these rate limits that prevent constant thrust saturation when the error is large (no thrust is generated once the rate limit is reached). To avoid a constant jittering over the drift channel, the gap between S3 and S5 (Fig. 2-5) needs to larger than the ΔV produced by a pulse.[13] S5 is currently set to $0.6 * \Delta V$ over the rate limit while S3 is $0.6 * \Delta V$ under it. The location of the switch curve S2 is directly determined by the ΔV produced by a pulse. This minimizes firing by allowing the rate after the pulse to get as close as possible to zero.

When the phase plane logic determines that a specific thruster needs to be actuated, it is actuated for the full duration of the thrust window. This results in a limit cycling around Region 5 in Fig. 2-5. When simultaneously controlling the satellite around multiple axes, the algorithm also makes sure that two opposing thrusters are not actuated at the same time. Experimentation has shown that this controller is more fuel efficient and presents less overshoot than a standard PID-type controller in the presence of unmodeled perturbations and over large slews.

All the control modules presented in this section require only a minimal amount of computation time and are usually run in the software interface presented in Sec-

[13]This ΔV, representing here a variation of rate, whether linear or angular, is a function of the duty cycle entered by the user.

tion 3.2.1 in the primary interface function called *gspControl*.

A series of more advanced controllers, like LQR-type controllers and nonlinear controllers [93], are currently under development. However, the control modules shown in this section provided enough flexibility and accuracy for the purpose of the autonomous docking experiments presented in this thesis. The next section covers the current implementation of path planning algorithms.

3.6 Implementation of path planning algorithms

The only path planning algorithm fully implemented is the glideslope algorithm. The approach used when using advanced path planning algorithms has been to compute a trajectory offline, and code it in the software uploaded to the satellite. More details are provided below on both concepts.

3.6.1 The glideslope controller

A version of the glideslope algorithm presented in Section 2.3.3 has been implemented for the purpose of planning and controlling a chaser in an approach trajectory toward a target. It is referred to as the glideslope controller because its current implementation gives it the capabilities of both a planner and a controller. Like the phase plane controller, the glideslope controller also specifies a trajectory to follow in the phase plane, which is linear. To improve the performance in the presence of gravity, where there are significant disturbances, the chaser follows the prescribed trajectory using a number of pulses that depends on the initial separation with the target. Trajectory corrections are only applied when the time comes to fire another pulse, which is not at every control period. The implementation of the glideslope algorithm is shown in Fig. 3-21.

Experimentation has shown that this algorithm is appropriate for tracking an approach along the docking axis, but not to make corrections in the plane perpendicular to the docking axis. In that case, a PID-type or a standard phase plane controller, with a faster control frequency, is more accurate, especially in the presence of un-

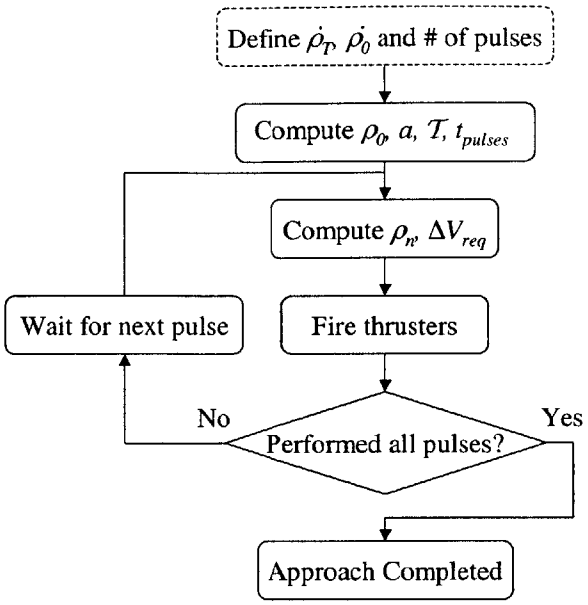


Figure 3-21: Implementation of the glideslope algorithm along the docking axis.

modeled perturbations. Therefore, the DOF controlled by the glideslope controller is programmable by the user. Also, the resulting trajectory in the phase plane tends to jitter over the intended linear trajectory. The solution to that problem might be to implement a glideslope corridor, similar to the drift channel in the phase plane controller, where the chaser does not fire if it is within a certain tolerance around the desired trajectory. This tolerance can also be decreasing with the distance-to-go.

3.6.2 Advanced path planning algorithms

For the trajectory planning of complex autonomous docking maneuvers, like the ones involving collision avoidance while minimizing fuel consumption, advanced algorithms are needed. However, the MPC and MILP algorithms presented in Section 2.3 require a solver to compute a solution. Approximations can be made to find a solution using only a LP solver [15].

At this time, no fully functional LP solver module has been implemented on SPHERES. Therefore, when performing a complex autonomous docking experiment requiring the solution of a MILP problem, the solution is found offline and uploaded

prior to the start of the experiment. Usually, a PID-type controller running onboard the chaser tracks the pre-planned trajectory. Since the trajectory is computed offline, the target needs to actively hold its intended position in order for the experiment to be successful. This approach has been successfully used in many occasions during ISS test sessions. Eventually, the implementation of a solver will allow the trajectory to be computed online.

The remaining sections cover the implementation of algorithms at a higher level of the GN&C architecture presented in Section 3.3.

3.7 Implementation of FDIR algorithms

The capability to quickly and accurately detect failures and take recovery actions is essential to all autonomous docking maneuvers when no human intervention is possible. It is also highly desirable to have FDIR capability at all levels of the GN&C architecture. Section 2.4 presented many avenues. More effort has been concentrated on the two presented in this section.

3.7.1 FD through filter innovation analysis

Following a series of docking experiments performed in the ISS, where measurement errors caused the estimator to lose convergence, a U/S measurement error detection system was embedded in the EKF used by the global estimator. The fault detection technique described in Section 2.4.1 was successfully implemented as follows. A fault is detected when the sum of the filter innovations ι coming from a given set of three or four pre-filtered measurements, as shown in Eq. (3.61), jumps above a given threshold. The threshold was determined from the analysis of data collected during ISS experiments:

$$\iota = \left|\left| \mathbf{z}_k - \mathbf{h}_k \left(\hat{\mathbf{x}}_k^{(-)} \right) \right|\right|_1 \tag{3.61}$$

where $||\cdot||_1$ indicates the L1-norm. Following detection, the faulty set of measurements is rejected, a counter is incremented and the EKF returns to nominal operation. If

no faults are detected in the next two seconds, the counter is reset and the fault is attributed to temporary measurement errors. If the counter keeps being incremented and reaches a value of 100 (threshold determined through simulations), the fault is attributed to be unknown and action is taken (the satellite is commanded to trigger a collision avoidance maneuver if necessary). This technique requires approximately six seconds for isolating hard failures of the navigation system.

Although the filter innovation provides fault detection capability, it does not necessarily provide fault isolation capability, unless the failure has a clear innovation signature. Tests have shown that temporary measurement errors produced by the U/S system do have a clear signature (less than five consecutive bad sets of measurements) observed through the filter innovation, therefore enabling this technique to isolate this type of failure.

Other situations can also lead to the growth of the EKF innovation. When the thruster inputs are used by the global estimator, a thruster failure (either stuck ON or OFF) or external perturbations also cause the EKF innovation to slowly grow. But in these cases, once the total innovation crosses the threshold, it remains high for an extended period of time, at which point an alarm is triggered as described above.

The main advantage of this technique is its integration with the EKF. The main disadvantage is that it is sensitive to many types of failures. The technique presented in the following subsection uses INS data to detect and isolate thruster faults.

3.7.2 Motion-based thruster FDI

The Motion-based thruster FDI module presented in this section is entirely based on the algorithm by Wilson [131, 130] presented in Section 2.4.2. The algorithm was tested on SPHERES, but not as part of an overall architecture. However, it has been implemented in the MATLAB® simulation presented in Section 3.2.3 and successfully used in autonomous docking simulations. Figure 3-22 illustrates the process envisioned for the implementation of the algorithm.

To ensure good acceleration and angular rate readings by the INS, it is highly desirable to require constant thrust in a reading period (e.g., 50 msec). However, a

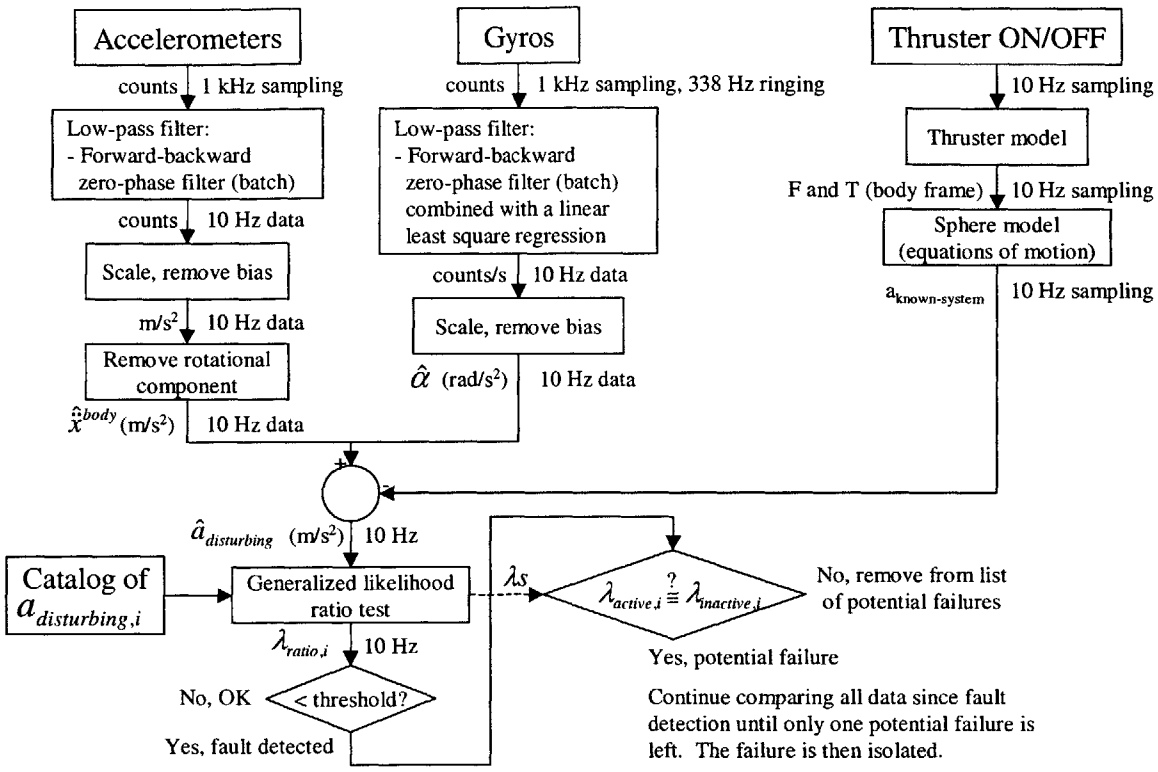


Figure 3-22: Process used for gyro-based thruster FDI.

thrust granularity of 50 msec might negatively affect the overall accuracy of trajectory tracking. Another constraint to verify through experimentation is the computation load. With the global estimator already requiring up to 42% of the total computation time (Section 5.2.1), the execution of the algorithm at 20 Hz might be problematic. Although docking simulations were successful, only experimentation will show if this technique is feasible for the purpose of thruster FDI. The last section focuses on the technique used for managing the sequencing of the different GN&C modes.

3.8 Implementation of mission & vehicle management algorithms

The technique that is currently implemented on SPHERES, to manage the scheduling of the different GN&C modes, is very simple. Each experiment is divided into a series of maneuvers. A high level management module runs a different set of controllers with

a different objective function for each maneuver. It runs at the same frequency as the controllers, in the same primary interface function (*gspControl*). It periodically compares the actual state with the objective until it is met, at which point the mode sequence proceeds with a different maneuver and its corresponding GN&C mode [87].

Mode sequencing is currently prescribed a priori in most cases, except when a CAM is autonomously triggered. This can occur at any point in the experiment, when a problem is detected and the satellite cannot recover in a timely fashion.

Although very simple and easy to implement, this technique has important limitations. For example, it cannot regenerate a new set of maneuvers if an unexpected event, requiring the chaser to take action, occurs halfway through a docking maneuver (it would simply trigger a CAM if necessary). This limits the autonomy of the system. New approaches, requiring a solver to schedule a series of events, have been developed in recent years and could address that problem [61].

3.9 Summary

This chapter first introduced the SPHERES testbed, along with the software used to interface with it, and the different tools that support algorithm implementation on the hardware. Different subsystems, including the navigation subsystem (based on the processing of TOF and gyroscope data) and the control subsystem (that generates thruster commands) were also presented. Details were also provided on the different software modules developed, implemented and used as part of this research.

Overall, this chapter presented some basic software modules that, when properly integrated, can perform complex tasks such as autonomous docking and formation flight. Experimentation, shown in the next chapters, has demonstrated that the combination of the modules presented in this chapter offers enough maneuvering accuracy for experimentation in the ISS and in ground laboratories. These modules are now used on a day-to-day basis on the SPHERES testbed at the MIT SSL.

Chapter 4

Experimental validation of the GN&C modules

Throughout the development and implementation of the GN&C software, some key experiments were conducted to first verify and later validate individual GN&C modules. Because of the unavailability of an *all software* simulation, these experiments used the hardware while it was still under development. The MIT SSL 2-D air table has been an invaluable facility when performing these experiments, by providing the capability to have repeatable testing conditions. This is highly desirable for testing improvements to the state estimators. The KC-135, NASA's previous reduced gravity aircraft, allowed short duration testing in a microgravity environment. Finally, the first few test sessions in the ISS, when only a reduced set of hardware was available (Table 3.3), were utilized to incrementally test the different GN&C modules described in Chapter 3.

This chapter validates the individual GN&C modules through tests conducted in a variety of environments. It is organized as illustrated in Fig. 4-1. Early laboratory experiments are presented along with the major difficulties encountered and their solutions. Although important flight results from KC-135 experiments are shown, the emphasis is placed on flight experiments performed in the ISS, the SPHERES operational environment. These experiments were used to validate the individual GN&C modules prior to using them in an integrated GN&C architecture to conduct

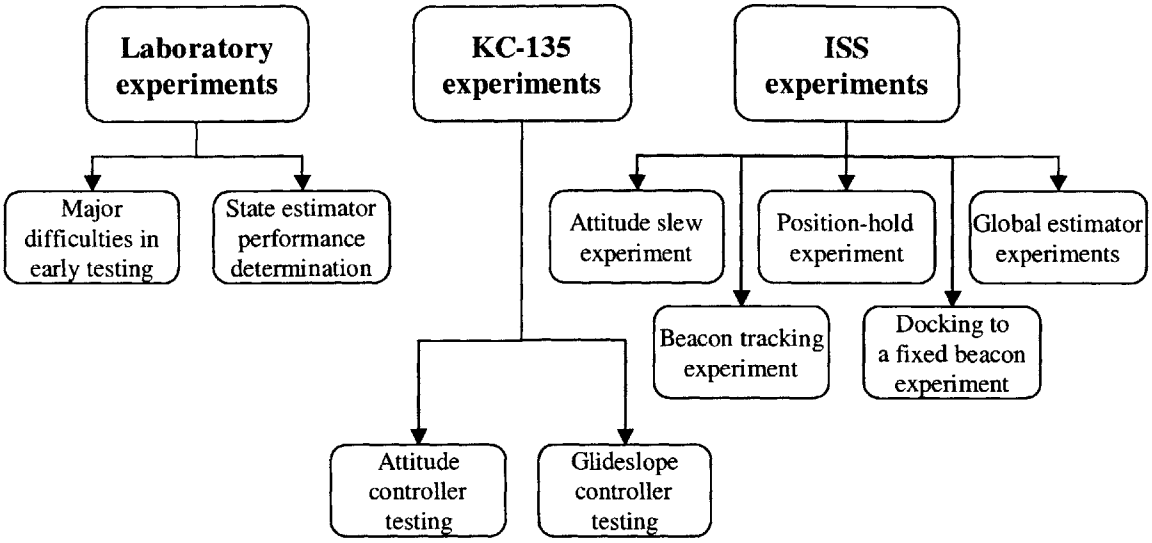


Figure 4-1: Overview of Chapter 4.

autonomous docking experiments. The integration of the GN&C modules and the testing of the integrated GN&C architecture are discussed in Chapter 5. The main outcome of this chapter is a series of flight qualified GN&C modules for on-orbit formation flight and autonomous docking.

4.1 Laboratory experiments

Laboratory experiments, to assist in the development of the GN&C software, were initiated as soon as some flight hardware became available. However, early testing of the GN&C modules, while the hardware and the software interface were still under development, did not always go smoothly. In fact, a large amount of effort was spent on solving various hardware implementation problems.

The most common problem was instability in the state estimates. In response, the controllers produced erratic behavior. This proved to be problematic during close-proximity maneuvers and often resulted in a collision. Consequently, the instability problems with the state estimators achieved high priority. Afterward, multiple static experiments, where the satellites were constrained to remain stationary, were performed to verify the performance of both the global and the relative estimators.

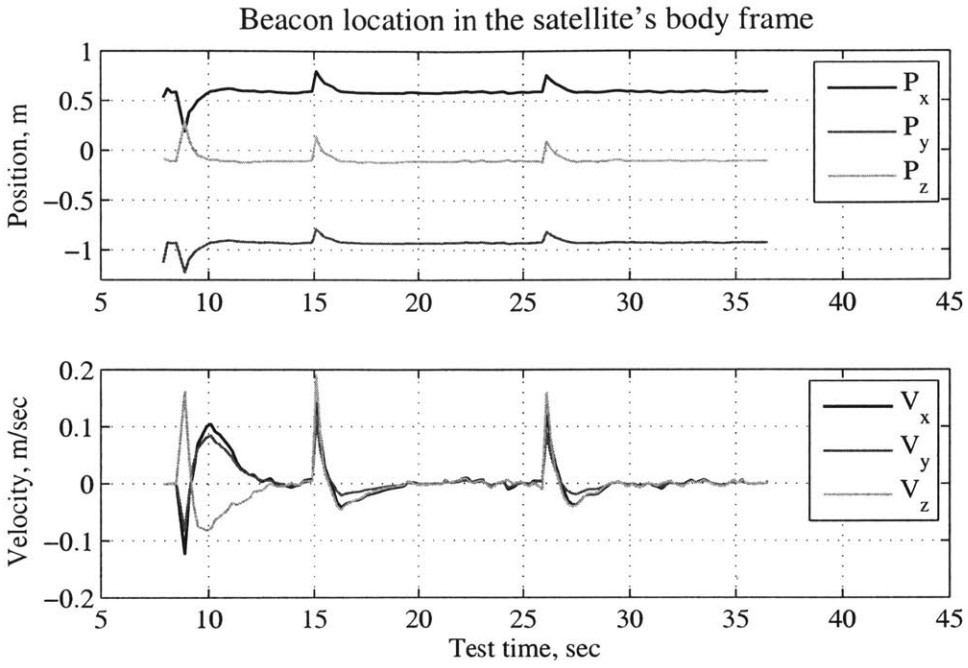


Figure 4-2: Jumps in the state estimates output by the EKF.

This section presents some of the problems encountered with the state estimators. The key experiments used to verify them, after all major problems were fixed, are also discussed.

4.1.1 Major difficulties in early testing

This section presents three major problems with the state estimators encountered at different stages of the implementation. They consist of intermittent jumps in the state estimates, a temporary divergence and the presence of an unmodeled bias in the U/S measurements. Each problem, and its solution, is discussed.

Outliers in the ultrasonic measurements

The first problem observed was the presence of jumps in the state estimates. This problem occurred in both the global and the relative estimators. Figure 4-2 shows the results of an experiment, using the single beacon estimator, that was performed with the satellite stationary. The position estimates seem to initially converge to

the correct solution. The true position of the external beacon was approximately x=0.6 meter, y=-0.9 meter and z=-0.1 meter. Position errors on the order of 0.2 meter, and velocity errors on the order of 0.2 m/sec, can clearly be observed at 9, 15 and 26 seconds. It is interesting to note that the corresponding covariance plot (not shown) does not exhibit any jump, meaning that the filter interpreted the bad measurements as accurate.

The first attempt to solve this problem added a condition before accepting a new state estimate. The condition stated that the new state estimate must not differ from the previous estimate by more than a certain limit. The determination of that limit posed a problem: setting it too high would still allow small jumps to occur, while setting it too low would also reject legitimate state updates. After many attempts, it was concluded that this technique could not ensure smooth state estimates for a period of approximately three minutes, the expected duration of a typical SPHERES experiment. It was therefore rejected and other solutions were pursued.

Analysis of the raw data showed that the problem was caused by outliers in the U/S measurements. The model used in Eq. (2.2) to derive the general EKF equations assumes that each measurement is approximately normally distributed with zero mean and a covariance $\mathbf{R}$. The presence of outliers contradicts that statement.

The source of the outliers in the measurements varies. On some occasions, a fraction of the U/S receivers would pick up a signal that reflected off an object near the satellite. This problem is analogous to the multi-path problem that GPS systems have in urban areas. Other times, the receivers directly facing the beacon would not pick up the signal, although receivers on other faces of the satellite would. Since the receivers do not work well when their axis is offset more than 45 degrees from the direction to the beacon, the TOF data passed to the EKF presented unexpected and unmodeled variations.

Because this problem could not be easily addressed by modifying the hardware, it had to be fixed in the software. The first solution was to design a pre-filter, based on heuristic rules, that remove the outliers among the 24 measurements collected for each beacon ping. The measurements are grouped according to the satellite face on

which their corresponding U/S receiver is located (Fig. 3-2a). After experimenting with many different rules, the following were adopted:

1. The pre-filter discards measurements with ranges larger than the dimensions of the test volume (>4 meters) and less than the minimum separation between a beacon and a microphone (<1 cm).

2. For each pair of opposing faces, it only keeps the measurements taken on the face closest to the beacon.

3. On each face, if any two measurements differ by more than 13 cm (slightly over the maximum distance between two receivers), it tries to identify the faulty receiver.

    - If the identification is successful, it discards the measurement output by the faulty receiver.
    - If not, it discards all four measurements on that face.

4. It determines the angle between each receiver and the line-of-sight to the beacon. If that angle exceeds 35 degrees, it discards the corresponding measurements.

5. It discards measurements with a range at least 25 cm higher than the smallest one initially recorded among all 24 measurements, since the satellite has a diameter of 20 cm.

6. If fewer than three valid measurements are collected on one face, it discards all four measurements on that face.

7. Finally, among the remaining measurements, it keeps only the ones corresponding to the face closest to the beacon (which presents the lowest TOF data).

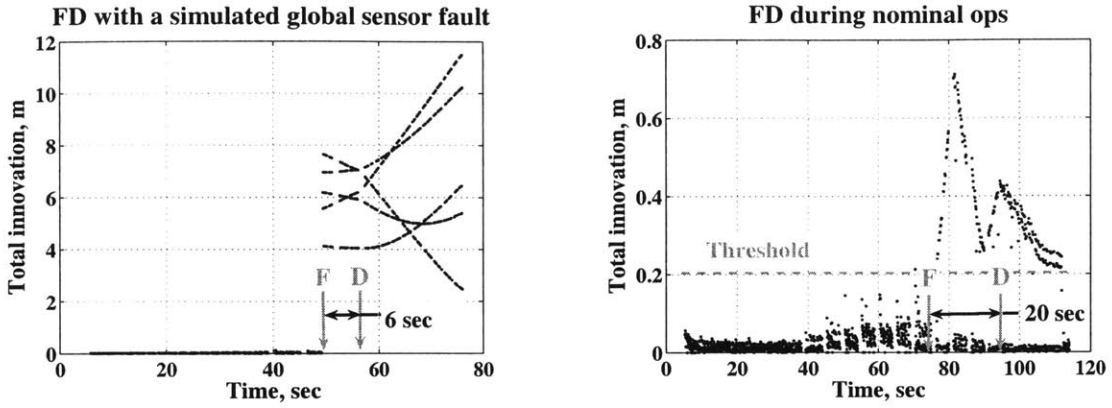
This method, of pre-filtering the raw data, has been effective in removing outliers. It follows the principle that it is better to disregard good data than keep bad data. At most four measurements, out of a total of 24, are passed to the EKF at the end of the process. However, it requires a fair amount of computational time (up to 17% of

the total computational load of the global estimator, Fig. 5-2) and needs to be coded efficiently to avoid looping many times through the same data.

After implementing this pre-filter, one form of multi-path remained problematic, and could not be fixed with the rules described above. When an U/S pulse travels for approximately 3.5 meters, gets reflected and travels another 3.5 meters before being received, the corresponding TOF registered is between 20 msec and 25 msec. However, with a time delay of 20 msec between beacon pings (initially thought to be long enough for the signal to dissipate), this measurement overlaps with the listening window associated with the next beacon in the sequence (Fig. 3-7). The computer treats that signal as if it is a ping from the next beacon, with a TOF between 0 and 5 msec. A known fix for this problem is to increase the time delay between two beacon pings by changing the beacon addresses as shown in Fig. 3-8. But this solution is only possible when a maximum of five beacons are used. An alternative was to use a modification to the software. This solution was only found after the first experiments in the ISS.

During ISS Test Session 04, other jumps were observed in the state estimates when two satellites were in close proximity. Since no IR noise was present in the testing environment, the source of the problem was believed to be multi-path caused by U/S reflection off of the second satellite. This problem led to a second software improvement. It was decided, since the pre-filter could not remove all outliers, to implement a fault detection (FD) technique through analysis of the filter innovation as described in Section 3.7.1.

Experiments were performed on the MIT SSL 2-D air table to verify the performance of the implemented FD technique. The results are shown in Fig. 4-3. In the first experiment, the TOF data passed to the EKF were intentionally corrupted to simulate a hard failure starting at time 49 seconds (arrow labeled F). The technique used to corrupt the data is illustrated in Fig. 4-4. At the flip of a software switch, the data collected from the U/S sensors is overwritten by corrupted data that was hard coded in the program. The total filter innovation is shown in Fig. 4-3a (each of the five traces correspond to data recorded from a different beacon). As soon as the failure



(a) With a simulated fault.

(b) With an actual multi-path fault.

Figure 4-3: Measurement rejection system based on filter innovation.

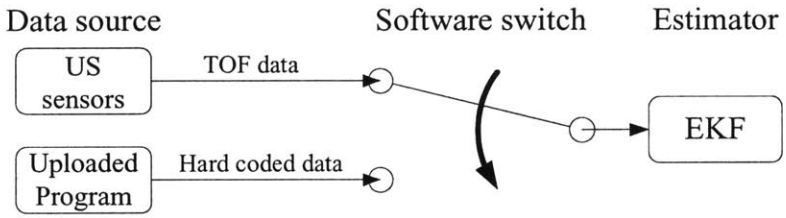


Figure 4-4: Technique to corrupt the raw U/S TOF data.

is simulated, the registered innovation immediately jumped above the threshold set at 0.2 meter. Six seconds later (arrow labeled D), because the innovation remained above the threshold, an alarm was sent to MVM module, confirming that a fault was detected. Although the experiment was successful following the triggering of the alarm, the method used to corrupt the data was suspected to have been unrealistic. Therefore, more realistic experimentation was needed to verify this technique.

The second experiment involved moving an obstacle close to the satellite to generate measurements that are corrupted by multi-path. The results are shown in Fig. 4-3b. This time, the response following the approach of the obstacle was believed to be more realistic. The alarm was sent approximately 20 seconds (arrow labeled D) after multi-path occurred (arrow labeled F), which successfully verified the technique.

This technique was then implemented in all experiments using the global estimator. Figure 4-5 illustrates the total filter innovation from an experiment performed in the ISS on November, 2006. The four faulty measurements are easily distinguished,

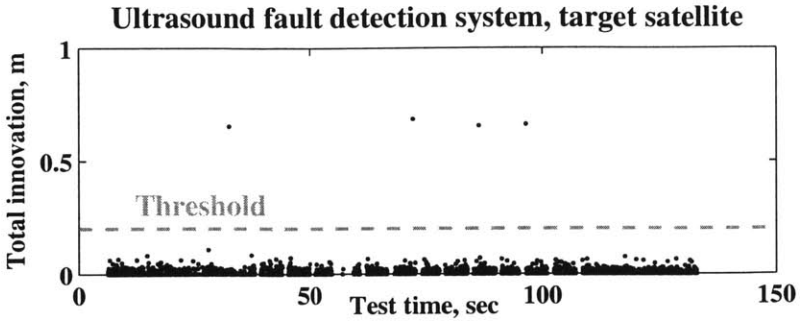


Figure 4-5: Total innovation recorded during ISS operations.

as they clearly have a combined innovation well above the threshold. Similar results were observed in other experiments.

Following the implementation of the pre-filter, together with this fault detection technique, no single jump in the state estimates of any satellite was observed during the next two test sessions in the ISS.

Temporary divergence of the state estimates

The second problem to be addressed was an occasional and temporary divergence of the state estimates as shown in Fig. 4-6. This problem was only affecting the global estimator. It led to collisions on multiple occasions. After a few seconds, the state estimates would converge to reasonable values.

A detained examination of the EKF was performed. This time, the covariance matrix did provide some clues about the source of the problem, as it became non-positive definite just prior to the divergence. More precisely, the source of the problem was found to be the portion of the covariance matrix that describes the uncertainty associated with the quaternions. Since the state vector used by the EKF for the relative estimators (Section 3.4.2) does not include quaternions, this problem did not appear in its estimates.

To solve the problem, the reduced version of the covariance matrix, introduced in Section 3.4.1 and proposed by Lefferts, Markley and Shuster [71], was implemented. This solution significantly increased the robustness of the global estimator at virtually no computational cost. After the implementation of this solution, no temporary

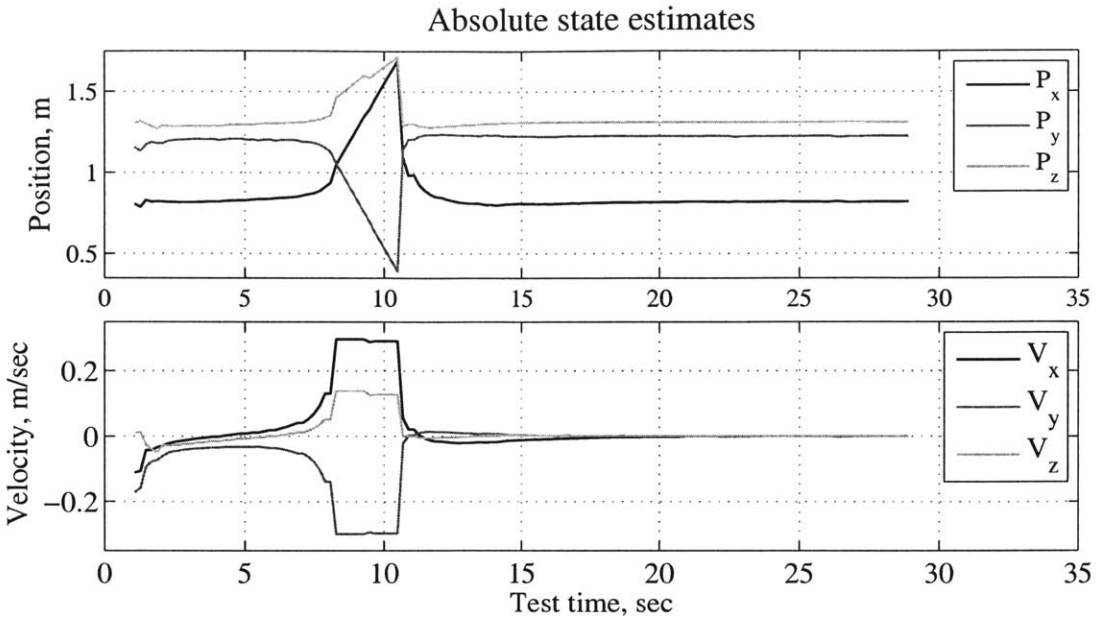


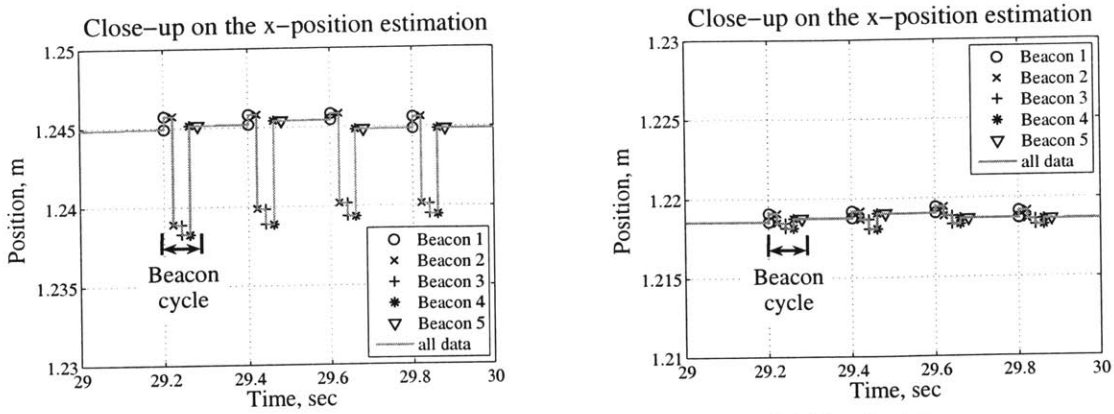
Figure 4-6: Temporary divergence of the state estimates output by the EKF.

divergence of the state estimates, similar to the one shown in Fig. 4-6, was ever observed.

Unmodeled bias in the ultrasonic measurements

The next step was to correct global state estimates during the state update phase of the EKF. Figure 4-7a shows the results from processing raw data collected with a stationary satellite using a MATLAB® version of the global estimator. It represents a close-up view of the absolute x-position estimate ($\hat{\mathbf{r}}_x$). The corrections applied to $\hat{\mathbf{r}}_x$ during a period of one second are clearly visible. Four beacon sampling cycles are shown. For each cycle, the x-position is corrected five times, every time a set of TOF measurements from one of the five beacons is processed.

During the calibration of the U/S system (Appendix B), it was found that the corrected distance measured by a receiver is accurate to within 6 mm over the operational range of the system (three meters separation between a beacon and a receiver, as shown in Fig. B-13). The variations shown in Fig. 4-7a, on the order of 8 mm during a single sampling cycle, are too large to be considered normal.



(a) Jumps with a misplaced beacon.

(b) Nominal jumps.

Figure 4-7: Jumps in the absolute x-position estimate during a series of U/S sampling cycles.

A close investigation revealed the source of the problem: the fourth beacon was misplaced by about 10 cm from its nominal location. Inputting the correct beacon location led to the results shown in Fig. 4-7b, which represents normal and expected variations. The actual estimates changed by more than 2 cm (from $\approx$1.245 meters to $\approx$1.219 meters). This new estimate is expected to be more accurate since there is a better agreement between the measurements. A biased beacon location can only be observed when using multiple beacons or when a truth measure is available. Therefore, it was not observed in the state estimates produced by the relative estimators.

This experience shows that accurate knowledge of the location of the beacons is essential to obtaining accurate state estimates. It also suggests that a change in the state estimates, during a single state update cycle (Fig. 4-7a), that is larger than the U/S resolution is an indication of a bias in the measurements, which can be caused by a misplaced beacon.[1]

Summary of the lessons learned

All of the problems presented in this section were solved after implementing the solutions described. Important lessons were learned after discovering the nature of

[1]Motion of the satellite would not cause such a large repetitive pattern in the position estimates during a beacon sampling cycle.

each problem. They are listed below.

- It is crucial to clearly understand the behavior of the navigation sensors in their operational environment after they are integrated into the system. The integrated sensors might show a behavior that is not predicted by the manufacturer's specifications.
- When verifying a GN&C module, it is important to experimentally check every assumption made associated with the algorithm. Numerical errors caused by the software implementation of the algorithm might sometime invalidate some assumptions.
- Regular jumps in the state estimates at the measurement update phase of the navigation filter are likely to be caused by unmodeled biases in the measurements.

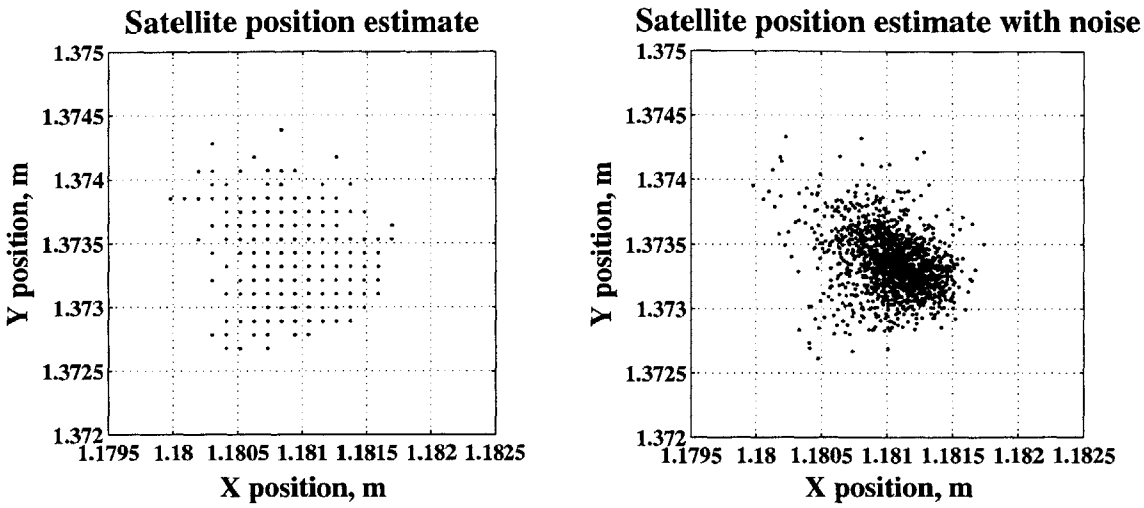
4.1.2 State estimator performance

Early experiments performed on the MIT SSL 2-D air table provided useful data supporting the development and implementation of the various GN&C modules. However, the modules that benefited the most from experiments at MIT were the state estimators. On the air table, it is possible to constrain the motion of the satellite simply by not floating the air carriage. This provides repeatable testing conditions and a truth measure on the velocity (0 m/sec) and the angular rate (0 rad/sec). These testing conditions are particularly hard to achieve in the KC-135 and in the ISS.

This subsection covers two experiments performed on the MIT SSL 2-D air table that greatly benefited from the testing conditions described above. They were used to determine the performance of both the global estimator and the EKF in the relative estimators.

Global state estimator performance

To get an idea of the precision provided by the global estimator, a series of four experiments were performed with a stationary satellite positioned in the middle of



(a) Raw estimates.

(b) With a little noise added.

Figure 4-8: Position estimates of a satellite in the middle of the test area.

the test area on the MIT SSL 2-D air table. The orientation of the satellite was changed between each experiment. The U/S beacons were sampled at 5 Hz and the gyroscopes at 1 kHz (but batch filtered and downsampled at 20 Hz).

Figure 4-8a shows the estimated x-y position of the satellite during one of the experiments, after letting the filter converge for ten seconds. The test lasted a total of five minutes with the estimates being downloaded at 5 Hz through the background telemetry. All the position estimates are contained in a 2 mm x 2 mm box. Because they are downloaded with a numerical precision of only 0.1 mm (they are downloaded as shorts, not floats), many data points are overlapping. To discriminate between overlapping points, and get a better idea of the distribution, a slight noise was added to each point. The results are shown in Fig. 4-8b. The majority of points are, in fact, contained in a 1 mm $\times$ 1 mm box.

Using the same set of data, the bandwidth of the global estimator was approximated as follows. Among all the states, the slowest ones to converge are the velocity estimates. This is explained by the lack of a sensor that directly measures the velocity. Figure 4-9 presents the velocity estimates computed during the first fifteen seconds of an experiment. Both the x- and y-velocity estimates reach their peak ($\approx$25 cm/sec) at time $t \approx$0.4 sec. For a first order system, the cutoff frequency can be approximated

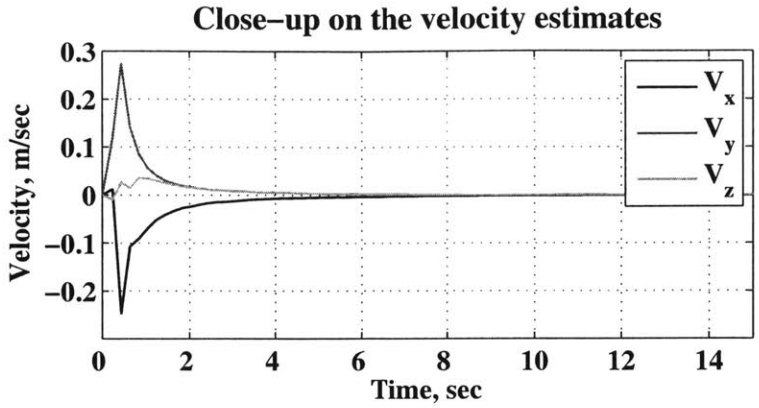


Figure 4-9: Convergence of the velocity estimates.

by $1/\tau$, τ being the time constant of the system. Under the rough approximation that the convergence shown in Fig. 4-9 is the step response of a first order system, a time constant $\tau \approx 0.4$ sec can be easily calculated (63% of the attenuation is reached at time $t \approx 0.8$ sec). The estimator bandwidth can then be approximated at 2.5 Hz, the theoretical limit for a system with a sampling frequency of 5 Hz. Although probably not very accurate, this approximation indicates that with the current level of filtering, the bandwidth of the system is not too greatly affected. A more accurate bandwidth measure can be obtained by analyzing the frequency response of this nonlinear filter at different position, attitude and rotational rates.

All four experiments (for a total of twenty minutes of testing) presented characteristics similar to those shown in this section. No deviations were observed, confirming the stability of the filter. The estimated position is contained within a 2 mm $\times$ 2 mm box and the approximate bandwidth remained similar for all four experiments. Although the location and orientation of the beacons in the ISS differ, the results of this experiment are believed to provide a good indication of the potential performance of the global estimator in the ISS.

Performance of the EKF used in the relative estimators

An experiment with a stationary satellite was performed to determine the precision of the position estimates output by the relative estimator. The results are shown in

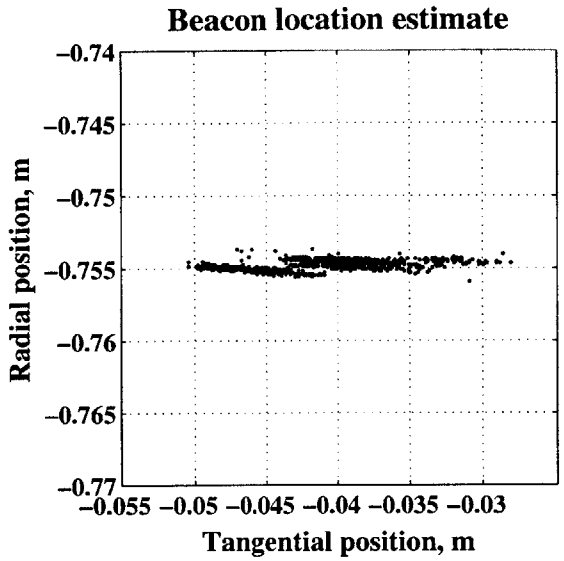


Figure 4-10: Beacon location estimate in the satellite's body frame.

Fig. 4-10. The beacon (also stationary) was located directly in front of the docking face (-x face), approximately one meter away. It was sampled at 5 Hz throughout the experiment. The radial position of the beacon is parallel to the x-body axis, while the tangential position corresponds to the misalignment parallel the y-body axis. This experiment was setup such that the range vector is approximately parallel to the x-body axis. A precision of ± 1 mm along the radial direction (x-body axis), and ± 1 cm along the tangential direction (y-body axis), can be observed. Since the estimate along the y-axis is obtained by the differentiation of range measurements from the U/S sensors on the four corners of the face in front of the beacon, it is expected that the precision along the tangential direction would be less than along the radial direction. With an alignment tolerance of 1.5 cm on the Velcro docking port, the results in Fig. 4-10 show enough accuracy to allow docking.

4.2 KC-135 experiments

From the very beginning of the SPHERES project, the team was given multiple opportunities to perform experiments in a microgravity environment onboard NASA's KC-135. The KC-135 (now replaced by the C-9B) is a reduced gravity research air-

craft that provides a short duration, reduced gravity environment by flying through parabolic trajectories. Although NASA claims a zero-g period of a little over 20 seconds, only a period varying between 5 and 10 seconds of continuous microgravity is realistically available to a free flying payload the size of a SPHERES satellite before it hits the wall of the aircraft. This is the result of the motion of the aircraft caused by the perturbations to its flight path.

Nevertheless, the short duration, microgravity environment available onboard the KC-135 allowed the SPHERES team to perform experiments that provided extremely useful data (Fig. 4-11). Most hardware components (especially the thrusters and the accelerometers) were successfully validated, since the microgravity environment onboard the KC-135 is very similar to the one onboard the ISS. These experiments also gave the SPHERES team a chance to experience microgravity operations, which proved to be invaluable in preparing the operations procedure for ISS.

However, the short duration of the microgravity periods prevented the validation of the GN&C modules. Moreover, following early experiments, it was concluded that the global estimator, with the five external beacons attached to the wall of the aircraft, could not provide useful state estimations to the satellites. IR noise, present in the testing environment, constantly corrupted the state estimates. Also, the bandwidth of the global estimator was too low for the frequency content in the motion of the satellite free floating inside the aircraft, which was submitted to external perturbations. Therefore, only the onboard beacons and the INS were usable for performing experiments in the KC-135.

Consequently, a series of simple and short experiments were designed to collect preliminary results on some of the GN&C modules that would play a critical role in future ISS experiments. These flight experiments provided the first chance to test these modules in a realistic microgravity environment, where six DOF motion is permitted. The results from two important experiments, each involving two satellites, are presented below. The first one was used to test the quaternion-based standard attitude controller in a 3-D environment, something not possible on the MIT SSL 2-D air table. The second one tested the combination of the attitude controller with the

Figure 4-11: SPHERES experiments inside NASA's KC-135.

glideslope controller in an autonomous docking maneuver.

4.2.1 Attitude controller testing

The first experiment started with the initialization of a relative state estimator on each satellite to locate the beacon of the other satellite. After an estimator convergence period of one second, when no thrusters were actuated, each satellite was commanded to point its beacon in the direction of the other satellite using a PD attitude controller coupled with the pulse-width modulator. This experiment had two main objectives:

- to collect preliminary data that could assess the performance of the quaternion-based attitude controller, and
- to demonstrate the tracking of a beacon by a satellite rotating along the shortest rotational path (generally not around a body axis).

Results for both satellites are shown in Fig. 4-12. The top plots represent the quaternion data, while the bottom plots show the body rates read from the gyroscopes. Two separate tests were performed during this zero gravity phase. Each test terminated when a satellite reached the limit of the test volume. It is important to note that the short duration of each test ($\approx$4 seconds) is the result of perturbations of the KC-135 due to poor weather, rather than of thruster actuation.

Inspection of the results shows that the pointing error is reduced over time as indicated by the q_1, q_2 and q_3 terms driven toward zero for both satellites, and q_4

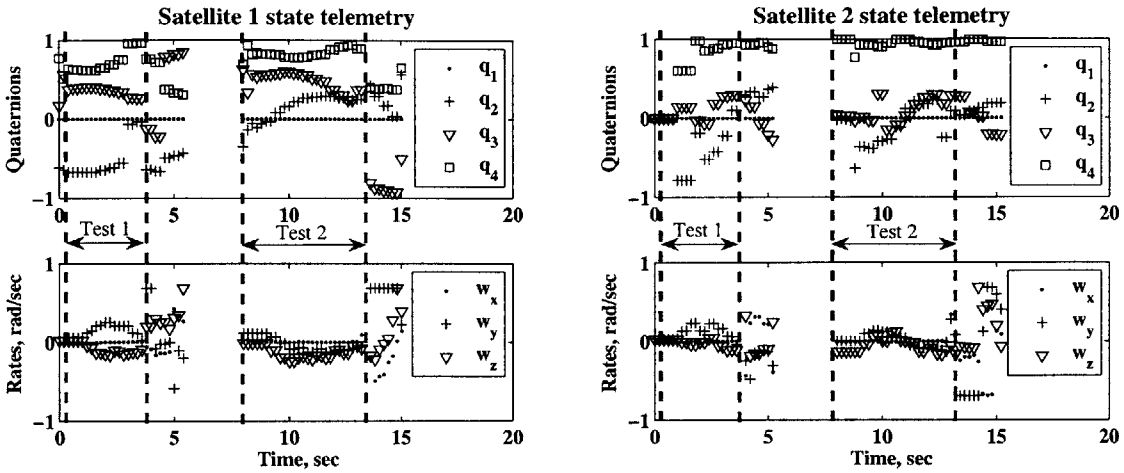


Figure 4-12: Beacon tracking experiment results: KC-135, November, 2003, Flight 4, Parabola 18.

toward one. As expected, the body rates show an initial angular acceleration followed by an angular deceleration as the pointing error decreases. These preliminary results showed that the relative estimator and the PD controller, coupled with the pulse-width modulator, operated successfully in a 3-D environment. The video of the experiment confirmed that the angular rotation followed the shortest path to the target. Unfortunately, because of the short duration of each test, no data could be collected on the pointing accuracy of the algorithm, or on the behavior of the algorithm in steady state.

4.2.2 Glideslope controller testing

The second experiment consisted of an autonomous docking maneuver. The objective of this experiment was to demonstrate the initialization of a basic docking maneuver in a 3-D environment using the glideslope controller. The short duration of microgravity time available during each parabola limited the demonstration to initial attitude hold and the initialization of a translation by one satellite towards the other. The maneuver was programmed as follows:

- The two satellites maintain their orientation while pointing their x-body axis toward each other using the attitude controller previously demonstrated.

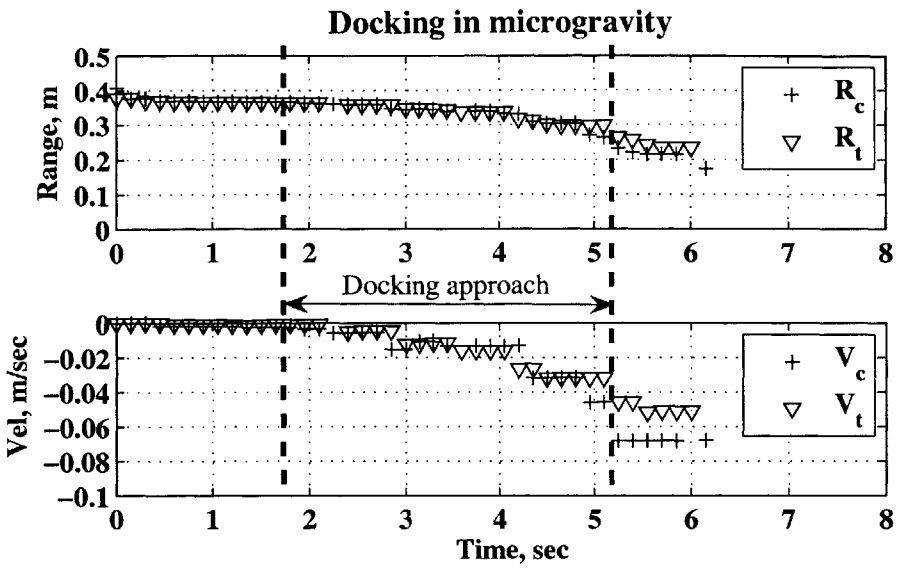


Figure 4-13: Glideslope docking experiment results: KC-135, November, 2003, Flight 4, Parabola 19.

- The chaser initiates a translation along its x-body axis to move toward the target.

- The algorithm is set such that the docking maneuver is completed in eight seconds, which is at the limit of the hardware capability for the desired initial separation of the two satellites but still outside the limit of what can be reasonably achieved onboard the KC-135.

To perform that maneuver, the relative estimator, the glideslope controller and the previous PD attitude controller were integrated into one docking algorithm. This algorithm was designed to be very aggressive, such that as much of the docking maneuver could be completed within each microgravity parabola. Because of poor weather, only four to six seconds of microgravity were available before one of the two satellites would hit a wall. Figure 4-13 shows data for both satellites collected during a successful test of the first phase of the glideslope docking algorithm. The plots represent the range (in meters) and the range rate (in m/sec) estimated by both the chaser and the target.

A period of approximately two seconds at the beginning of each test was used to initialize the relative estimator. Then, thrusters were enabled and the docking

approach was initiated. For the test shown in Fig. 4-13, the first 3.5 seconds of the docking maneuver were successfully achieved. The two satellites kept pointing at each other throughout the approach segment. The range plots show decreasing range to target and the velocity plots show increasing velocity. The data collected by each of the two satellites confirm the results. These results are encouraging, although the microgravity environment in the ISS is required to fully validate the glideslope controller in a 3-D environment.

Tests performed in the ground laboratory and onboard NASA's KC-135 have confirmed the functionality of SPHERES as an autonomous docking and formation flight testing platform with dynamics representative of true spacecraft. The GN&C modules used in the experiments performed in the KC-135 were refined in preparation for the ISS experiments presented next.

4.3 ISS experiments

Throughout 2006, a series of five test sessions were performed in the ISS (May 18, May 20, August 12, August 19, November 11). Each test session built upon the results of previous test sessions, incrementally adding complexity and capability to the experiments being performed. Because controllers designed for a microgravity (six DOF) environment can only be validated to TRL-6 through operations in such an environment, the first experiments performed in the ISS were designed to validate the individual software modules that could not be validated through ground experiments (Fig. 4-14). Each was designed to test one new module at a time, such that if there would be a problem with a particular module, it would be easy to identify by analyzing the telemetry.

This section covers some of these experiments by describing them with emphasis on the most important flight results. The order in which the experiments are presented does not exactly correspond to the chronological order in which they were performed. The first experiment is a series of attitude slews, demonstrating the standard quaternion-based attitude controller. The second experiment provided the first

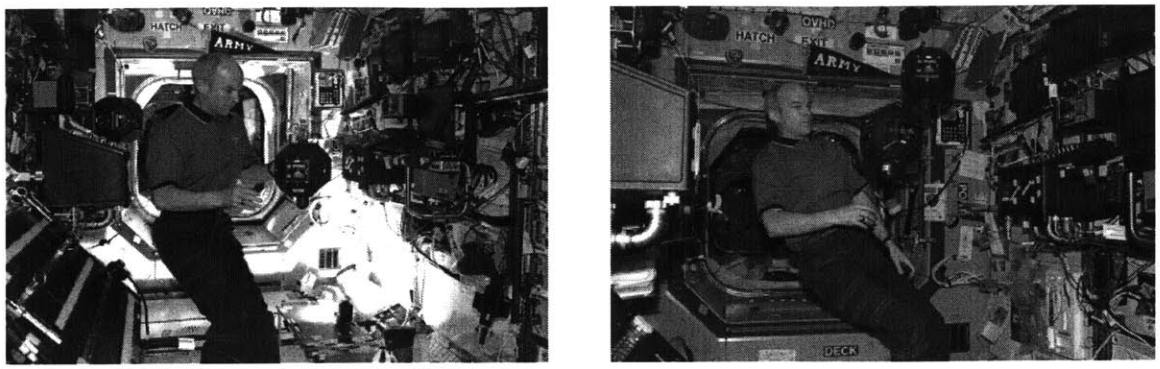


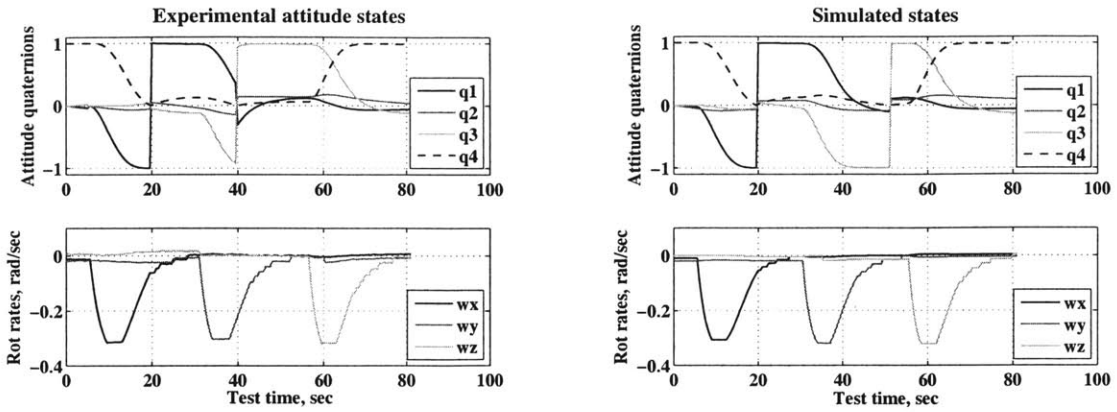
Figure 4-14: SPHERES experiments inside the ISS.

state estimates computed using data from the U/S-based navigation system using the relative estimator. The third is a position-hold experiment, where the satellite held its position with respect to a beacon attached to the wall of the ISS, to validate the position controller. The fourth is a docking maneuver with a beacon attached to the wall of the ISS, which was used to validate the glideslope controller. Finally, the last two involved two satellites in close proximity, estimating their states using the standard global estimator.

4.3.1 Attitude slew experiment

The objectives of this experiment were to validate the standard PD-type attitude controller in a six DOF environment and to collect data to validate the models (the integrated thruster, sensor and dynamics models) used in performing ground simulations of the same controller. Because it was the first time the SPHERES team performed closed-loop experiments in a long duration, microgravity environment, only gyroscope data was used in the feedback loop, so as to keep the experiment as simple as possible. The maneuvers consisted of three 180 degree rotations around each satellite body axis, with the spacecraft actively controlling its angular rotations based on integrating data received from its gyroscopes.

The experiment was attempted during the very first test session in the ISS. However, a flash memory problem, affecting the gyroscope scaling factors, corrupted the estimated rates measured by the gyroscopes and the test failed. After correcting the



(a) Experimental states.

(b) Simulated states.

Figure 4-15: Results for the attitude slew experiment.

Table 4.1: Final attitude error after completing the three rotations.

	x-axis	y-axis	z-axis
Experiment final error (deg)	-6.0	4.6	-13.1
Simulation final error (deg)	-6.6	11.1	-14.7

problem, the experiment was successfully repeated during the second test session.

The resulting attitude states (attitude quaternions and angular rates) are shown in Fig. 4-15a. The three consecutive rotations are clearly visible in the plots. The attitude quaternions show an overshoot of ≈20 degrees[2] after the first rotation (at time 30 seconds), which was also mentioned by the in their notes. The second rotation ended with the satellite being within ≈7 degrees of its desired attitude (at time 55 seconds). The third rotation brought the satellite within ≈15 degrees of its initial orientation with the final attitude error around each axis shown in Table 4.1.

Figure 4-15b shows results from a ground simulation of the same experiment that was performed a few weeks later. The similarity of both simulated and experimental data is obvious. The simulated data also shows an overshoot after the first rotation and a final attitude error of ≈20 degrees. These errors fell within expected values given

[2]The author uses, in its description, the magnitude of the attitude error (θ_e) in degrees, since it is easier to conceptually understand than quaternions. However, the plots show the quaternions to present to the reader the data sent through the telemetry. The translation between both formats can easily be performed using q_4 in Eq. (3.2).

the parameters used in the controller as well as the modeled hardware imperfections, such as thrust and sensing noise (Appendix B and Refs. [12, 19]).

The success of this simple experiment validated the attitude controller in a six DOF environment. The results allowed the tuning of the controller gains through simulation to reduce the attitude error in future experiments down to an acceptable level for docking experiments (<5 degrees). This process supports the idea of performing *incremental maturation* of the GN&C architecture [87, 113]. The models used in the ground simulation were also validated by the close resemblance between the simulated and the experimental results. This experiment fulfilled all of its objectives.

4.3.2 Beacon tracking experiment

During the first test session in the ISS, a series of beacon tracking and beacon docking experiments were also attempted. These experiments were designed to test the U/S navigation system (including the relative state estimator providing range and bearing information) and the capability to maneuver the satellite.

Figure 4-16 shows the state estimates collected during a beacon tracking experiment. The range, range rate (velocity), total attitude error (from the fourth quaternion) and the first three attitude quaternions are displayed. The dotted line in the first three plots indicates state estimates collected during the initialization phase of the experiment (thrusters disabled), while the solid line indicates state estimates during the attempted tracking. Although the satellite did not track the beacon because of the problem with the rates read from the gyroscope data (Section 4.3.1), the results were correlated with a video of the experiment and demonstrated the ability to operate the SPHERES global metrology system aboard the ISS. The range measurement, which depends solely on the U/S data, presented encouraging results. The state estimates are rather clean after a five second initialization period. The quaternion (bearing) estimates, which do not depend on the rates, but only on the U/S navigation system, present reasonable estimates which reflect what is observed in the video (in terms of both the total attitude error and its distribution along each body axis shown by the first three quaternions).

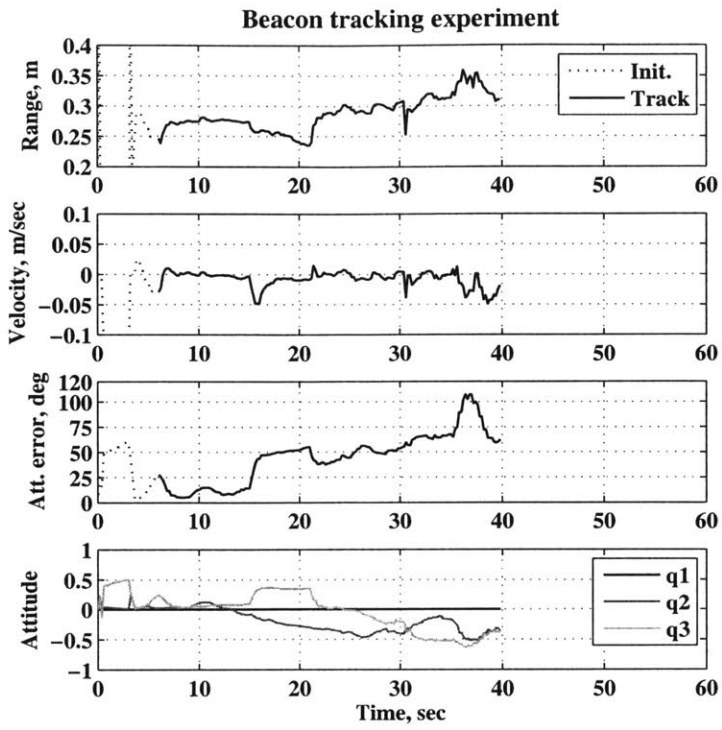


Figure 4-16: Results for the beacon tracking experiment.

For the duration of the test, the perturbations observed in attitude data are caused by actual motion of the satellite. The noise in the range data can also be correlated, in the video of the experiment, to the motion of the U/S beacon being held by the crew. Although the control capabilities could not be tested because of the flash memory corruption described above, the U/S navigation system and the relative estimator were successfully validated, using the video of the experiment as a truth measure of the general orientation and location of the satellite with respect to the beacon.

4.3.3 Position-hold, single beacon estimator experiment

Following the repair of the corrupted flash memory and the successful validation of both the attitude controller and the relative estimator, a position controller was tested. The main objective of this experiment was to validate a PD position controller that would become the controller of choice to hold a position or maintain alignment in front of the target's docking face. Secondary objectives were to collect data to

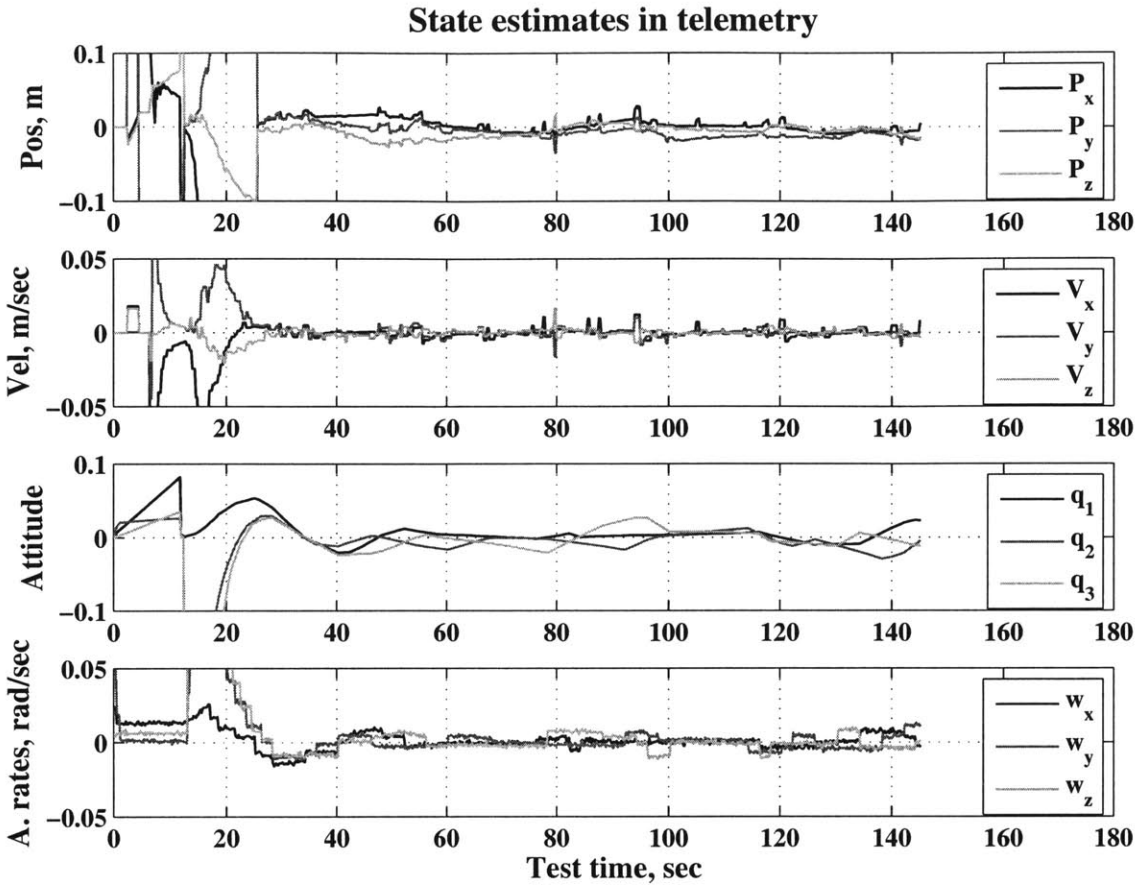


Figure 4-17: Results for the position-hold experiment with a fixed beacon.

tune the controller and to validate the single beacon estimator (Section 3.4.2), the variant of the relative estimator which provides six DOF state estimates using only the single beacon. Since it is based on the same EKF as the one validated in the previous experiment (Section 4.3.2), the risk associated with validating it, along with the position controller, in a single experiment was judged acceptable.

The same attitude controller, as the one validated in Section 4.3.1, was used to point the docking face at the beacon (initialization phase) and to maintain the attitude fixed throughout the remainder of the experiment. The position controller was used to maintain the satellite in place after it pointed its docking face at the beacon.

The results are shown in Figs. 4-17 and 4-18. The top plot in Fig. 4-17 shows the displacement of the satellite with respect to its reference position. The reference

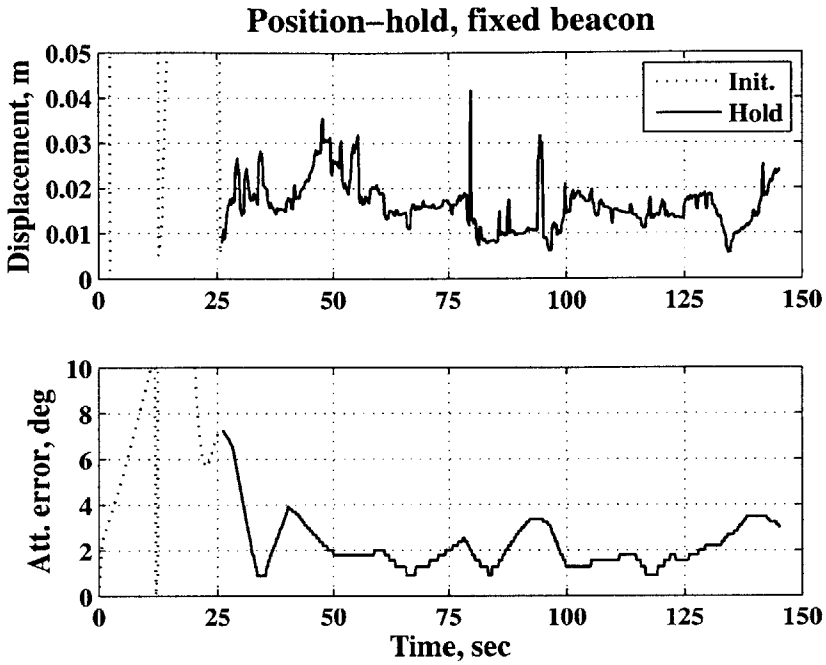


Figure 4-18: Closer view on the results for the position-hold experiment.

coordinate frame is approximately oriented such that the x-axis is collinear with a vector defined from the center of the satellite to the beacon. The drift associated with the gyroscope causes this reference coordinate frame to slowly change its orientation as the gyroscope data is integrated. The second plot shows the velocity. The third plot presents the attitude quaternions obtained by integrating the gyroscope data (as explained below) and the last plot shows the angular rates directly read from the gyroscopes. From time 25 seconds until the end of the test (two more minutes) the satellite actively maintained its attitude and position, as confirmed by all the states in Fig. 4-17 remaining constant at zero. The first plot in Fig. 4-18 presents a closer view of the magnitude of the displacement estimates, while the second plot provides an estimate of the pointing accuracy obtained from the fourth quaternion.

The experiment proceeded as follows. The integration of the gyroscopes was initiated as soon as the test was started. The single beacon estimator was initialized after two seconds. A period of 10 seconds was allowed for the estimation algorithm to converge. At time 12 seconds, an estimation of the beacon pointing attitude error was computed and the slew was executed. It took approximately thirteen seconds for

the satellite to achieve that change of orientation and point at the beacon. This new orientation became the reference orientation for the remainder of the test. At time 25 seconds, the position of the beacon in the satellite coordinate frame was recorded as the reference position. The subsequent displacement shown in the top plot of Fig. 4-17 is calculated with respect to that reference position.

This experiment was very successful. During the holding phase, the magnitude of the displacement vector remained within 3 cm for most of the time while the magnitude of the attitude error vector remained within four degrees (except for the first five seconds). Although no truth measures could directly confirm the state estimates shown in Figs. 4-17 and 4-18, the video of the experiment showed that the satellite did hold its position and attitude as expected. The state estimates are quite smooth for the duration of the test, with no apparent jumps. The results of this experiment successfully validated both the PD position controller and the single beacon estimator. Valuable data was also collected to tune the different gains used by the position controller. Similar performance of the estimator was observed in subsequent experiments.

4.3.4 Docking to a fixed beacon experiment

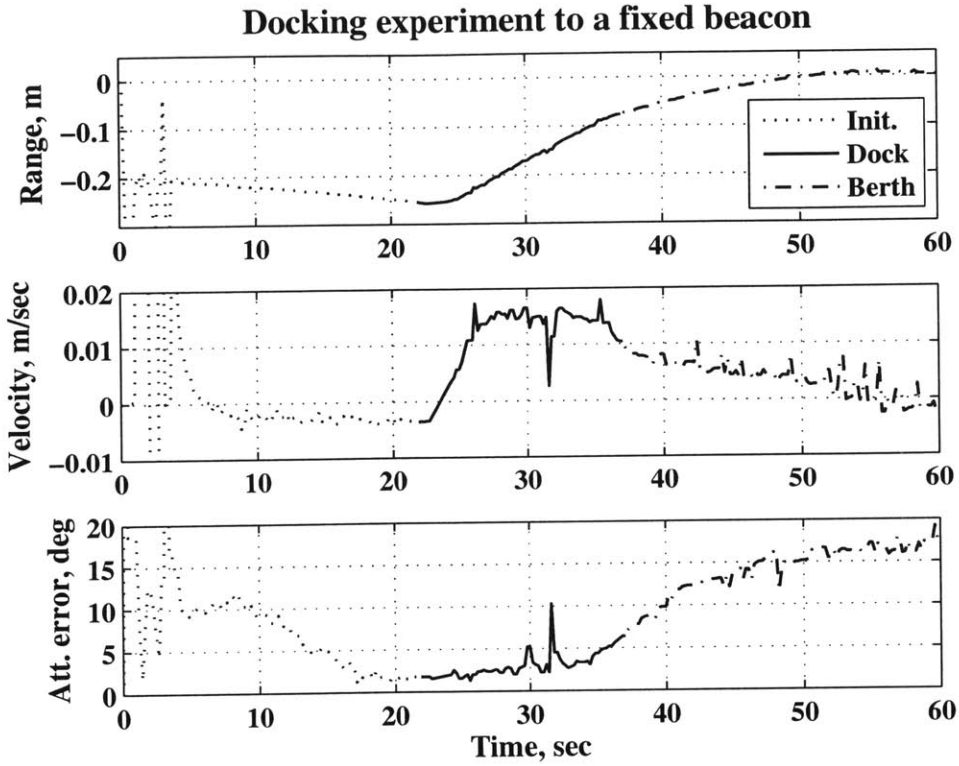
Even with the limited hardware available aboard the ISS during the second test session on May 20, 2006 (only one satellite, one beacon and few other accessories), the SPHERES team attempted a preliminary docking experiment by having a satellite follow an approach trajectory to a beacon mounted to the wall. The purpose of that experiment was to validate, through on-orbit testing, not only the glideslope controller itself, but also the hypothesis that the attitude corrections induced by the attitude controller, as well as the unmodeled orbital dynamics effects, would have only negligible impact on the capability of the glideslope controller to follow a desired velocity profile.

Because of difficulties encountered during the first attempt of the experiment in the second ISS test session, it was attempted a second time during the third test session on August 12, 2007. The results of both attempts are described below.

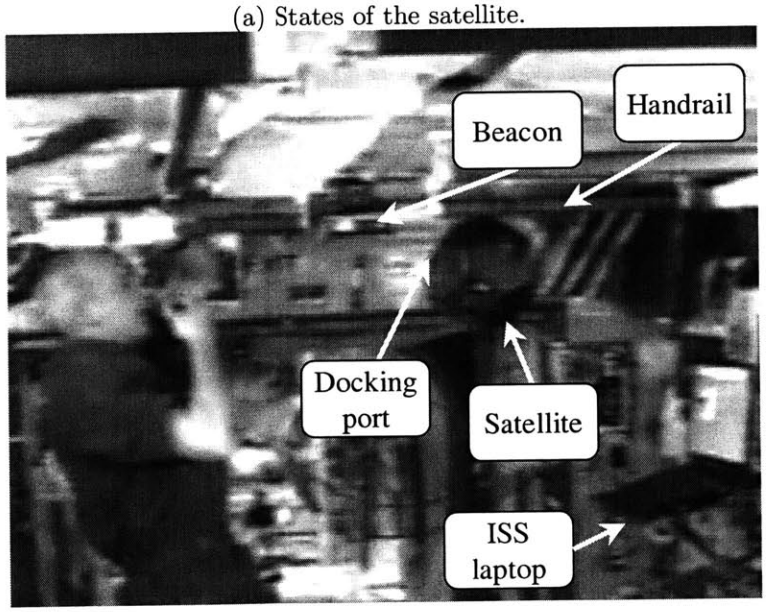
First attempt

For this attempt, the PD-type attitude controller and the relative estimator (providing range and bearing information) were used along with the glideslope controller. The results of the experiment are shown in Fig. 4-19a. Because of a miscommunication with the crew, the satellite was deployed 35 cm from the beacon, instead of the expected two meters. The first 22 seconds were used for the satellite to acquire an attitude solution and to point its docking face at the beacon. During the glideslope approach, starting at time 22 seconds, the satellite performed one accelerating and one breaking pulse bringing itself within 8 cm of the target. Because the glideslope algorithm did not expect such close initial proximity of the satellite to the beacon, the accelerating pulse resulted in too large an approach velocity, which the breaking pulse did not counteract appropriately. The satellite was then commanded to keep pointing at the beacon during the berthing phase, until a certain pointing accuracy was met. It slowly made contact with the handrail to the right of the beacon at time 60s (Fig. 4-19b).

Although the satellite did not make contact with the beacon, valuable information was collected. At time 23 seconds, the glideslope algorithm commanded an acceleration pulse ($\Delta\dot{\rho}$) of 2.07 cm/sec. The resulting acceleration pulse was of 1.85 cm/sec, 11% lower than the desired $\Delta\dot{\rho}$. At time 36 seconds, a breaking pulse ($\Delta\dot{\rho}$) of 0.65 cm/sec was commanded, imparting an actual breaking pulse of 0.63 cm/sec, 3% lower. In both situations, a lower than commanded $\Delta\dot{\rho}$ occurred because of the unmodeled thrust drop when multiple thrusters are simultaneously opened. Except for two temporary jumps at 30 and 32 seconds, the magnitude of the attitude error vector remained below six degrees for the duration of the glideslope approach phase (between 22 and 37 seconds). Furthermore, there is no apparent coupling between attitude and trajectory corrections. These results, although limited, confirm the accuracy of the model used by the glideslope controller. This experiment also showed that the glideslope controller needs to be able to adjust itself to a wider range of initial conditions.



(a) States of the satellite.



(b) Contact with the handrail.

Figure 4-19: Results for the docking experiment with a fixed beacon.

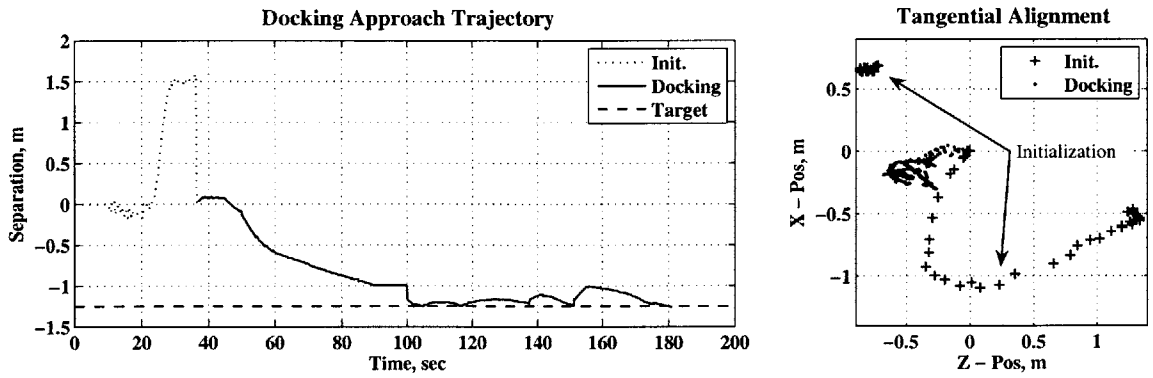


Figure 4-20: Autonomous docking maneuver to a fixed beacon.

Second attempt

Following the limited results obtained in the first attempt, a second attempt of the experiment was performed. To increase the chances of success, a function was implemented in the glideslope algorithm to determine the total number of pulses used in the experiment, as a function of the initial separation. The single-beacon estimator and a PD position controller, validated in the previous test session, were used in the experiment. The single beacon estimator provided the state estimates necessary to navigate in the test volume, which allowed the PD position controller to maintain the tangential alignment with the beacon. The PD-type attitude controller was still used to maintain the attitude of the satellite.

Figure 4-20 shows the results for the docking experiment performed on August 12, 2006. The left plot shows the displacement of the chaser satellite as it moves towards the target, with respect to the initial position of the chaser. The target began approximately 1.2 meter away from the docking face of the chaser. The right plot shows the alignment in the plane perpendicular to the docking axis. The closer the points are to the origin, the better is the alignment.

In the initialization phase, the satellite induced a random tumble to simulate potential high tip-off rates that could occur after deployment from a launch vehicle. The satellite damped out all rotations using its gyroscopes, and initialized its single beacon estimator to acquire a position and an attitude solution. After successful acquisition, it pointed its -y face toward the beacon between time 20 and 36 sec-

onds. The position of the beacon in the satellite's body coordinate frame (at time 36 seconds) became the reference position for the remainder of the experiment. The satellite then initialized its approach toward the beacon. At time 85 seconds, IR noise from an unknown external device started to corrupt the U/S data transmitted to the single-beacon estimator and forced the thrusters to remain closed because of a software constraint.[3] This caused the satellite to temporarily get off track until time 142 seconds. Nevertheless, it regained control after the IR noise disappeared and eventually made contact with the beacon at time 180 seconds.

A close analysis of the results proved that, prior to the appearance of IR noise in the test volume, the actual $\Delta\dot{\rho}$ produced when the glideslope docking module commanded a velocity correction that was within 5 mm/sec of the desired value for 70% of the time (larger errors were caused by the initial pulses asking for large $\Delta\dot{\rho}$ and saturating the thrusters). This error was judged acceptable if it would occur in docking experiment using SPHERES. The attitude controller displayed good performance prior to the appearance of the IR noise. However, the tangential alignment was not well maintained. Video of the experiment showed that this was caused by nominal attitude errors, which the single-beacon estimator interpreted as tangential errors. Although this experiment validated the glideslope docking module, it revealed a problem with the current implementation of the single-beacon estimator, that led to tangential misalignments during the approach.

4.3.5 Global estimator experiments

The global estimator was last component of the GN&C architecture to be validated prior to the start of docking experiments with two satellites. The Shuttle mission STS-121 brought up the remaining beacons necessary for the SPHERES global navigation system in July, 2006. Starting on August 19, 2006, the global estimator became the standard estimator on SPHERES.

[3]To avoid corruption of the U/S-based navigation system by the U/S noise component of the thrusters (Section 3.1.2), the software was programed to turn OFF all thrusters for a period of 200 msec following the reception of an IR flash.

The full validation of the global estimator spanned multiple experiments. The absence of a truth measure, with a higher accuracy than the global estimator, was a challenge that affected the validation process. In the end, it was determined that the best validation experiment would be a docking experiment where two global estimators, running in two separate satellites, provide state estimates that are required to be in agreement with each other (on the order of a few millimeters), for docking to occur. This section presents two of the experiments that led to the validation of the global estimator.

Two free-floating satellites

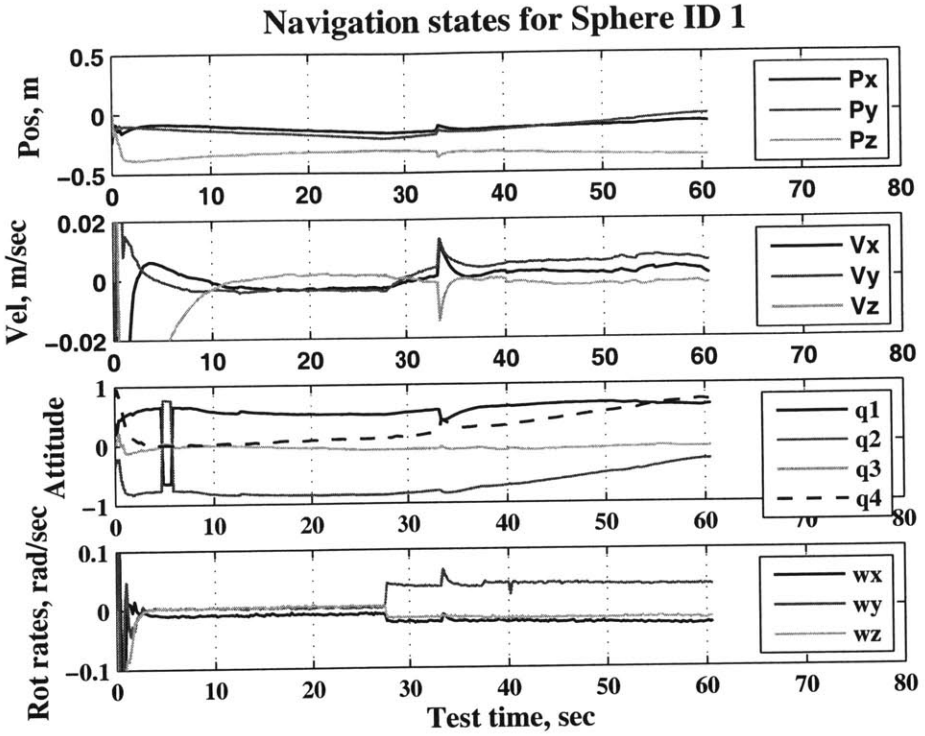
The experiment was designed to demonstrate the capabilities of the global estimator in the ISS with two satellites in close proximity.[4] Five beacons were mounted on the walls around the test volume. The location of each beacon was entered in the ISS laptop by the crew. Two satellites were deployed 0.2 meter apart in the test volume with nominally zero rates. No control was performed for the experiment.

The results (global position, velocity, attitude and angular rate with respect to a coordinate frame attached to the ISS) are shown in Fig. 4-21. For both satellites, the state estimates computed by the global estimator are very smooth. Only two jumps are present: one in Fig. 4-21a at 33 seconds and one in Fig. 4-21b at 23 seconds. They were caused by bad measurements leaking through the pre-filter, possibly the result of multi-path.[5] The state estimates on both satellites converged within 10 seconds. Looking closely at the rate plot of Fig. 4-21a, a sudden angular rate change is observed at 28 seconds, which could be explained by a collision of some sort. In the video of the experiment, the crew is observed gently poking at the satellite to prevent it from hitting an exercise machine attached to the wall of the ISS.

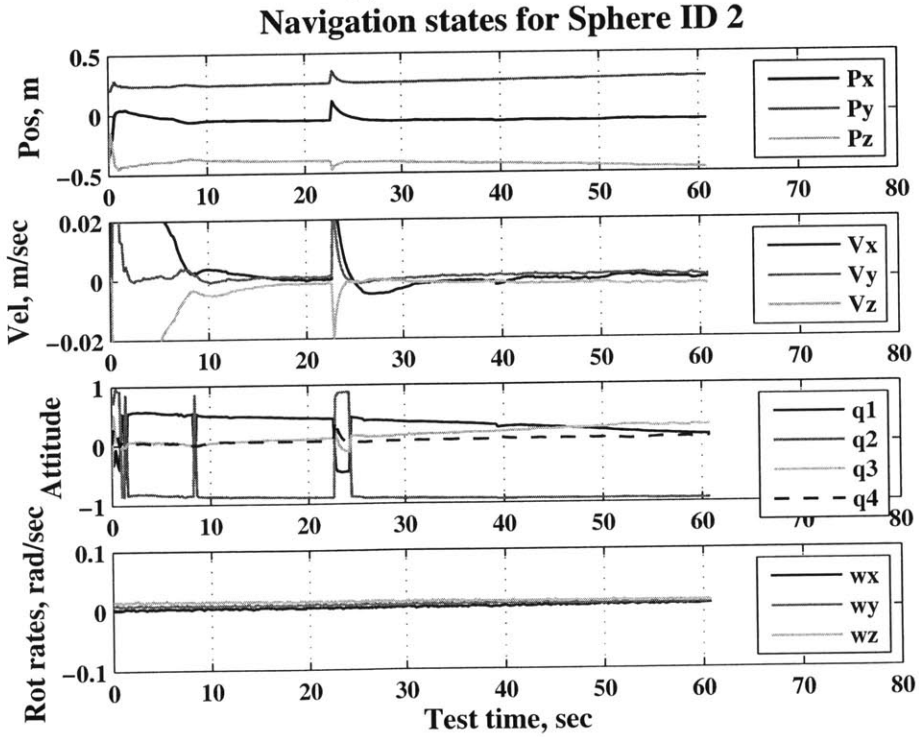
Overall, this experiment, along with a series of similar ones, were successful. They showed good initial convergence and good precision of the estimator throughout the test volume. They also confirmed the need to better filter bad measurements to reduce

[4]In close proximity, the satellites can potentially hide beacons from each other, reflect U/S waves, and induce multi-path effects.

[5]This experiment was performed before FD through filter innovation analysis was implemented.



(a) States of satellite 1.



(b) States of satellite 2.

Figure 4-21: Global navigation states for two satellites separated by ≈0.2 meter.

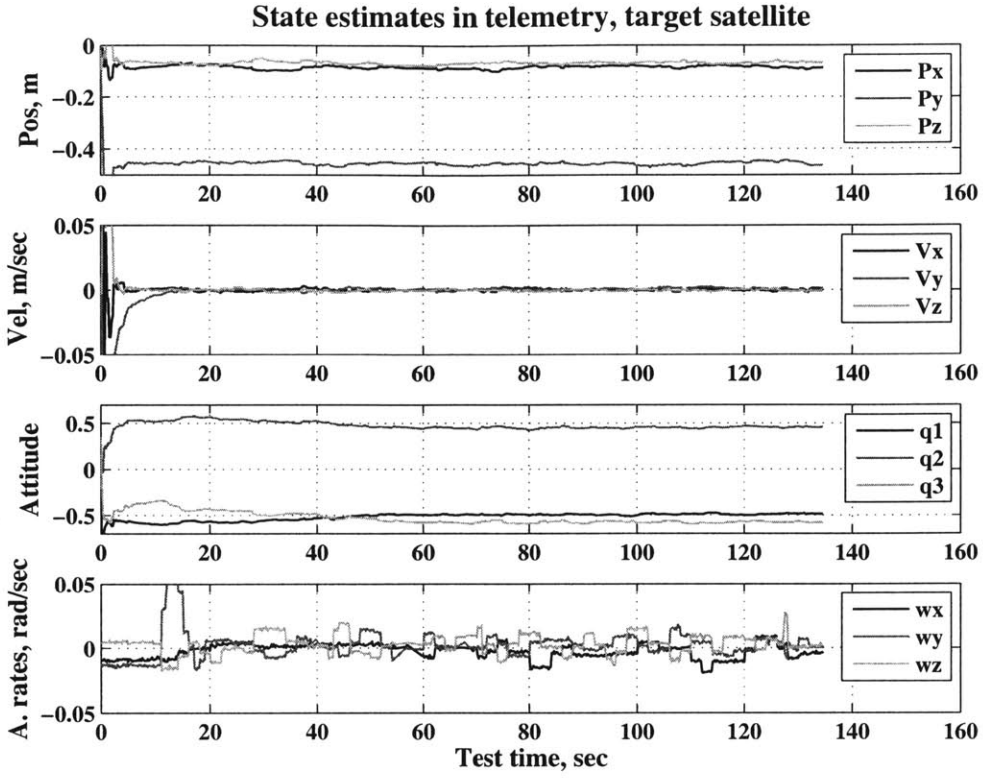


Figure 4-22: State telemetry from an experiment in the ISS using the global estimator.

the risk of registering jumps in the state estimates, which could lead to collisions when the satellites are operating in close proximity.

Docking to a cooperative target

After implementing the fault detection technique described in Sections 3.7.1 and 4.1.1, the global estimator appeared to be much more robust with two satellites flying in close proximity. Its validation was confirmed with the success of a docking experiment involving a global estimator independently running in two different satellites. This experiment is explained in more detail in Chapter 6, but important results regarding the global estimator are presented here.

Figure 4-22 illustrates state estimates computed by the target satellite. The target was free floating for the first 10 seconds of the experiment to allow the global estimator

to converge. For the next 47 seconds, it was commanded to hold its position and keep its docking face oriented toward the chaser. At time 57 seconds, when it detected that the chaser was approaching, it switched modes and started to hold both position and attitude for the remainder of the experiment. Docking occurred at 127 seconds.

Figure 4-5 shows the total filter innovation computed throughout the experiment by the target satellite. Without rejection of the faulty measurements, the state estimates would have been perturbed. This would have, potentially, led to a collision, as in the previous test session. Instead, the state estimates shown in Fig. 4-22 are smooth throughout the docking maneuver, even in the presence of measurement errors, validating the global estimator, for experiments involving close proximity maneuvering. The results presented in this section are representative of all the experiments performed in the ISS using the global estimator with embedded fault detection capability.

The global state estimator now provides sub-centimeter precision for most of the test volume defined by the location of the external beacons in the ISS. The precision around the center of the test volume, where all five beacons are within line-of-sight, is $\approx \pm 1$ mm.

4.4 Summary

During the implementation of the GN&C modules presented in Chapter 3, a series of problems with the robustness and the accuracy of the state estimators were identified, from jumps in the state estimates due to the presence of outliers to occasional divergence caused by the covariance matrix being driven non-positive definite. These problems were found to be the main causes of failures when performing early autonomous docking experiments on the MIT SSL 2-D air table. They were solved by the use of a pre-filter, to discard the outliers prior to sending the measurements to the EKF, by using a reduced form for the covariance matrix as proposed by Lefferts, Markley and Shuster [71], and by using a fault detection technique involving the filter innovation computed when updating the state estimates.

This chapter also discussed the experiments performed to incrementally mature the different GN&C modules, to be integrated in a GN&C architecture for autonomous docking. They consisted of preliminary experiments performed onboard NASA's KC-135 reduced gravity research aircraft, and more recent experiments performed in the ISS. An important outcome of these experiments is the validation, in the SPHERES operational environment (the ISS), of all the GN&C modules used in the experiments presented in the next chapters. These validated GN&C modules represent an important contribution as part of this research.

Chapter 5

Software integration

The integration of the GN&C modules is based on the proposed GN&C architecture in Section 3.3.1. The result of the integration process is the GN&C software used to conduct the autonomous docking experiments. The integration must ensure that the correct information is properly communicated between the modules, as well as the peripheral devices such as sensors and thrusters, and that the processor and memory are not overloaded by the various processes. System integration also includes V&V through ground experiments.

This chapter discusses the integration process to develop the GN&C software. It is organized as shown in Fig. 5-1. The rational behind the selection of the GN&C modules is briefly discussed, while the key hardware considerations are explained in more detail. Three autonomous docking experiments performed in flat floor facilities, as well as two on-orbit formation flight experiments, are used to demonstrate different operational scenarios using the integrated software. They also provide valuable experience on the establishment of an array of GN&C modes, which are used by the MVM module. Finally, this chapter ends with a summary of the GN&C architecture adopted for the on-orbit docking experiments presented in the next chapter.

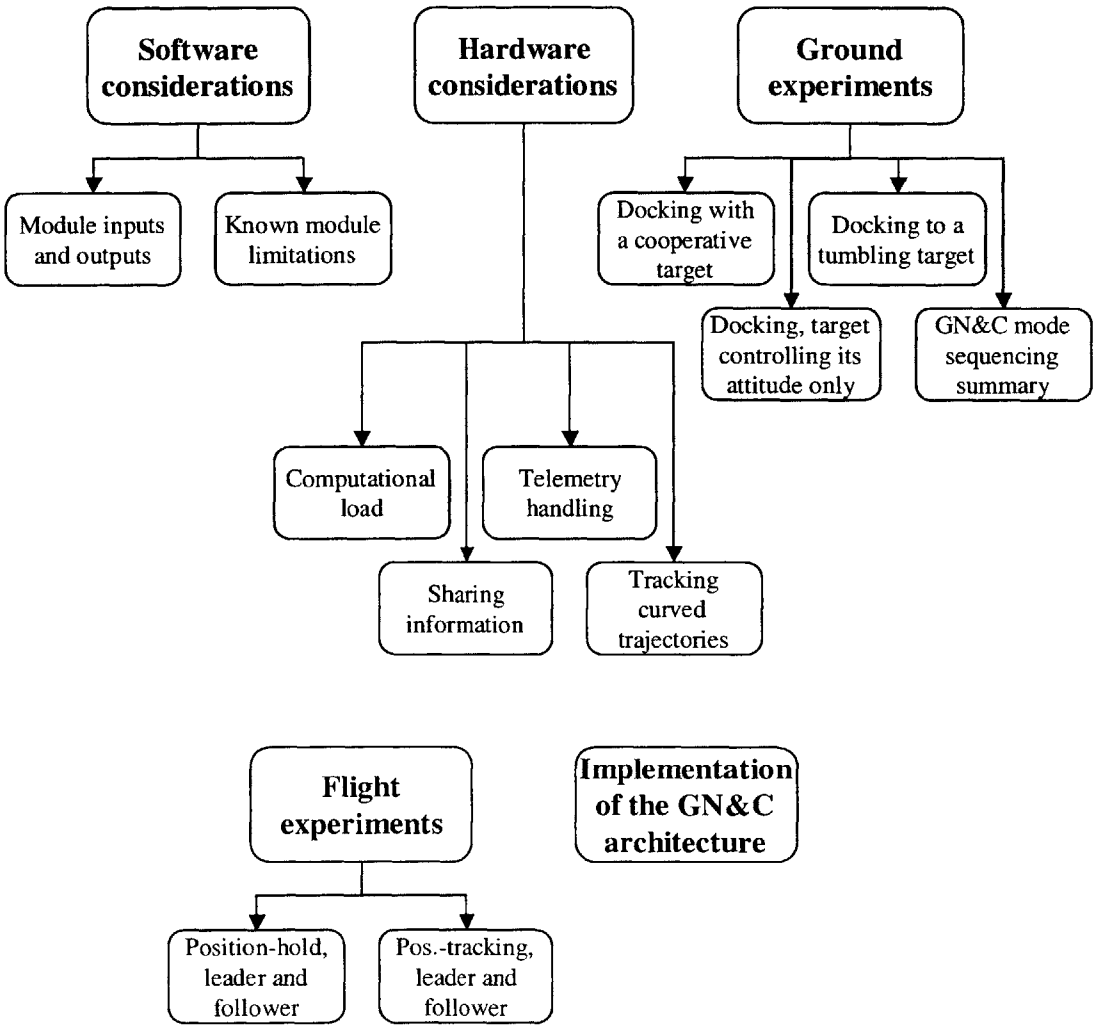


Figure 5-1: Overview of Chapter 5.

5.1 Software considerations

The modularity of the GN&C architecture presented in Section 3.3 offers considerable flexibility in developing GN&C software for a particular experiment. Moreover, GN&C modules with different capabilities can be quickly integrated to allow an investigator to test different combinations. The selection of the right combination of modules is largely dependent on the available hardware and on the experiment to be performed. For example, the global estimator uses all five external beacons. With only two satellites (and their onboard beacons), the three variants of the relative estimator can be used. If one satellite and one external beacon are available, but six DOF trajectory tracking is required, then only the single beacon estimator variant is appropriate.

This section provides software related information to consider when selecting which GN&C modules to integrate. The inputs and outputs of each module are reviewed in detail. Some known limitations are also discussed.

5.1.1 Module inputs and outputs

With the limited number of modules currently implemented, the inputs and outputs associated with each module is an important factor when selecting which module to use for a particular experiment. Depending on the hardware being used, the number of modules with the desired inputs and outputs can be reduced to one or two only. This subsection provides the inputs and outputs associated to the GN&C modules presented in Chapter 3.

Estimation modules

Table 5.1 provides an overview of the inputs and outputs for the estimation modules currently implemented. The beacon address corresponds to the pinging delay after the IR flash synchronizing all the beacons (Table 3.1 and Fig. 3-7). The TOF data is recorded by each of the 24 U/S receivers following a beacon ping, while the gyroscope data is typically the output of the anti-aliasing filter removing the 338 Hz resonance

on the SPHERES gyroscopes (Appendix B). Among all the state estimation modules, only the global estimator and the single beacon estimator directly integrate gyroscope data to compute the state estimates. Finally, the displacement and velocity estimates output by the single beacon estimator are expressed in a coordinate frame attached to the beacon, but rotating according to the gyroscope drift (Section 3.4.2).

Control modules

The inputs and outputs for the control modules are shown in Table 5.2. For the PID-type controllers, the integration period, the time interval between each call of the corresponding controller, has to be entered explicitly. Also, the controllers only update the elements of the control input vector that they are assigned through either the non-zero gains or the control mode. For the pulse-width modulator and the phase plane controller, the force inputs, the position errors and the velocity errors need to be rotated in the body frame, if the flag indicates that they are expressed in the inertial frame. The rotation is performed using the quaternion elements of the state vector, which is also input to the controller. Finally, the phase plane controller can provide either a vector of forces and torques, or vectors of thruster ON and OFF times.

Path planning modules

Table 5.3 contains the inputs and outputs for the two path planning modules. Although the glideslope controller has the option to control all three translation DOF, it is typically used on SPHERES to control only the DOF aligned with the docking axis. When the satellite is used with a heavy air carriage, there is a chance that a thruster pulse corresponding to a desired $\Delta\dot{\rho}$ cannot be achieved within a single glideslope period (typically the control period). Therefore, the controller provides the option to execute multiple shorter pulses during consecutive periods to achieve the desired $\Delta\dot{\rho}$. The maximum number of consecutive periods allowed per thruster pulse has to be provided by the user. The glideslope controller also outputs two flags indicating if a pulse is being executed in the incoming period and if the controller reached the end of the desired trajectory (when all the thruster pulses are executed).

Table 5.1: Inputs and outputs for the estimation modules.

Estimation modules	Inputs	Outputs
Global estimator	Full global update • Beacon address • Timestamp* • TOF data (24 U/S measurements) from all 5 beacon pings, processed 1 ping at a time) Attitude update • Timestamp* • Gyroscope data (vector of 3 measurements)	• Timestamp* • Satellite state estimates in a 13 elements vector, including: o Position (3), inertial frame** o Velocity (3), inertial frame** o Attitude (4 quaternions) o Angular rate (3)
Relative estimator	• Beacon address • Timestamp* • TOF data (24 U/S measurements) from a given beacon ping)	• Timestamp* • Beacon state estimates in a 6 elements vector, including: o Position (3) o Velocity (3)
Relative estimator (range & bearing variant)	• Beacon state estimates in a 6 elements vector, including: o Position (3) o Velocity (3)	• Beacon state estimates in an 8 elements vector, including: o Range (1) o Range rate (1) o Bearing angles (pitch and yaw, in 4 quaternions)*** o Bearing rates (2)
Relative estimator (variant combining the states from two satellites)	• State estimates of the other satellite's beacon in an 8 elements vector, including: o Range (1) o Range rate (1) o Bearing angles (pitch and yaw, in 4 quaternions) o Bearing rates (2) • State estimates of the onboard beacon computed by the other satellite in a 6 elements vector, including: o Position (3) o Velocity (3)	• Relative state estimates in a 12 elements vector, including: o Range (1) o Range rate (1) o Tangential alignment (2) o Alignment rate (2) o Bearing angles (pitch and yaw, in 4 quaternions)*** o Bearing rates (2)
Relative estimator (single beacon estimator variant)	Position update • Beacon state estimates in a 6 elements vector, including: o Position (3) o Velocity (3) Attitude update • Timestamp* • Gyroscope data (array of 3 measurements)	• Timestamp* • State estimates in a 13 elements vector, including: o Displacement (3) o Velocity (3) o Attitude change (4 quaternions) o Angular rate (3)

* The timestamp is the time the corresponding measurement was taken.

** When *inertial frame* is specified, the corresponding states are expressed in a coordinate frame attached to the testing environment where the beacons are mounted. Otherwise, they are expressed in the body coordinate frame.

*** When the bearing angles are expressed in quaternions, q_1 remains 0.

Table 5.2: Inputs and outputs for the control modules.

Control modules	Inputs	Outputs
PID position controller	• Position gains (proportional, integral, derivative) • Integration period (interval between function calls) • State errors in a 13 elements vector	• Control inputs in a 6 elements vector, including: o Updated forces (3) o Torques (3), unchanged
PID-type attitude controller	• Attitude gains (proportional, integral, derivative) • Integration period (interval between function calls) • State errors in a 13 elements vector	• Control inputs in a 6 elements vector, including: o Forces (3), unchanged o Updated torques (3)
Pulse-width modulator	• Control inputs in a 6 elements vector, including: o Forces (3) o Torques (3) • State estimates in a 13 elements vector • Minimum thruster opening time • Duty cycle • Flag indicating the coordinate frame associated with the forces in the control input vector	• Control inputs in 2 vectors of 12 elements, including: o Thruster ON times (12) o Thruster OFF times (12)
Phase plane controller	• Control mode indicating which DOF to control • State errors in a 13 elements vector • Flag indicating the coordinate frame associated with the position and velocity errors in the state error vector • State estimates in a 13 elements vector • Type of output (force/torque vector or thruster ON/OFF times)	• Control inputs in a 6 elements vector, including: o Updated forces (3) o Updated torques (3) OR • Control inputs in 2 vectors of 12 elements, including: o Thruster ON times (12) o Thruster OFF times (12)

Table 5.3: Inputs and outputs for the path planning modules.

Path planning modules	Inputs	Outputs
Glideslope controller	• Time • State errors in a 13 elements vector • Duty cycle • Desired safe terminal velocity • Number of glideslope periods allowed per thruster pulse • Flag indicating which DOF to control	• Control inputs in a 6 elements vector, including: o Updated forces (3) o Torques (3), unchanged • Flag indicating an ongoing thruster pulse • Flag indicating the end of the glideslope sequence (after the last pulse has been fired)
Pre-computed trajectories (through MILP or MPC)	• Time	• Target state vector in a 13 elements vector • Nominal control inputs in a 6 elements vector (optional)

The second module provides the desired trajectory and an optional nominal control input vector. Both can be calculated offline by an advanced path planner.

FDIR modules

Table 5.4 shows the inputs and outputs of the only FDIR module currently implemented. Since the module is embedded in the EKF, the U/S measurements that are input are pre-filtered measurements (Section 4.1.1). Therefore, the maximum number of measurements input to the module is four. The expected measurements are an estimate of the measurements computed using the latest state estimates. The module outputs a vector of five flags indicating the status associated with the measurements following each beacon ping. When 15 bad measurements are collected with less than 2 seconds between each of them, the flag associated to the beacon corresponding to these measurements is set to *TRUE*. When more than two of the five flags are set to *TRUE*, the global estimator is declared faulty since, for each cycle, the measurements following the ping of at least three different beacons are necessary to fully update the states.

5.1.2 Known module limitations

An important constraint imposed by the hardware on all state estimators is a maximum sampling frequency of 5 Hz, which limits the bandwidth of the system to 2.5 Hz

Table 5.4: Inputs and outputs for the FDIR modules.

FDIR modules	Inputs	Outputs
FD through filter innovation analysis	• Time • Beacon address • Vector of U/S measurements • Vector of predicted U/S measurements • Innovation threshold	• Current fault status in a 5 elements vector, 1 element per beacon • Global estimator status (nominal or failed)

(from Nyquist's theorem). This limitation applies only to the position states, not the attitude states, unless the attitude is solely determined by TOF measurements. When the attitude determination process uses gyroscope data (e.g., the global estimator and the single beacon estimator), the bandwidth is increased to 50 Hz (cutoff frequency of the gyroscopes).

When using the PID-type controller, coupled with the pulse-width modulator, any commanded force or torque vector resulting in a thruster opening time longer than the allowed thrusting window is scaled down to maintain the direction of the commanded control input, at the expense of the amplitude. In the current implementation, the fraction of the impulse that is not applied is lost. In practice, this limitation is recognized during controller design and, as a result, no impulse is lost.

Experimentation has shown that the phase plane controller provides better performance than a PID-type controller, when the state estimates exhibit occasional jumps. This results because it produces thruster pulses with a pre-determined duration and therefore cannot saturate the thrusters. However, when stable state estimates are available, a PID-type controller provides more accuracy because of the fine resolution on the thruster pulses (down to the minimum pulse width, usually set at 10 msec).

When the state errors are large (>20 cm in position and >45 degrees in attitude), PID-type controllers, using gains computed as described in Section 3.5.1, tend to overshoot. Either a PD-type or a phase plane controller might be more appropriate.

The glideslope controller, that is currently implemented, does not perform well with an initial error <15 cm along the axes it controls. Therefore, it is preferable to use a PID controller for tangential corrections, when, after a good initial alignment, the initial error is <15 cm. The glideslope algorithm performs well along the docking

axis, when the initial separation is >15 cm.

The FD module through filter innovation analysis is currently tuned to detect a hard failure of the U/S navigation sensors no earlier than six seconds after they occur. This limit is reasonable given that the state estimates computed from gyroscope measurements, and the propagation of the position estimates, are likely to remain accurate for that short period of time, because of the low perturbation level in the ISS.

This section summarized important software considerations to be made when integrating the GN&C software. The inputs and outputs to each module were covered in detail, as well as known limitations associated with some of the GN&C modules developed in Chapter 3. The next section discusses important hardware considerations.

5.2 Hardware considerations

The following issues have to be considered when integrating software for real-time operation on SPHERES:

- the computational load;
- the transfer of information between the modules;
- the transfer of information through the telemetry.

Each of these issues, applied to the integration of the software on SPHERES, is covered in this section. A subsection also discusses the limits imposed by the propulsion system when tracking a curved trajectory (e.g., when docking to a tumbling target while maintaining alignment with its docking face).

5.2.1 Computational load

Among all the modules currently implemented, the global estimator is the most computationally intensive and the most prone to saturate the computer. The main reason

is the short time delay between U/S beacon pings (20 msec) and the processing of the gyroscope data (typically 50 msec). Under nominal operation, the state estimates are propagated and updated as soon as measurements (either from the U/S system or the gyroscopes) are passed to the computer. However, since that computation can take a fair amount of time (on the order of a couple of milliseconds), it is performed in a low priority thread (more specifically, in the primary interface function called *gspTask*). This thread is allowed to be interrupted by higher priority processes to ensure their timely execution.

To make sure that the global estimator runs smoothly on the hardware, the total computation time must not exceed the shortest U/S system sampling interval, which is 20 msec. To estimate the computational load, an experiment was performed where the internal clock was read before and after different processes during the execution of the EKF. The difference between the two readings is an estimate of the amount of time the computer spent for performing each process (it is possible that the computation was briefly interrupted by a higher priority process). The following results were collected for a test with a stationary satellite.

- Global update ($\approx$17 msec, distributed as follows)
  - Propagation ($\approx$3.5 msec)
  - Pre-filter ($\approx$3.5 msec)
  - Measurement Update ($\approx$10.3 msec, distributed as follows)
    - $\mathbf{h}$ and $\mathbf{H}$ ($\approx$3 msec)
    - $\mathbf{HPH}' + \mathbf{R}$ ($\approx$2 msec)
    - $(\mathbf{HPH}' + \mathbf{R})^{-1}$ ($\approx$1 msec)
    - $\mathbf{K}$ ($\approx$2 msec)
    - $\mathbf{x}^+$ ($\approx$0.3 msec)
    - $\mathbf{P}^+$ ($\approx$2 msec)
- Gyroscope update ($\approx$3.5 msec, distributed as follows)

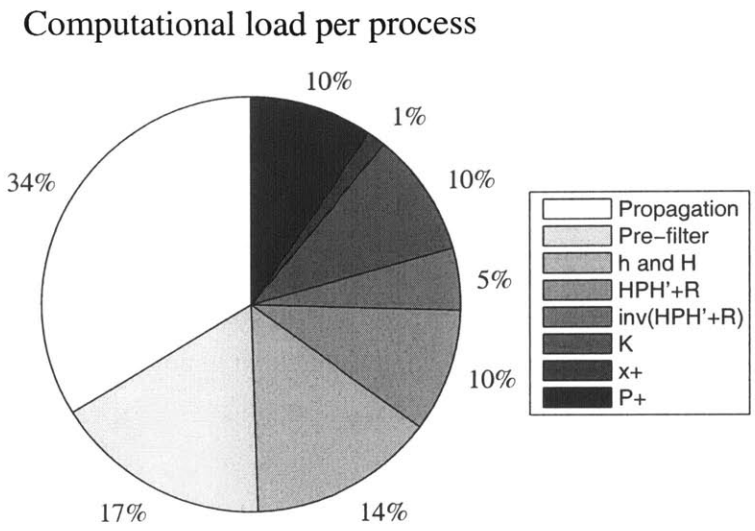


Figure 5-2: Percentage of the total computation time (416 msec) taken by each process.

- FIR low-pass filter (≈0 msec)
- Propagation (≈3.5 msec)
- Measurement update (≈0 msec)

Since the global updates and the gyroscope updates are executed within their limits (20 msec and 50 msec, respectively), the computer does not become saturated. The execution of each process completes successfully prior to the start of the next one. The overall computation load is estimated as follows. Typically, the global estimator processes TOF data from the five beacons pinging at 4 Hz and processes gyroscope data at 20 Hz. This translates into a total computation time of 416 msec for every one second control period. The total computational load caused by the standard global estimator can then be estimated at 42%. Figure 5-2 shows what percentage of the total computation time is taken by each process. The propagation process is clearly the one taking the most computation time, mainly because it is called when either a set of U/S or gyroscope measurements is processed. On the other hand, the matrix inversion in the U/S measurement update process takes only 5% of the total computation time.

5.2.2 Sharing information between modules

Typically, the data to be transfered between modules is stored in a memory space that is accessible by both modules. Each module can read and overwrite it when needed. Since it is very unlikely that two modules, running in two different software threads, are called one immediately after the other, any information that varies in time, like the state estimates, must be time tagged. Whenever a module requires that information, it can be actualized if necessary.[1]

5.2.3 Telemetry handling

The proper handling of the telemetry has to ensure that the communication system is not saturated. Therefore, the selection of the right variables to transmit, via telemetry, is critical. As a rule of thumb, enough data should be transmitted to allow the reconstitution of the control input vector using the telemetry and a ground simulation of the controller. Data at critical steps in the computation of the control input vector can also be instrumental in identifying problems in the software.

When the data to be transmitted exceeds the capacity of the communication system, there exist data compression schemes that can significantly increase the amount of data transfered, without increasing the communication bandwidth requirement. A common technique consists of sending a reference point, and then the differences between the following data points, instead of the data points themselves, since the difference is likely to be expressed with a lower number of bits. To avoid the loss of too much data when a packet may be lost, reference data points must be sent at regular intervals.

5.2.4 Tracking curved trajectories

The propulsion system imposes limits on the ability of the SPHERES satellites to track curved trajectories. For example, when docking to a tumbling target, a limit

[1] Actualization is the process of propagating the information from the time it was created to the time it is used.

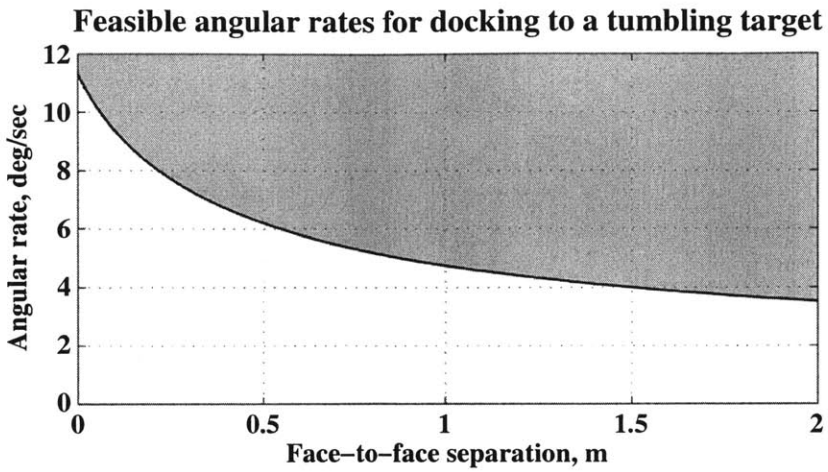


Figure 5-3: Feasible angular rates for circular trajectories at constant velocity, given the thrust produced by the thrusters.

exists on the maximum feasible angular rate, for a circular trajectory around the target at constant velocity, that the chaser can track by continuously thrusting to maintain line-of-sight with the target's docking face. The maximum feasible angular rate can be computed as a function of the face-to-face separation with the tumbling target.[2] For the SPHERES satellites, the result is shown in Figure 5-3 for a separation between 0 and 2 meters. The white zone represents the feasible zone, while the grey zone is the infeasible zone. The limit (separation between the white and the grey zones) was calculated using a satellite mass of 4.3 kg, a thruster force of 0.09 N (representative of the force obtained with the regulator set at 25 psi), and a duty cycle of 20%. Although it is theoretically possible to maintain a circular trajectory at the maximum feasible angular rate, experience has shown that it is preferable to remain well within the limits to imposed by the propulsion system.

5.3 Ground experiments

The GN&C architecture to be described in Section 5.5 is the result of years of experimentation. These experiments were initiated soon after key GN&C modules were implemented for real-time operation on the hardware. The objectives were:

[2]The tumbling motion of the target is such that the docking axis sweeps a plane.

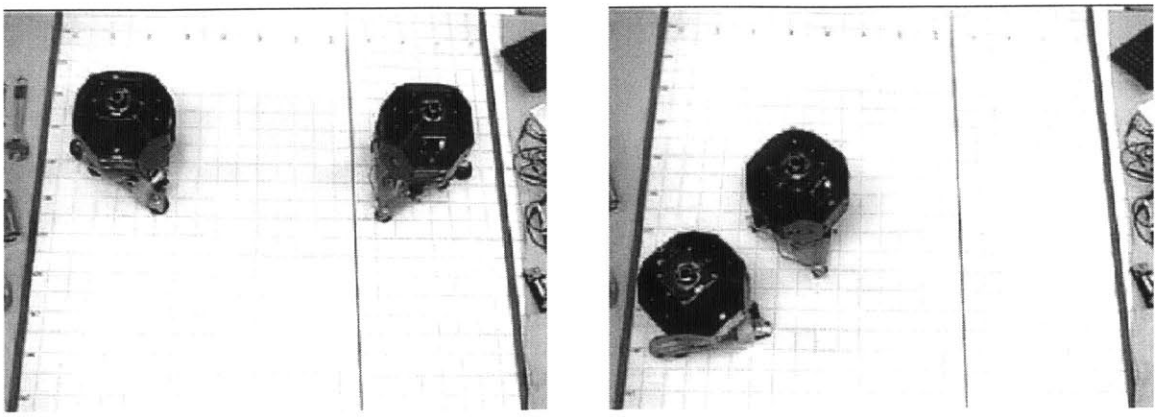


Figure 5-4: SPHERES docking experiments on the MIT SSL 2-D air table.

- to identify a proper sequence of maneuvers and GN&C modes leading to docking with cooperative and tumbling targets;
- to establish requirements regarding the transition between the maneuvers;
- to confirm the capability of each software module to fulfill its corresponding requirements, when integrated into a GN&C architecture.

This section covers three key ground experiments that contributed to fulfilling these objectives. Two autonomous docking maneuvers were performed on the MIT SSL 2-D air table in March 2004 to demonstrate two different docking scenarios (Fig. 5-4). The first involved a fully cooperative target while the second involved a target controlling only its attitude. A third experiment performed at the MSFC flat floor demonstrated the capability to dock to a tumbling target (when the tumble is constrained to be about the vertical axis). This section describes each of these experiments and discusses the combination of GN&C modes that provided the best results.

5.3.1 Docking with a fully cooperative target

For this experiment, the following GN&C modules are integrated:

- The global estimator provides full state estimates early in the maneuver, while the relative estimator (combining the states of both satellites) provides relevant

state information toward the end of the maneuver. As a result, there is a mode switch in the modules, from global to relative estimation.

- The glideslope controller computes a trajectory along the docking axis using the ranging information provided by the relative estimator. It also plans a series of thruster pulses to achieve that trajectory.

- A phase plane controller is implemented to have the two satellites initially pointed at each other and to keep the target pointed at the chaser during the approach.

- A PD-type controller is implemented to keep the chaser pointed at the target during the approach.

The GN&C modes established using these modules are shown in Table 5.5, while their sequencing as coded in the MVM module is illustrated in Fig. 5-5. For the duration of the experiment, the target has full control authority and is able to communicate its state to the chaser. Both satellites are originally pointing in random directions. Since they communicate their states, they can point their docking faces at each other by comparing the states of the other satellite with their own, computing a bearing error and driving that error to zero. Once they are pointed at each other, the chaser switches to a relative estimator to get direct range information. It then initiates its approach and docks with the target. At this point in the development of the architecture, the transition between the modes is mostly based on timers.[3] Eventually, transitions between modes will occur according to the states of the satellite systems.

Figures 5-6 and 5-7 show the results from this autonomous docking demonstration. In Fig. 5-6, the plots represent the separation between the geometric centers of both satellites[4] projected onto the docking axis (plot on the left) and the alignment of the

[3]A synchronization between the clocks onboard each satellite occurs at the beginning of every test.

[4]The separation is the distance between the geometric centers of the satellites, parallel the docking axis (x-body axis of the target). A separation of 21.5 cm (the diameter of a satellite) corresponds to contact.

Table 5.5: GN&C modes for the autonomous docking with a fully cooperative target.

GN&C mode	GN&C modules	Exit condition
EKF initialization	Global estimator	Timer
Pointing	Global estimator Phase plane attitude controller Phase plane position controller	Timer
Position and attitude hold	Global estimator Phase plane attitude controller Phase plane position controller	Timer
Approach	Relative estimator Glideslope controller PD-attitude controller	Range within 10 cm
Capture	Open-loop thruster firing	Timer

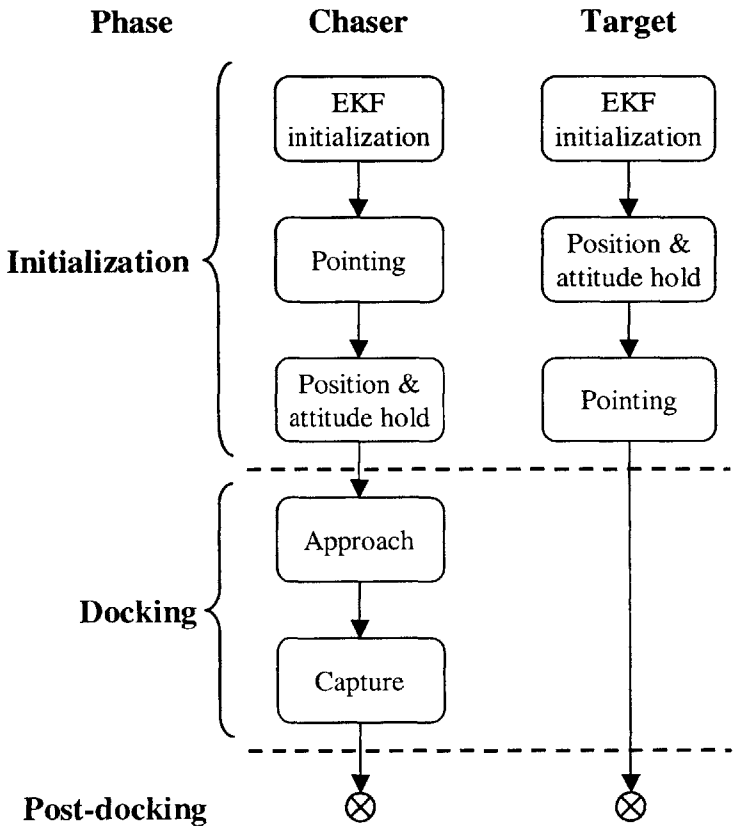


Figure 5-5: Mode sequencing for the autonomous docking with a fully cooperative target.

chaser in the plane perpendicular to the docking axis (plot on the right). The initial separation close to zero is explained by the target not pointing at the chaser. Therefore, the projection of the separation vector on the docking axis is small. Although the satellite is constrained in 2-D, the global estimator, which is configured for 3-D experiments, keeps an estimate of the z-position for each satellite, which explains a non-zero alignment along the z-axis. Figure 5-7 shows the tracking of the desired velocity output by the glideslope algorithm during the approach. Also, as indicated on the plots, the maneuver was divided into three distinct phases (initialization, docking and post-docking). Transition from the first phase to the second was simply driven by the timer in both satellites, while transition to the third phase was driven by the separation, as explained below. The process was as follows:

1. The EKF on each satellite is initialized. The satellites are free floating for the first 10 seconds and then start holding their position and attitude. The initial separation is limited by the size of the air table. At time 16 seconds, the chaser orients its docking face toward the target, while the target keeps holding its attitude. At time 37 seconds, the target orients its docking face toward the chaser. With the target spinning, the apparent position of the chaser in the target body frame changes, which translates to the relative separation increase shown in Fig. 5-6 between 40 and 60 seconds. Both satellites are pointing at each other by the end of this phase, allowing the relative estimator to be initialized.

2. The chaser initiates its approach toward the target and follows the trajectory planned by the glideslope controller. The initial desired velocity is set at 2 cm/sec, while the final desired velocity is set at 0.5 cm/sec. Looking closely at the velocity curve along the docking axis (Fig. 5-7), it is observed that the initial pulse (at 65 seconds) did not induced the desired ΔV because of a perturbation on the air table. The controller fired the thrusters for a little longer on the second pulse (at 73 seconds) to get back on the desired trajectory. It then started to reduce its velocity at the third pulse (at 81 seconds). Due to hardware limitations, the relative estimator does not give accurate relative

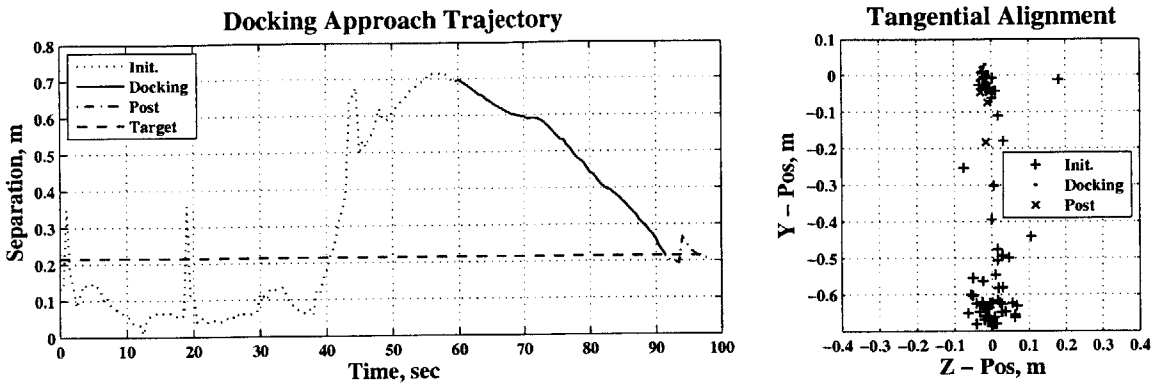


Figure 5-6: Autonomous docking with a fully cooperative target.

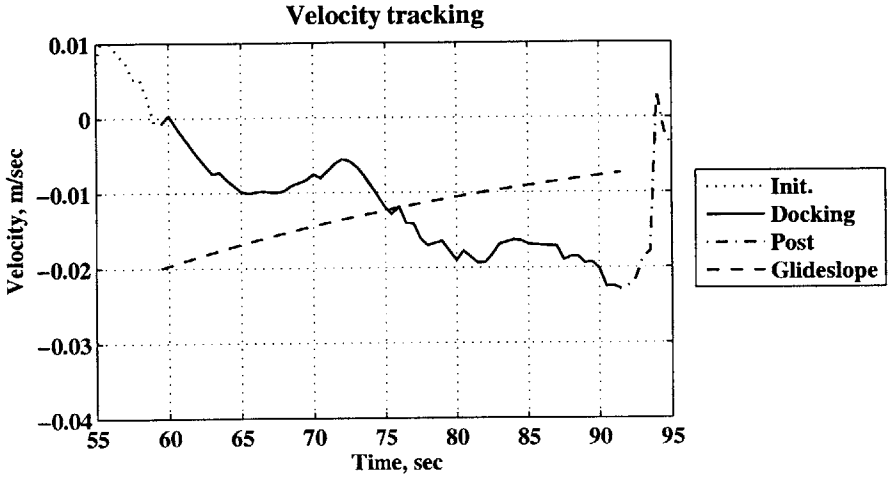


Figure 5-7: Velocity tracking for the autonomous docking with a fully cooperative target.

measurements when the two satellites are within 30 cm separation. For this reason, and because the capture mechanism (Velcro) requires a non-zero velocity to latch properly, a pulse is fired when the chaser is within 32 cm separation of the target. This explains the increase in velocity toward the end of the maneuver. The target was successfully captured at 91 seconds.

3. The last phase represents the motion with both satellites connected. No control is applied by the chaser, but the target keeps maintaining its position, resulting in trajectory corrections after docking is achieved.

This experiment was a success. It provided the first evidence of a successful integration of the different GN&C modules. However, the repeatability of this experiment

was questionable, as the chaser performed successful capture only ≈20% of the time. Other than perturbations on the air table, the main reasons were jumps in the state estimates (e.g., in Fig. 5-6 at 19 and 45 seconds) and the glideslope controller having difficulties in tracking the desired velocity profile.

5.3.2 Docking with the target controlling only its attitude

In the second docking scenario, the target has only attitude control authority, but is able to transmit its states to the chaser. Any drift velocity is considered as a perturbation, which is not modeled in the planning of the trajectory by the glideslope controller.

Even after leveling the air table, the residual acceleration and the limited testing area did not allow the execution of this experiment with a floating target. This scenario was simulated with the target *partially* floating on the air table (i.e., with just enough friction to cancel the drift caused by the tilt of the table, but not the drift caused by plume impingement).

The following GN&C modules are integrated:

- The glideslope algorithm is used to compute the approach trajectory.
- A PD-type controller is implemented to maintain the chaser facing the target. It also compensates for the tangential position error in between thruster pulses planned by the glideslope algorithm.
- The relative estimator (combining the states of both satellites) is used to provide relevant state information to the chaser.

The GN&C modes are shown in Table 5.6, while the adopted mode sequence is illustrated in Fig. 5-8. Initially, the target is oriented toward the chaser. The chaser detects the target's beacon, points toward it and aligns itself in front of it. When properly positioned, it initiates its approach and docks with the target.

Figures 5-9 and 5-10 show the relative state estimates collected during this docking demonstration. In Fig. 5-9, the left plot represents the face-to-face separation seen

Table 5.6: GN&C modes for the autonomous docking with the target controlling only its attitude.

GN&C mode	GN&C modules	Exit condition
EKF initialization	Relative estimator	Timer
Pointing	Relative estimator PD-type attitude controller Pulse-width modulator	----------
Range hold	Relative estimator PD-type attitude controller PD position controller (range only) Pulse-width modulator	Timer
Arc traverse	Relative estimator PD-type attitude controller PD position controller Pulse-width modulator	Position within a 5 cm x 5 cm box
Approach	Relative estimator PD-type attitude controller PD position controller (tangential only) Glideslope controller Pulse-width modulator	Range within 12 cm
Capture	Open-loop thruster firing	Timer

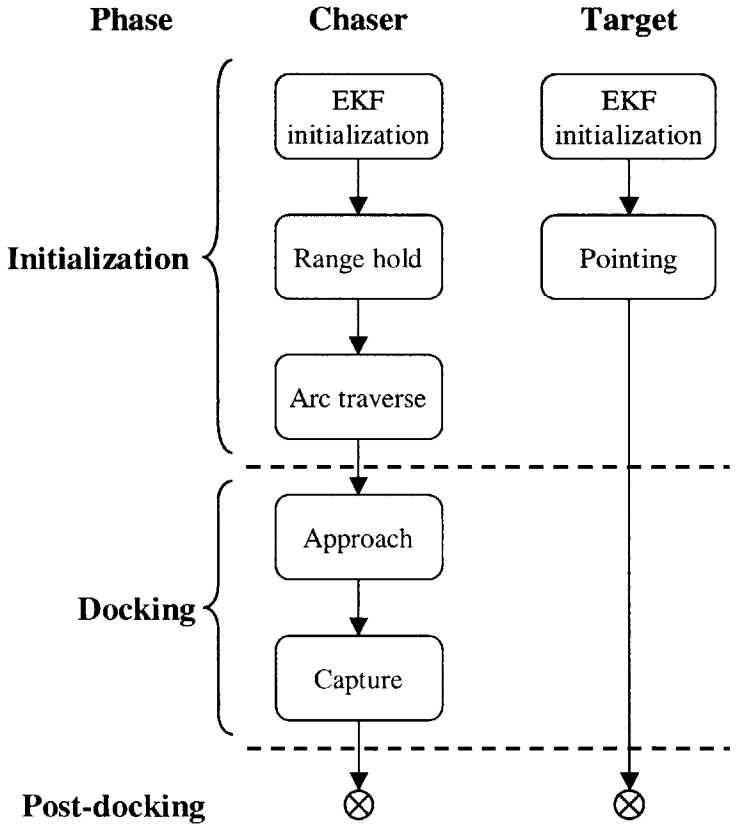


Figure 5-8: Mode sequencing for the autonomous docking with the target controlling only its attitude.

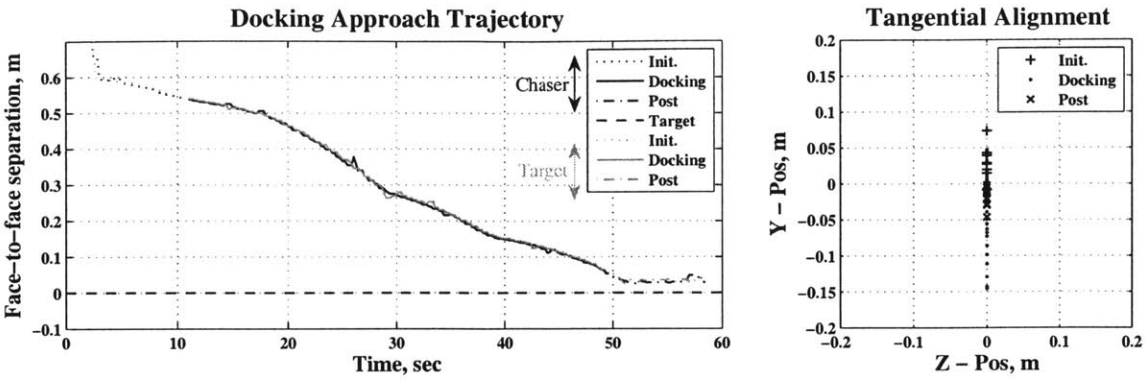


Figure 5-9: Autonomous docking with the target controlling only its attitude.

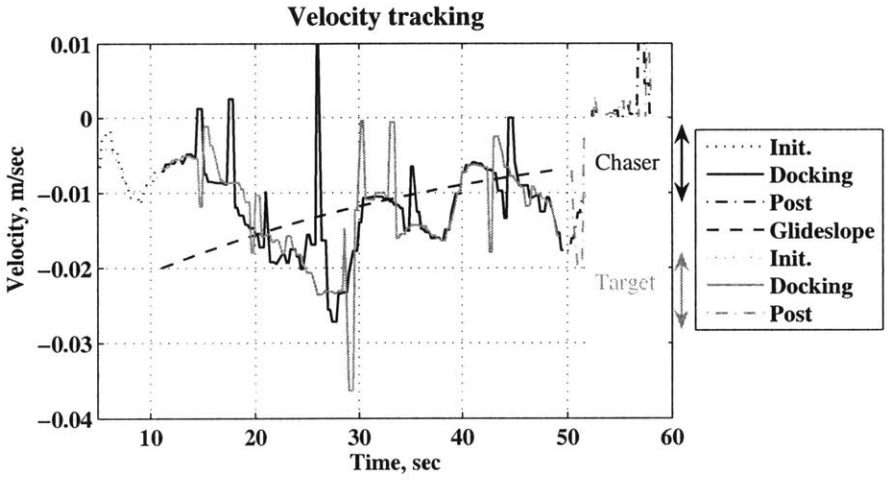


Figure 5-10: Velocity tracking for the autonomous docking with the target controlling its attitude only.

by both satellites.[5] The right plot shows the alignment of the chaser. The alignment along the z-axis is set to zero because the relative estimator is only estimating the alignment along the y-axis (horizontal axis) for this 2-D experiment.

The chaser is free floating for the first five seconds, and then moves into position in front of the target. It performs its approach during the docking phase. The face-to-face separations sensed independently by both satellites are essentially identical, confirming proper function of both EKFs. The range decreases toward zero as expected, although it levels off at ≈4 cm face-to-face separation, which is indicative of

[5]The face-to-face separation is the distance between the centers of the docking faces of both satellites. A zero face-to-face separation corresponds to contact.

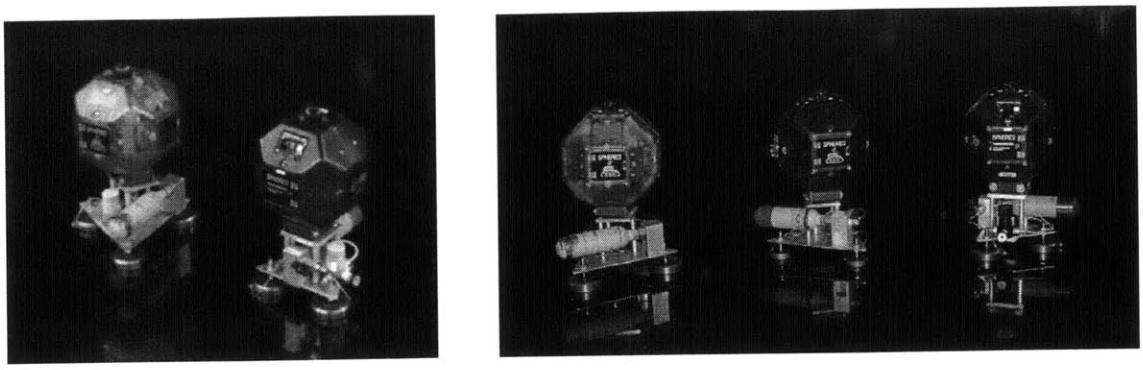


Figure 5-11: SPHERES experiments at NASA MSFC.

a bias at very close proximity. The only indication that the target is held in place by friction comes from its constant body rates (not shown), except for two slight bumps showing perturbations caused by the plume of the chaser. Successful capture occurred at 50 seconds (detected by a jump in the gyroscope data at that time).

This time, the tracking of the velocity output by the glideslope controller is better, as seen by the velocity curves computed from both satellites in Fig. 5-10. Overall, the velocity appears to be decreasing with the distance-to-go until time 48 seconds, when the final push to capture was initiated. Although the docking maneuver was a success, it showed that improvements in the performance of the tangential error and velocity control are required. Also, the repeatability of this experiment was approximately the same as the previous one ($\approx$20%).

5.3.3 Docking to a tumbling target (rotating in a 2-D plane)

Because of the limited testing area at MIT, the autonomous docking with a target tumbling (rotating in a 2-D plane) presented in this section was performed on the flat floor at NASA MSFC in June and December 2004 (Fig. 5-11). The objective of this experiment was to demonstrate the capability of the GN&C architecture, used in Section 5.3.2, to perform docking with a target that is slowly tumbling. The GN&C modes are shown in Table 5.7, while the sequence of modes adopted is illustrated in Fig. 5-12.

The maneuver used the relative estimator, combining the states from both satel-

Table 5.7: GN&C modes for the autonomous docking to a tumbling target (rotating in a 2-D plane).

GN&C mode	GN&C modules	Exit condition
EKF initialization	Relative estimator	Timer
Tumble	Relative estimator PD-type attitude controller Pulse-width modulator	----------
Approach	Relative estimator PD-type attitude controller PD position controller (tangential only) Glideslope controller Pulse-width modulator	Timer
Berth	Relative estimator PD-type attitude controller PD position controller Pulse-width modulator	Range within 3 cm Tangential error within 2 cm Pointing within 10 deg Velocity within 0.5 cm/sec Rate within 0.005 rad/sec
Capture	Open-loop thruster firing	Timer

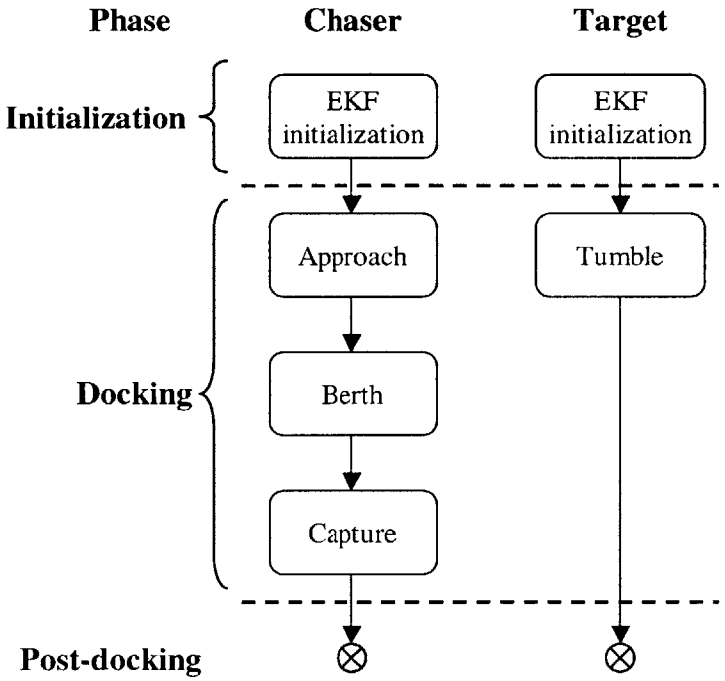


Figure 5-12: Mode sequencing for the autonomous docking to a tumbling target (rotating in a 2-D plane).

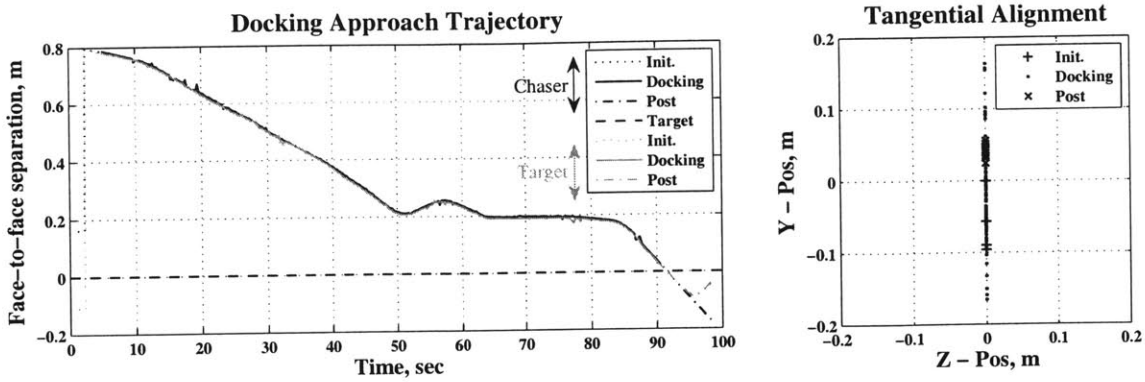


Figure 5-13: Autonomous docking to a tumbling target (rotating in a 2-D plane).

lites. The rotation rate was set to 2.4 deg/sec. The target was allowed to use its thrusters, so as to maintain its angular rate to counteract the friction on the floor. Both satellites were deployed with an initial face-to-face separation of approximately 80 cm. Throughout its maneuver, the chaser was regulating its tangential error to maintain its alignment at a specified radial distance on the axis normal to the target's docking face (i.e., docking axis). To increase the chances of success, the chaser was commanded to stop its radial approach at a *berthing* face-to-face separation $\approx$17 cm, where it aligned itself precisely. When the alignment reached an acceptable tolerance (Table 5.7), the chaser fired its thrusters to accelerate in the direction of the target, so as to gain some velocity, and captured the target using its Velcro docking mechanism.

Figures 5-13 shows the separation as seen by both satellites, as well as the tangential alignment of the chaser. The target starts to rotate only after the chaser successfully aligns itself, since the relative estimator requires the beacon of the target to remain within line-of-sight. The face-to-face separations, estimated independently by both satellites, are very similar. The velocity along the docking axis did not follow the trajectory predicted by the glideslope algorithm (Fig. 5-14). The chaser approached the target too quickly, causing it to suddenly stop and even back off at time 55 seconds. The chaser reached its berthing separation at time 66 seconds, where it maintained its close position and precisely aligned itself before the final push to capture. Successful capture occurred at 91 seconds.

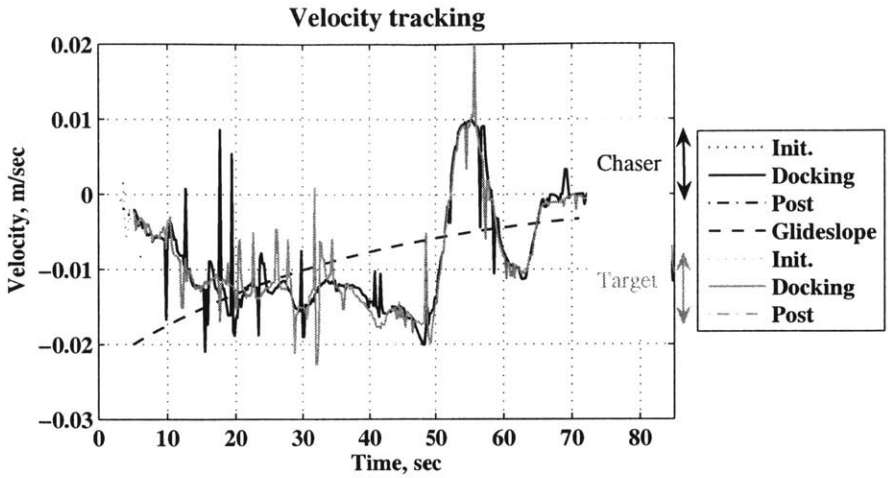


Figure 5-14: Velocity tracking for the autonomous docking to a tumbling target (rotating in a 2-D plane).

This experiment was performed four times at MSFC. All attempts were successful. This success was the result of adding a holding point (*berthing* mode) at close proximity with tight exit conditions, ensuring a good alignment prior to attempting the capture. This experiment proved that autonomous docking with a target tumbling, such that the docking axis sweeps a plane, was possible using the proposed GN&C architecture.

5.3.4 GN&C mode sequencing summary

Many iterations were performed in order to obtain an appropriate sequence of GN&C modes for the docking experiments presented in this section. The exit conditions for each mode were also the results of multiple iterations to ensure timely transitions while maintaining the chaser along an appropriate trajectory. These experiments permitted the tuning of different parameters in each GN&C module (controller gain, estimator process noise, minimum impulse bit) to ensure good docking performance.

The main lessons learned from the experiments performed in this section are listed below:

- a holding point (berthing) is necessary at a safe but close proximity to the target to make precise alignment corrections prior to capture;

Table 5.8: GN&C modes for the formation flight experiments on ISS.

GN&C mode	GN&C modules	Exit condition
EKF initialization	Global estimator	Timer
Position & attitude hold	Global estimator PD-type attitude controller PD position controller Pulse-width modulator	Timer
Position & attitude follower	Global estimator PD-type attitude controller PD position controller Pulse-width modulator	----------

- the closer to the target, the more constraining the exit conditions must be on each GN&C mode, to ensure that the tracking errors are within tolerance of the capture mechanism prior to contact;
- the controllers must be tuned together with the exit conditions to ensure a smooth and timely transition between modes at a reasonable fuel cost.

5.4 Flight experiments

Following the validation of the key GN&C modules (Chapter 4) and the launch of a second satellite by STS-121, formation flight experiments involving two satellites were performed on ISS in preparation for the docking experiments. The objectives of these experiments were:

- to demonstrate microgravity operations involving two satellites;
- to validate the segment of the GN&C architecture ensuring coordinated motion through inter-satellite communication;
- to collect valuable data to tune the controllers used to track a trajectory;
- to demonstrate a multi-satellite formation flight in close-proximity of the ISS (centimeter precision).

Two of these experiments are described below. Both use the GN&C modes and the mode sequence shown in Table 5.8 and Fig. 5-15, respectively. Basically, the follower satellite tracks the leader's attitude and maintains a constant offset with it.

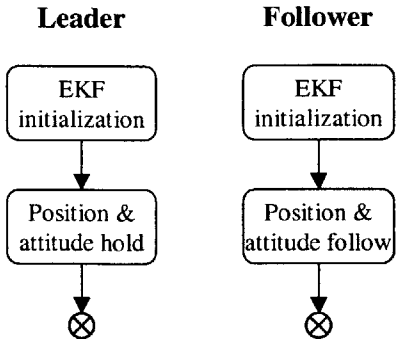


Figure 5-15: Mode sequencing for the formation flight experiments.

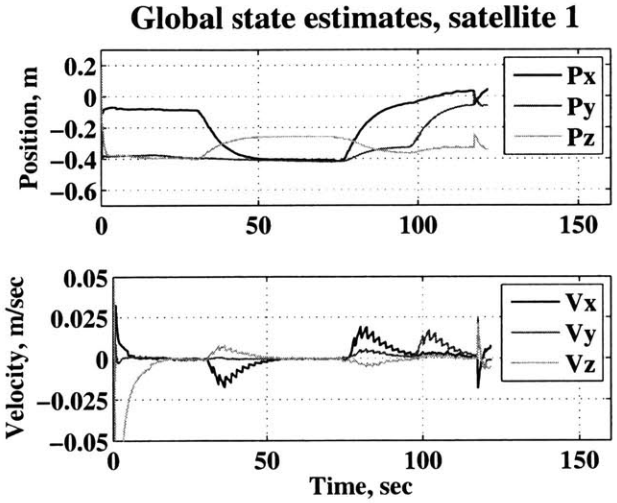
5.4.1 Position-hold, leader and follower

For this test, the leader is commanded to damp its linear velocity and maintain its attitude. The crew was required to perturb the leader to test the control system and get the response to a step input. When the leader is pushed, the follower satellite, tracking the attitude of the leader and holding its initial position relative to the leader, moves to follow the motion of the leader. The results for both satellites are shown in Figs. 5-16a and 5-16b. The estimator converged within 10 seconds. The crew applied a push to the leader in the -x direction at 30 seconds, a push in the +x direction at time 76 seconds and a push in the +y direction at time 98 seconds.

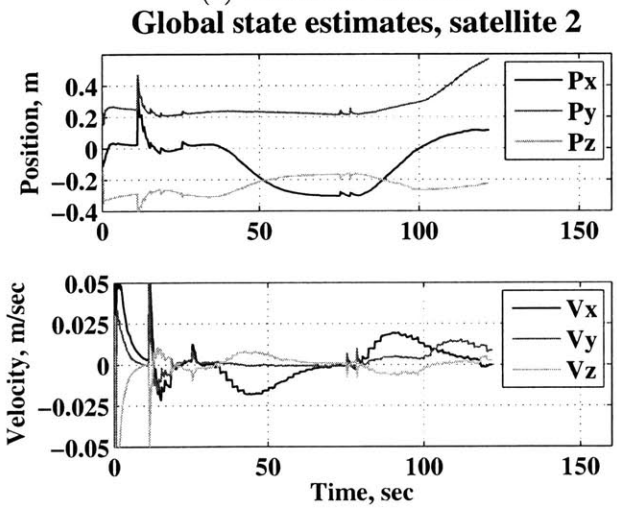
The response of the follower is seen in Fig. 5-16b. The state estimates of both satellites show a perturbation at time 118 seconds. Although no video is available to confirm it, it is suspected that this perturbation was caused by the crew manipulating both satellites at the end of the test (no IR interference was registered and the gyroscopes of both satellites registered the perturbation). One jump in the state estimates, caused by bad U/S measurement, occurred in the follower at time 11 seconds. This experiment was successful, as both satellites responded as expected to the step input by the crew.

5.4.2 Position-tracking, leader and follower

This time, the motion of the leader tracks a series of waypoints that are hard coded in the software. The follower tracks the attitude of the leader and moves with it to



(a) The leader satellite.



(b) The follower satellite.

Figure 5-16: Position hold experiment, leader and follower.

maintain its initial position relative to the leader. This test was designed to get the response of both satellites to step inputs in the leader. The results of the experiment are shown in Figs. 5-17a and 5-17b. For the first 20 seconds, the leader is commanded to hold its initial position and attitude. At time 20 seconds, it is commanded to move 40 cm in the +x direction. At time 50 seconds, it is commanded to translate 40 cm in both the +y and +z directions. Finally, at time 80 seconds, it is commanded to get back to its initial position.

The response of the follower is shown in Fig. 5-17b. It shows a lag of 5 to 10 seconds. The global state estimator performed very well throughout the experiment. Only one jump is present in Fig. 5-17b at time 70 seconds.

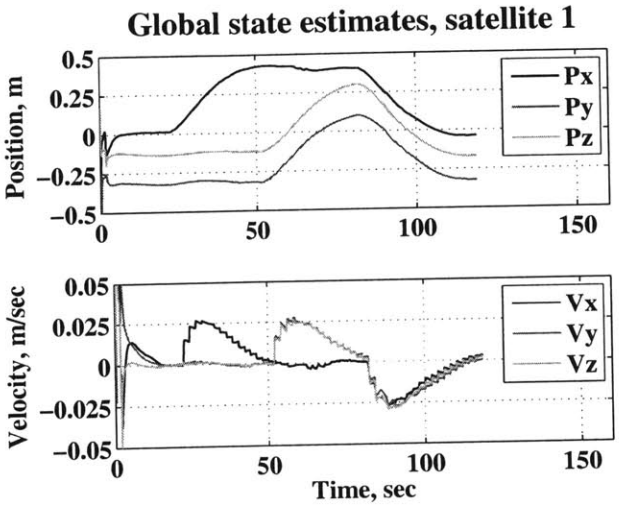
This successful test demonstrated the capability of the satellites to follow each other in a 3-D environment. Valuable data was collected to tune the controllers and reduce the lag in the tracking, in preparation for future formation flight and docking experiments.

The experiments presented in this section were the first three-satellite formation flight demonstrations, in the ISS, with centimeter-level precision control.[6] They validated the GN&C architecture for position and attitude tracking, which is the last step prior to performing the docking experiments. The next section presents the details behind the implemented architecture that is used in subsequent docking experiments.

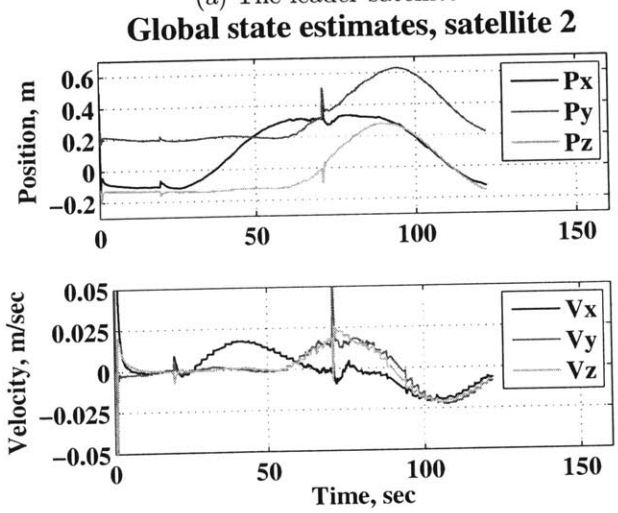
5.5 Implementation of the GN&C architecture

With the experience gained while performing the experiments shown above, a GN&C architecture was established for the execution of autonomous docking experiments on-orbit (Fig. 5-18). The frequency at which each module is executed is shown by the line type associated with the inputs of that module (a legend defines each line type). Two line types are associated with variable frequencies (indicated by $\approx$ and their averaged frequencies). The architecture uses a global estimator on each satellite, which proved to be more accurate and more stable than the relative estimators using

[6]The ISS was also part of the formation since U/S beacons were attached to it.



(a) The leader satellite.



(b) The follower satellite.

Figure 5-17: 3-D formation flight experiment.

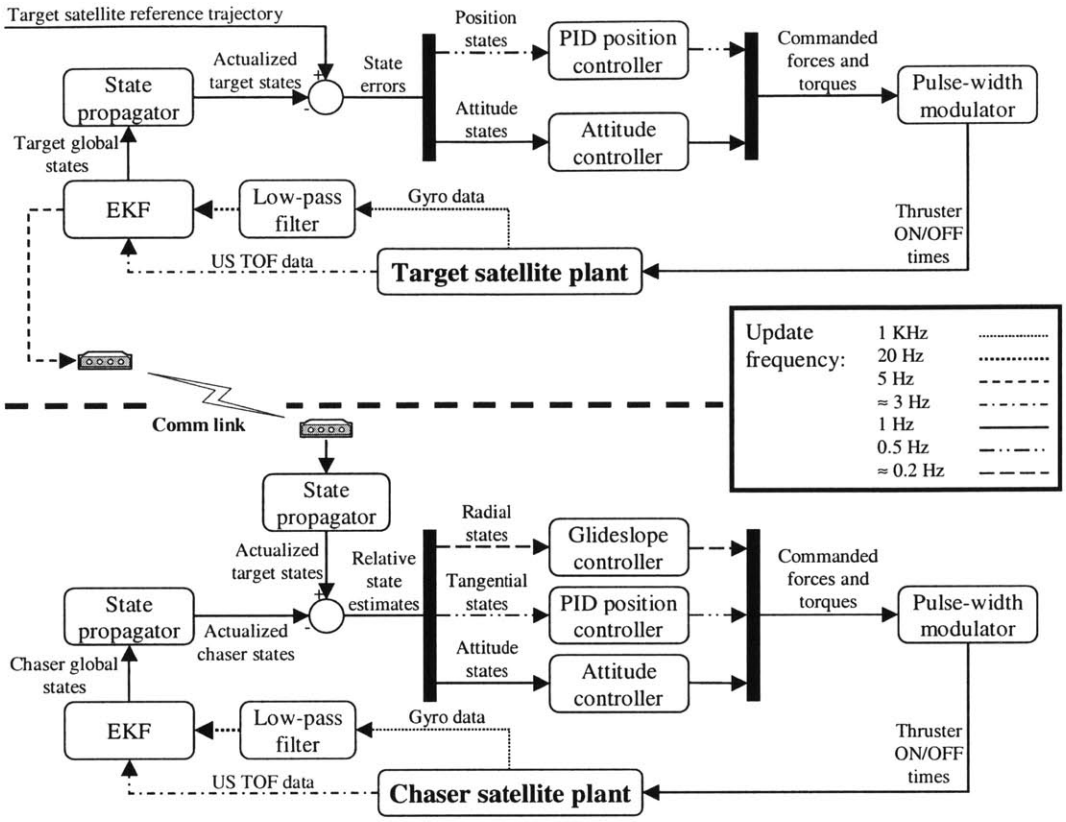


Figure 5-18: A detailed view of the GN&C architecture used for autonomous docking experiments.

a single beacon. Relative state estimates are computed by subtracting the states of the chaser from those of the target.[7] The state propagator box corresponds to the segment of the EKF used to propagate the states.

The following are necessary to better understand the software:

- A low-pass filter is necessary to process the gyroscope data prior to sending it to the EKF to attenuate an internal resonance at 338 Hz and prevent aliasing. The filtered gyroscope data is then down-sampled and sent to the EKF at 20 Hz.

- Because the modules can run in different software threads, depending on the level of computation they require, the state estimates need to be actualized with a state propagator every time a controller, running in a different thread, exercises them. The same idea applies to the state estimates, sent wirelessly by

[7]Both state vectors are expressed with respect to a coordinate frame attached to the ISS.

the target satellite, since the communication process induces a delay.

- The position controllers on the chaser satellite use relative state estimates, which are defined in a coordinate frame attached to the target body frame. The radial state of the chaser is along the target's docking axis. The tangential states of the chaser are in the plane perpendicular to the docking axis.

- It is also known that the thrusters on SPHERES generate U/S interference that can corrupt the measurements taken by the navigation system. Alternating between global navigation updates and thrust windows is therefore mandatory. Figure 5-19 illustrates the different thrust and navigation patterns implemented with this architecture. When multiple controllers are exercised simultaneously, the pattern is selected based on the controller requiring the largest thrust window.

- Experimentation has demonstrated better performance of the position and glideslope controllers when the attitude is well maintained. However, the attitude controller sampling frequency (set at 1 Hz) cannot be increased without reducing the U/S sensors sampling frequency. Consequently, the position controller sampling frequency was reduced to 0.5 Hz.

The GN&C architecture currently implemented for autonomous docking experiments has the advantage of being easily upgraded by updating or replacing the different modules comprising it. For example, one can easily replace the glideslope controller with a different position controller and use the resulting architecture for formation flight experiments.

5.6 Summary

This chapter provided an overview of the integration process to obtain a GN&C software architecture capable of executing autonomous docking. The software is obtained from the integration of GN&C modules covering state estimation, control, path plan-

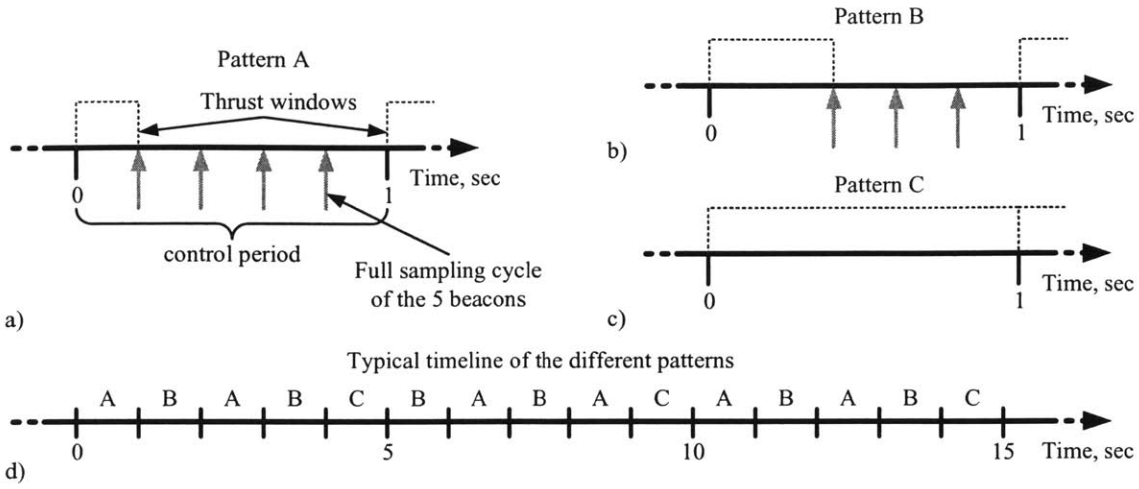


Figure 5-19: The thrust and navigation patterns used after executing a) the attitude controller alone, b) the attitude and position controllers and c) the attitude, position and glideslope controllers. The typical arrangement of the patterns for a docking experiment is shown in d). Pattern C does not present any beacon sampling.

ning and FDIR. A MVM module is in charge of the proper execution of the sequence of GN&C modes established for the experiment.

Some criteria leading to the selection of the GN&C modules to be integrated were presented, including an overview of the inputs and outputs of each module as well as their known limitations. Important hardware considerations like computational load, data and telemetry handling were also discussed.

Throughout the integration process, multiple ground experiments were performed to gain experience with different sequences of GN&C modes for different docking scenarios. Important lessons learned from these experiments were mentioned. Two formation flight experiments were also performed to demonstrate coordinated motion between two satellites and to collect valuable data to tune the different controllers onboard. These experiments were the first multi-satellite formation flight experiments performed in close-proximity of the ISS with centimeter level precision.

Finally, the outcome of this chapter is the implementation of a GN&C architecture for autonomous docking experiments on orbit. Many details were provided about the architecture, including the integrated modules, the frequency at which they are

executed and the key data being transfered between the modules. The next chapter will present autonomous docking flight experiments used to validated the GN&C architecture.

Chapter 6

Autonomous docking experimentation onboard the ISS

On August 19, 2006, after testing of the global metrology system, all the necessary SPHERES hardware, for performing autonomous docking maneuvers between two satellites, was confirmed ready for experimentation onboard the ISS. A series of four autonomous docking experiments, with cooperative and uncooperative targets, took place that same date. Five more advanced experiments were performed on the following test session on November 11, 2006, including the very first autonomous docking, with a tumbling target, in microgravity. Out of these nine experiments, seven were classified as very successful. The other two, both performed on August 19, 2006 were classified as partially successful as they made contact, but with a misalignment caused by instabilities in the state estimates, likely the result of multi-path.

This chapter presents the most important results collected during this research. Eight out of the nine autonomous docking experiments are discussed in detail, including:

- Docking to a cooperative target (two experiments)
- Docking to a drifting target
- Docking to a cooperative target with FD through filter innovation analysis

- Safe docking with a cooperative target (two experiments)
- Docking to a tumbling target (two experiments)

6.1 Docking to a cooperative target

Two slightly different variants of that experiment were performed on August 19, 2006. Both consist of a basic autonomous docking maneuver with a target actively holding its position and attitude. The objective of these tests was to validate the GN&C architecture, described in detail in Chapter 5, for a standard docking approach, following a straight path. The global estimator, the glideslope controller, the PD position controller and the attitude controller formed the architecture for that experiment. The chaser used relative state estimates to navigate in the test volume, which were obtained by subtracting its own global state estimates from those transmitted by the target, as shown in Fig. 5-18.

The two satellites were deployed $\approx$0.5 meter apart. During the initialization phase, both satellites acquired a navigation solution and oriented their docking faces at each other. The docking phase consisted of the glideslope controller bringing the chaser to a separation within 30 cm.[1] The chaser was then commanded to closely align itself, using a high bandwidth position and attitude controller, while slowly drifting toward the target at a safe approach velocity.

Figure 6-1 illustrates the results. The left plot shows the separation along the docking axis at different phases of the maneuver; the horizontal target line indicates when the geometric centers are one diameter apart from each other, resulting in contact. The right plot shows the tangential alignment between the satellites in a plane perpendicular to the docking axis.[2] The state estimates are very smooth throughout the experiment. After initialization, the chaser remained closely aligned with the target up to initial contact with it at 63 seconds. Because of a small misalignment and a slightly large approach velocity (9 mm/sec instead of the expected 5 mm/sec), the

[1] In this chapter, the separation refers to the distance between the geometric centers of both satellites.

[2] The docking axis is defined as the x-body axis of the target satellite for all docking experiments.

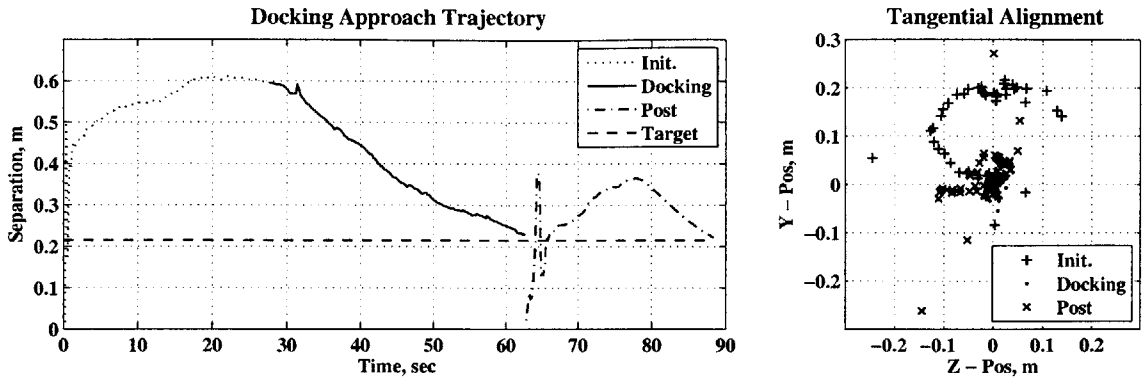


Figure 6-1: Autonomous docking with a cooperative target in microgravity.

velcro docking mechanism did not latch properly on contact, and the chaser bounced off the target. This can be seen in the data labeled *Post* in Fig. 6-1 and in the video of the experiment.

This experiment was overall a success, although capture was not accomplished because of a misalignment at contact. It provided valuable data to tune the glideslope controller, but also showed the need for the chaser to better align itself with the target prior to contact.

Figure 6-2 shows the results from the second run. In the initialization phase, both satellites first acquired a position and an attitude fix from the global estimator. This time, both induced a random tumble, damped it and pointed at each other, also during the initialization phase. For the remainder of the experiment, the target was commanded to hold its position and to keep pointing at the chaser. At 39 seconds, the chaser initiated its approach. A jump in the state estimates of the target induced an unexpected tumble at 53 seconds, while another jump in the state estimates of the chaser occurred at 64 seconds and resulted in a slight collision at 69 seconds. At 83 seconds, the chaser initiated its final open-loop capture firing and made contact with the target at 87 seconds.

The controllers behaved as expected. The root cause of the perturbations in the approach trajectory followed by the chaser was the presence of jumps in the state estimates. Since the video of the experiment clearly did not show jumps in the

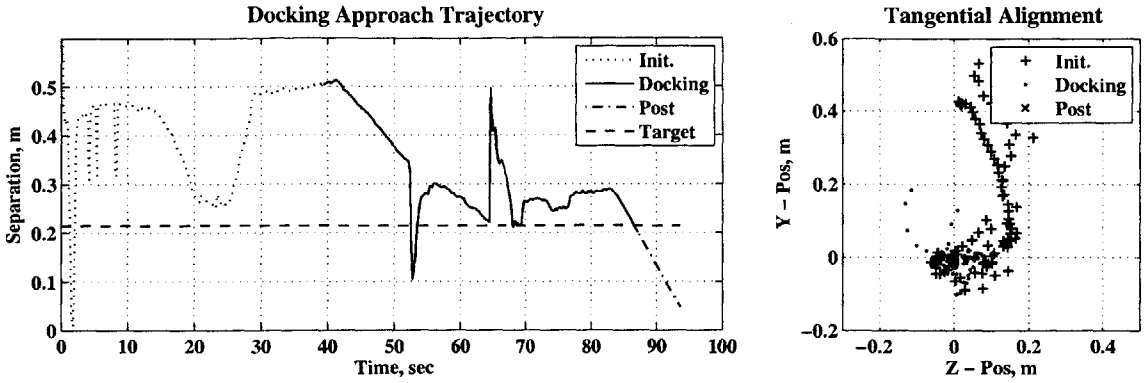


Figure 6-2: Autonomous docking with a cooperative target in microgravity (chaser initially tumbling).

separation as plotted in Fig. 6-2, it was determined that the problem was a navigation error. A close analysis of the results indicated that no IR noise was present in the environment during this experiment. These jumps are believed to be caused by multi-path in the U/S signal transmission when the satellites are in close proximity. They were not observed during ground testing using the same GN&C system, probably because of the different beacon locations around the test volume. This experiment clearly illustrates the need for very smooth state estimates when performing close proximity operations. Following the results from this experiment, a decision was made to implement the FD module, through filter innovation analysis, introduced in the previous chapters, prior to the next test session.

6.2 Docking to a drifting target

The objective of this experiment, also run on August 19, 2006, was to demonstrate an autonomous docking maneuver with an uncooperative target. The same GN&C architecture was used as for the cooperative target, except that the thrusters on the target were commanded to remain closed after the separation distance dropped below 0.5 meter.

Results are presented in Fig. 6-3. Both satellites acquired a navigation solution and pointed at each other during the initialization phase which lasted until time

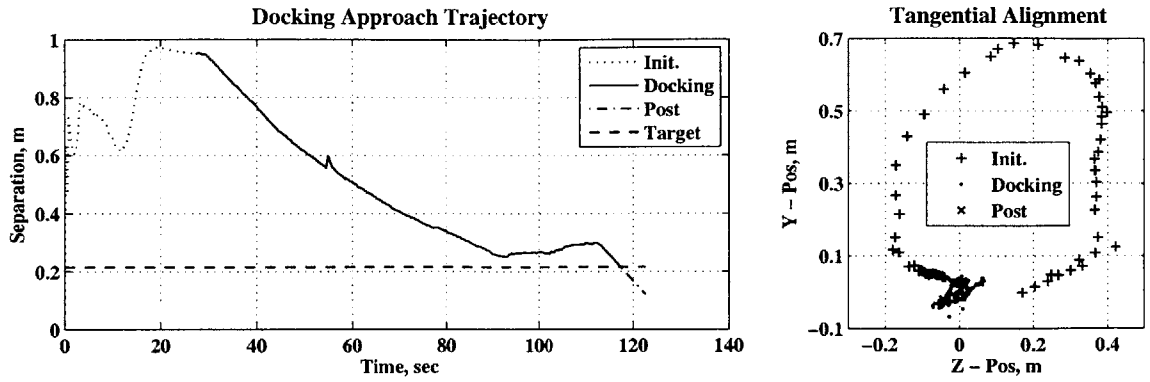


Figure 6-3: Autonomous docking with a drifting target in microgravity.

27 seconds. The chaser then initialized its approach. Following the interruption in thruster actuation of the target (at 63 seconds), every time the chaser reduced its closing rate by firing a plume in the direction of the target, the target was pushed away. The plume impingement force increased as the chaser closed in. This effect can be clearly observed in between 91 seconds and 112 seconds, where the separation distance slowly increases after a braking pulse by the chaser. Contact occurred at 117 seconds, as confirmed by the gyroscope data and by video of the experiment. Each plume impingement produced minimal angular rate changes (of less than 0.5 deg/sec) but substantial velocity changes of up to 6 mm/sec. Overall, this experiment was very successful and provided valuable information on the potential effect of plume impingement when docking with an uncooperative target.

The first three experiments presented in this chapter demonstrated the capability of the GN&C software, based on the architecture proposed in Section 5.5, to perform a straight path autonomous docking maneuver with a cooperative and a drifting target. However, none of these experiments led to a successful capture of the target because of misalignment at contact.

Following this test session, many improvements were implemented to increase the chance of a successful capture. The most important of these was the implementation of the FD module through filter innovation analysis, which increased the robustness of the GN&C software to measurement errors, especially the ones caused by multi-path or temporary IR noise. This resulted in more robust state estimates even with

satellites in very close proximity [85]. It enabled the use of closed-loop control during the capture stage, as opposed to the open-loop thruster firing that was used in the previous experiments, when the chaser was within 30 cm separation from the target. Moreover, the increased robustness allowed the implementation of a more accurate PID-type control (instead of PD).

6.3 Docking to a cooperative target with FD enabled

During the fifth test session, on November 11, 2006, another cooperative docking experiment was conducted. Figure 6-4 shows the results. The initialization phase completed at 33 seconds after both satellites pointed at each other. The chaser then initialized its approach toward the target. At a separation of approximately 29 cm, the chaser terminated its glideslope approach. It switched GN&C modes, slowly moving down to 26 cm separation and precisely aligning itself. It then switched GN&C modes again and executed the capture maneuver with closed-loop control at time 127 seconds; contact occurred but the Velcro used as the capture mechanism did not latch. For most of the approach, the tangential alignment was maintained within a box of 2 cm $\times$ 2 cm (within docking tolerance), which confirmed the good tuning of the GN&C system. The video of the experiment and communication from the crew also confirmed precise alignment at contact.

Figure 6-5 shows the filter innovation computed by the EKF on both satellites for every set of U/S measurements. Any point above the rejection threshold represents a faulty set of measurements. Although the chaser did not collect any bad measurements, four bad sets of measurements were detected on the target satellite and successfully rejected online. Had the implementation of the fault detection system using EKF innovation been omitted, the measurement errors would have caused the state estimates to jump, as observed in previous experiments.

Close examination of the data showed that, although it did not have an impact

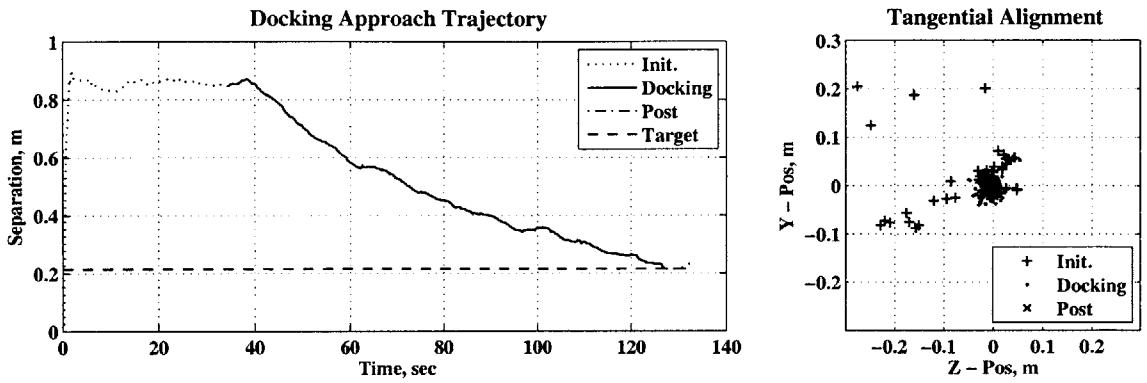


Figure 6-4: An autonomous docking with a cooperative target in micro-gravity with U/S measurement FD.

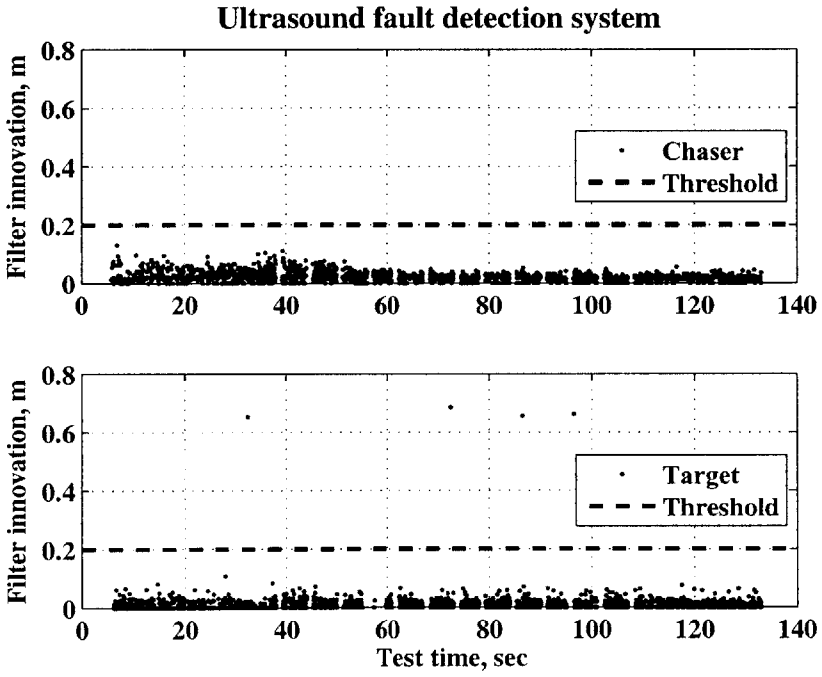


Figure 6-5: Navigation error detection system, docking to a cooperative target.

on the success of the experiment, messages transmitted between satellites sometimes took much longer than expected to reach their destination. This illustrates the importance of handling acknowledgements between satellites when communicating critical information. Also, because of the good attitude and tangential alignment, the contact between both satellites was barely perceived by the gyroscopes, confirming the need for a different docking sensor.

6.4 Safe docking with a cooperative target

This test, conducted during the fifth test session, followed a trajectory demonstrating safe rendezvous [15] with a simulated detection of a failure toward the end of the trajectory. This is called a *safe docking* because when detection of a failure occurs, the GN&C system responds by shutting off all thrusters, forcing the satellite to enter a drift mode. This drift mode will either cause the chaser to drift past the target or cause the chaser to dock with the target at an acceptable impact velocity. The time of the simulated failure detection is programmed to occur randomly within the last few seconds of the test, when the optimized trajectory has been guaranteed to be safe in the presence of failures. Since the optimized trajectory is designed a priori, without knowledge of the failure time, this test demonstrates the ability of the safety algorithm to handle arbitrary failures that occur during that window. A similar GN&C architecture, as in the other docking tests during this test session, is used. However, a PID control module, replacing the glideslope module, is used to follow a pre-computed safe trajectory. Also, once the initial location of the target is determined, it becomes the target location for both satellites throughout the experiment (and therefore the expected docking location).

Two runs of that experiment were performed. Only the first one involved the simulation of a failure, while the second was intended to demonstrate docking by following a safe trajectory. The results of the first experiment are shown in Fig. 6-6. The left plot shows a close-up view of the separation just before capture, while the right plot shows the projection of the trajectory of the chaser on the y-z plane

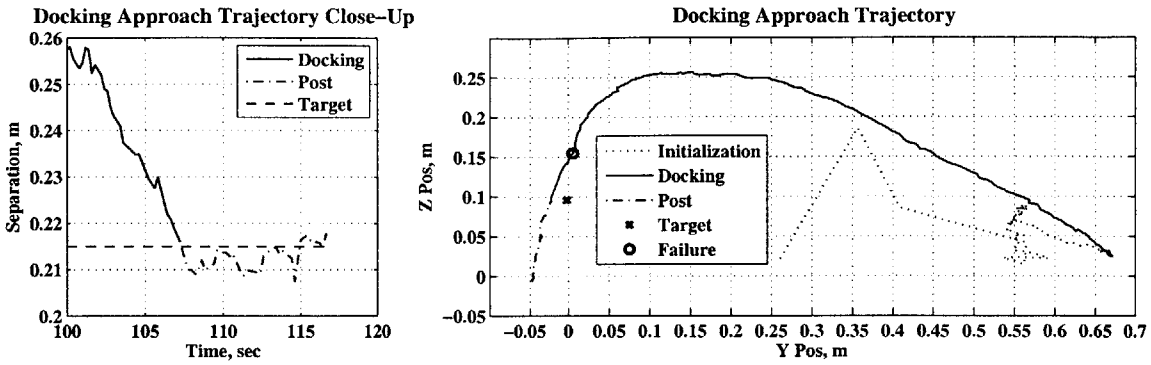


Figure 6-6: Autonomous docking following a safe trajectory, with faults.

of the coordinate frame attached to the ISS. During the initialization phase, both satellites acquired a navigation fix, pointed at each other, and moved to a separation distance of approximately 90 cm (performed by the chaser). The chaser started to follow the pre-computed approach trajectory at 50 seconds. The GN&C shutdown occurred at 102 seconds, at which point the chaser disabled its thrusters and entered a drift mode as expected. Video of the experiment showed that the chaser successfully docked 107 seconds into the test, even in the presence of a detected failure. Since a safe trajectory is defined specifically in terms of collision avoidance, and the terminal docking box was not inside the failure avoidance region,[3] the resulting docking, even with disabled thrusters, is consistent with a safe maneuver [14].

For most of the time, the target held its position with little state error (within ± 1 cm along each axis) and the chaser spacecraft followed its trajectory accurately. The maximum deviation from the trajectory occurred at the point of maximum curvature in the desired trajectory, where the rate of change reached its maximum, indicating that the controller gains should be tuned for future experiments. The target point on the right plot of Fig. 6-6 represents the reference point used to generate the desired trajectory. This point corresponds to the initial location of the target, not the location at contact, which is represented by the junction between the *Docking* and the *Post* trajectories. The actual docking occurred ≈ 2 cm from its intended location, which was caused by a combination of the tracking errors from both the chaser and

[3]Failures in the terminal docking box result in docking, while failures in the avoidance region result in an avoidance maneuver.

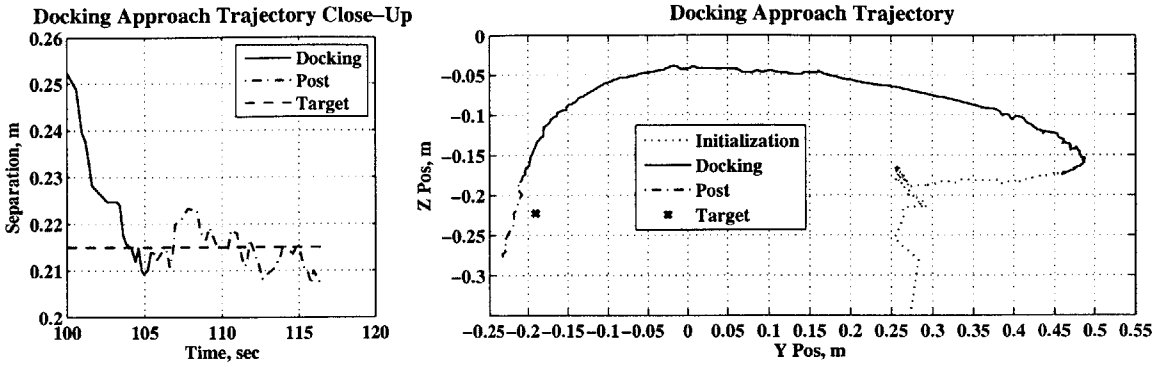


Figure 6-7: Autonomous docking following a safe trajectory, no faults.

the target.[4]

Figure 6-7 shows the results of the second run of the experiment. The initialization phase proceeded as previously, although it ended with the chaser having a significant positive velocity along the y-axis because of a deployment too close to the target. This increased the lag in the tracking of the trajectory, which explains the different trajectory profile from the one observed in Fig. 6-6. Successful docking occurred at time 106 seconds, but ≈4 cm from the intended contact point. The left plot of Fig. 6-6 shows a separation at the time of contact oscillating between 0.21 meter and 0.215 meter, consistent with the goal of a safe docking trajectory (near zero approach velocity along the docking axis, resulting in a tangential approach at contact).

The two experiments presented in this section demonstrated the flexibility of the GN&C architecture presented in Section 5.5. In this case, the simple replacement of a GN&C module with another one permitted the testing of an exotic docking scenario. This experiment was performed on hardware in a microgravity environment, without the risk of losing the hardware would a problem have occurred.

6.5 Autonomous docking to a tumbling target

After the improvements implemented between August 19, 2006 and November 11, 2006, the GN&C architecture was determined ready for more challenging and in-

[4]The target was also regulating itself around the target point.

novative scenarios, like docking to a tumbling target. The two repetitions of this experiment on November 11, 2006 were the first autonomous docking maneuvers, to a tumbling target, accomplished on orbit. The objective was to validate the GN&C architecture presented in Section 5.5 for such a maneuver. To simulate a tumbling target, the target satellite was commanded to maintain an angular rate of -2.25 deg/sec around its z-body axis during the docking approach phase. The use of a constant rate allows comparison with other docking experiments, also involving a tumbling target and performed in Test Session 06, that use a pre-computed approach trajectory.

The results of the first run are shown in Fig. 6-8. The initialization phase consisted of both satellites acquiring an absolute navigation fix, pointing at each other, rolling into position[5] and reducing the separation down to ≈60 cm (performed by the chaser). This phase was completed at 56 seconds. The target began its constant rotation about its z-body axis, as illustrated by the rotation angle shown in Fig. 6-8. The chaser then began its approach toward the target. A PID position controller maintained its docking face aligned in front of the target's docking face. The glideslope controller was used to track a decreasing approach velocity profile along the docking axis. With the target rotation axis perpendicular to the docking axis, the docking axis swept out a plane. The result was a spiral motion of the chaser in the plane of rotation as shown in Fig. 6-9 (the origin of the coordinate frame is located at the reference position used by the target satellite). The video of the experiment and the data indicates that contact occurred at ≈111 seconds with both satellites well aligned.

Because of the increased amount of control input required to maintain a spiral trajectory, the relative trajectory plot shown in Fig. 6-8 appears more perturbed than the ones shown in other docking experiments. The separation is still mostly decreasing during the docking approach, and the tangential alignment remained mostly within a 2 cm × 2 cm box for the last 20 seconds prior to docking.

Video of the experiment shows the astronaut taking pictures of the satellites at 109 seconds. Camera flashes and range finders are known sources of IR, which in-

[5]For simplicity, the angular trajectory, used to bring the satellites aligned and facing at each other, did not follow the shortest angular path. After initial pointing, the chaser still had to regulate its roll while approaching the target, to align its docking face with the target's docking face.

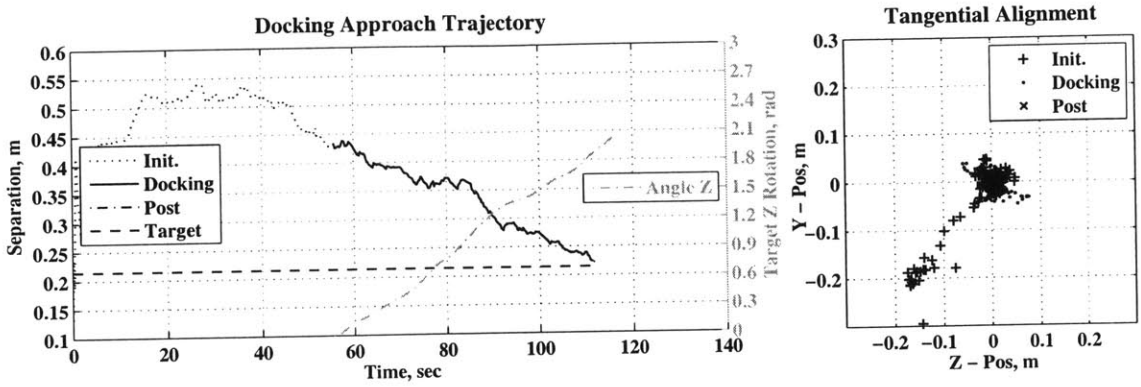


Figure 6-8: Autonomous docking with a tumbling target, first run.

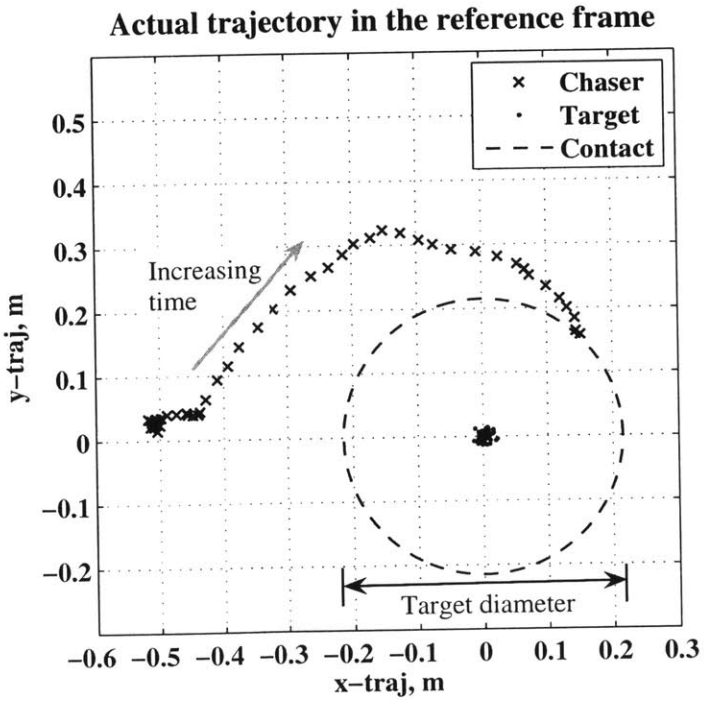


Figure 6-9: Autonomous docking to a tumbling target: trajectory in the rotation plane, first run.

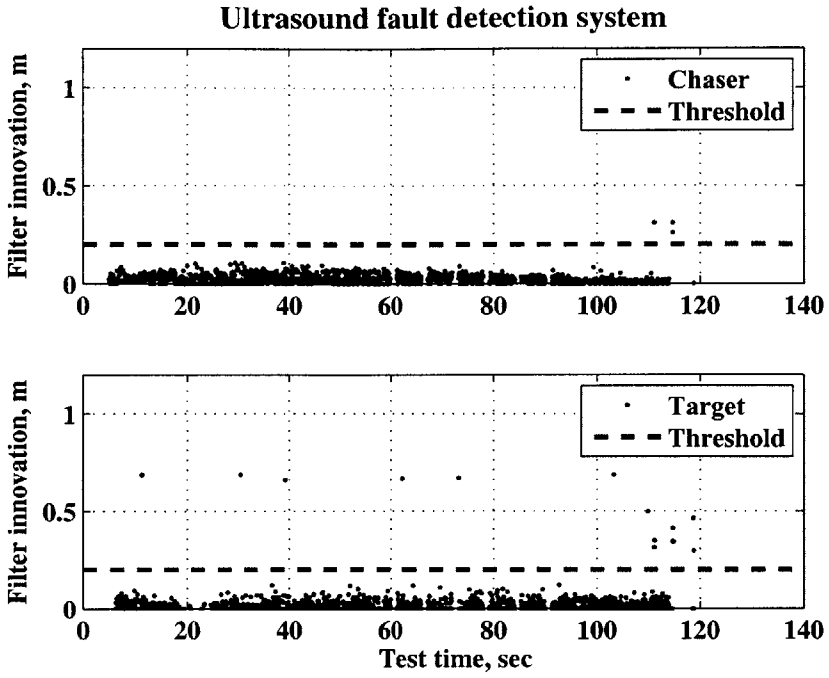


Figure 6-10: Navigation error detection system, docking to a tumbling target, first run.

terferes with the U/S system (Section 3.1.2). Figure 6-10 shows the filter innovation used by the measurement error detection system on both satellites. A couple of points are observed above the rejection threshold for the chaser. These occur precisely at times when unexpected IR flashes were also recorded and when the crew was observed taking pictures of the satellites. Other bad measurements were observed above the threshold for the target and were probably caused by multi-path. The faulty measurements were successfully rejected, allowing the state estimates to remain stable on both satellites in the presence of measurement errors. Therefore, this experiment went beyond its intended objectives and demonstrated the capability of the GN&C architecture implemented on SPHERES to perform an autonomous docking maneuver to a tumbling target in the presence of failures.

Results for the second run are shown in Fig. 6-11 and in Fig. 6-12. This time, the chaser was initially approaching too quickly and came close to docking around time 95 seconds. Nevertheless, it stopped its approach, flew in close formation for over 20 seconds to properly align itself, and successfully captured the target at 116 seconds.

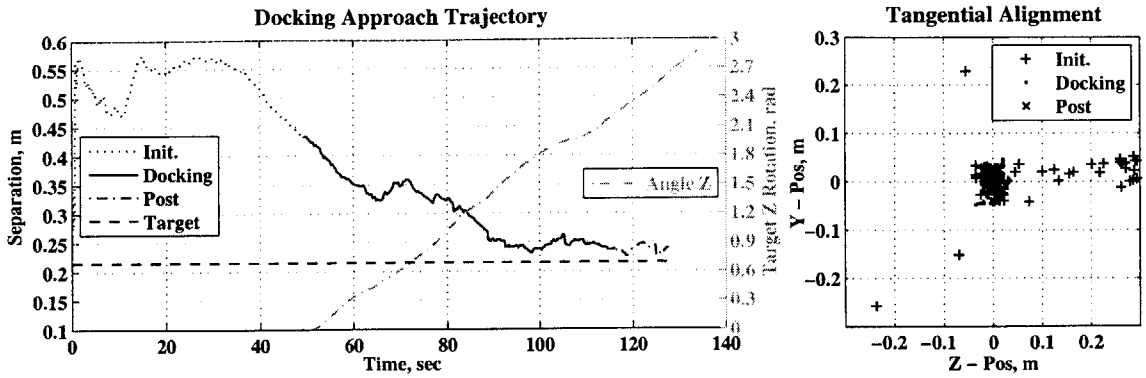


Figure 6-11: Autonomous docking with a tumbling target, second run.

The overall similarity between the trajectories shown in Figs. 6-9 and 6-12 confirm the repeatability of this experiment.

Once again, a close analysis of the telemetry on the chaser showed the presence of sensor measurement errors caused by IR noise just prior to docking. The video of the experiment allowed correlation between the IR noise and the crew taking pictures of the experiment. Figure 6-13 shows the innovation corresponding to each set of U/S sensor measurements on both satellites. The target satellite is again subjected to bad measurements, as in the first run, all of which were successfully detected and rejected online.

Although safe contact occurred in both experiments, the second one achieved full capture, resulting in the first successful autonomous docking to a tumbling target ever achieved in microgravity. These results demonstrated the capability of the GN&C architecture, using common estimators and controllers, to perform an autonomous docking maneuver with a target tumbling at a rate of $\approx$2 deg/sec in a planar rotation (typical tip-off rate induced by a variety of common launch vehicles).

6.6 Summary

Between the two ISS test sessions on August 19, 2006 and November 11, 2006, a total of nine autonomous docking experiments were attempted with a cooperative, drifting and even a tumbling target. These experiments represent the most significant

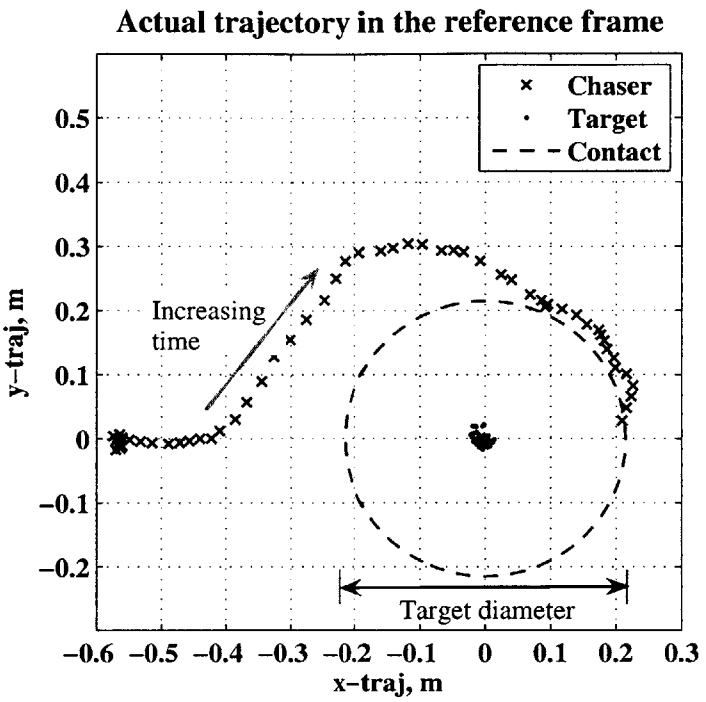


Figure 6-12: Autonomous docking to a tumbling target: trajectory in the rotation plane, second run.

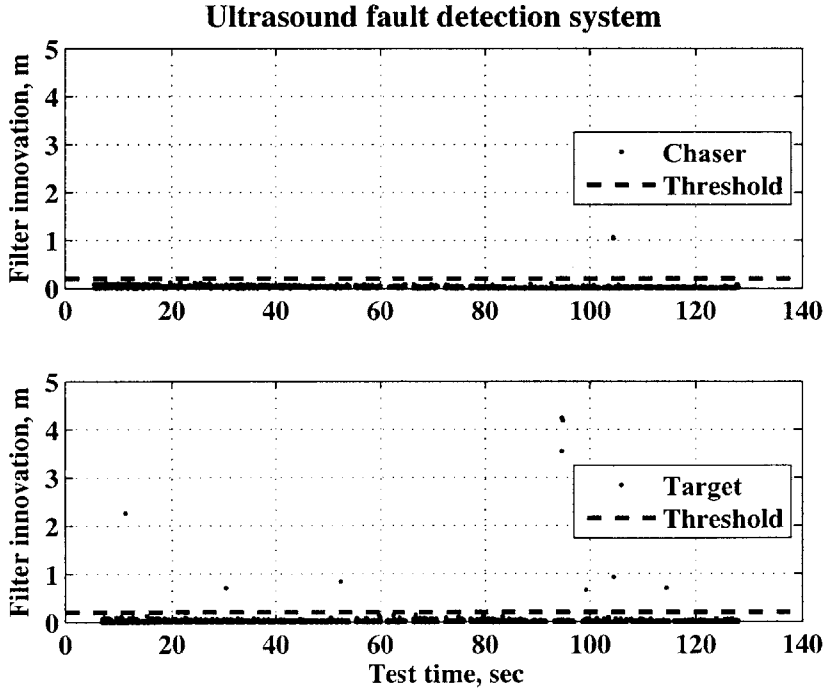


Figure 6-13: Navigation error detection system, docking to a tumbling target, second run.

experimental results related to the research in this thesis.

On August 19, 2006, out of four docking experiments, two were very successful (including one with a drifting target), while two others were slightly misaligned at contact. Following this test session, the addition of a FD module through filter innovation analysis significantly improved the robustness of the global estimator to bad measurements caused by multi-path and IR noise.

On the following test session (November 11, 2006), five docking experiments were attempted. They were all very successful and made closely aligned contact. Two of these experiments were performed using a safe approach trajectory, which is robust to failures occurring a few seconds prior to contact, when safety is critical. Two others were performed with a tumbling target, resulting in the first autonomous docking with a tumbling target ever achieved in microgravity. All docking experiments were successful in spite of unexpected measurement errors, induced by the crew when taking pictures of the satellites, which were successfully detected and rejected online.

The results presented in this chapter proved the accomplishment of an important objective of this thesis, which was to demonstrate in a relevant environment a safe docking with a tumbling target in the presence of anomalies. The next chapter will discuss the process, developed as part of this research, that verifies the GN&C software prior to flight.

Chapter 7

Synthesis of a verification & validation process

An autonomous docking maneuver is complex and presents considerable risk to the two spacecraft involved. To reduce the risk of failures, the proper function and performance of the following software modules must be verified prior to flight [40]:

- The algorithms used by all onboard systems controlling the docking (i.e., the GN&C, MVM and FDIR algorithms);
- The control software in which the algorithms are implemented;
- The interface with the sensors required for the docking trajectory and attitude control;
- The interface with the reaction control system (actuators);
- The integrated algorithms, software, data management system, sensors and reaction control system that together form the onboard docking control system.

In the previous chapters of this thesis, a series of ISS experiments were presented that validated these modules for SPHERES (since the ISS is the real operational environment for SPHERES, the term *validated* is used here instead of *verified*). However, for an operational space system, direct access to space for validation purposes is usually not possible before the actual mission is flown. Therefore, systems engineers must

rely on verification techniques to reduce the risk, associated with the mission, to an acceptable level prior to the launch.

This chapter synthesizes a process for the V&V of GN&C software for flight testing on an experimental free flying platform in the ISS, which is a contribution made as part of this research. There exists a body of literature covering the V&V of both software and hardware throughout the development phases of space systems [36, 40]. However, they do not provide much details on the verification process prior to flight, after the hardware is manufactured and the software is completed. The process shown in this chapter concentrates on that later phase. It was derived from the experience gained through this research.

The first section of this chapter proposes a role that a microgravity testing platform like SPHERES can fulfill in terms of V&V. The second section presents the V&V process used by the SPHERES team to verify the flight software prior to upload into the satellites. Finally, the third section shows a new V&V process, that can further increase confidence in the flight software architecture, after a SPHERES simulation, written in MATLAB®, was developed through this research.

7.1 SPHERES as a validation platform

There exists a number of ground testing facilities dedicated to the verification of software and hardware components necessary for the docking of two satellites. The NRL 12 DOF robotic facility [27] and the European EPOS simulator [55] are two examples. The approach used by these facilities has been an attempt to replicate, by mechanically actuated platforms, the dynamics involved in the docking problem. However, Fehse [40] pointed out that each of the following requirements alone is difficult to implement, and their combination is almost impossible unless important compromises are made:

1. Six DOF per satellite

2. Two satellite models with accurate mass properties

3. Compensation for gravity effects

4. Correct contact velocities

5. Realistic translational and rotational misalignments

Because of the high number of tests required to verify that all the requirements are met, and because of the high cost associated with integrated testing on these facilities [40], only a few scenarios can be verified that way. Verification of the *full required performance spectrum* can only be effectively performed using software simulation tools running faster than real time. Therefore, it is necessary to validate the simulation tools to give credibility to the verification process.

Software simulation tools (including simulation models and simulation programs) are validated in many ways. Validation of the simulation models is usually performed by analysis, through hardware-in-the-loop testing or by comparison with validated data. In a similar way, simulation programs are validated by comparing simulation outputs with data from past missions, or with data from another simulation that has already been validated.

SPHERES can handle most of the five requirements presented above, since it is composed of satellites operating in microgravity. It can be used to validate simulation tools by providing them with actual flight data from relevant subsystems for comparison purposes. What distinguishes SPHERES from traditional satellite formation flight ground testing facilities is its ability to provide six DOF per satellite in a microgravity environment. It is also unconstrained in terms of attitude operational range and therefore is capable of simulating a wide range of autonomous docking maneuvers to a tumbling satellite. Since it is an actual satellite with all the common subsystems, it is also prone to hardware anomalies that an actual satellite can face during a docking maneuver, as shown in Table 7.1.

SPHERES cannot be directly used for the V&V of an actual mission since it does not carry the actual hardware for the mission, nor does it run the same software. However, the following process can be used to validate a simulation program that will be used to verify flight software on an actual mission:

Table 7.1: Anomalies occurring in past missions involving autonomous rendezvous and/or docking

Pasts Missions	Anomalies	SPHERES equivalent
Cosmos-186, Cosmos-188	Misaligned capture	Chaser bouncing off when misaligned
ETS-VII	Thruster anomaly	Occasional thruster stuck-OFF
XSS-10	Telemetry dropout in close-proximity operations	Occasional communication interruptions
DART	Faulty GPS receiver causing divergence of state solution	Multi-path before convergence of the EKF, leading to divergence
Orbital Express	Bad initialization of the GPS-INS system	Multi-path before convergence of the EKF, leading to divergence

- The integrated simulation representing the actual satellite can be programmed such that physical parameters of that satellite are in a separate module (e.g., mass properties, sensor and actuator models, etc.).
- Swapping that module can then allow for the verification of the simulation program for different satellites.
- Using the physical parameter module for SPHERES, the integrated simulation program can be validated up to TRL-6,[1] by comparing the data it generates with that collected on SPHERES. This can be achieved at an early stage of the design process, and does not require testing the mission on actual hardware.
- Swapping the physical parameters module back to the one of the actual system, the verification process can then be performed with increased confidence.

A similar process as the one introduced above has been used at the MIT SSL to verify and validate GN&C software for experimentation on SPHERES inside the ISS. In this case, the actual mission is represented by SPHERES in the ISS, while the verification platform is SPHERES on the MIT SSL 2-D air table. Details are provided in the next sections.

[1]System/subsystem model or prototype demonstration in a relevant environment, more details can be found in Appendix A.

7.2 Initial V&V process

On many occasions, the SPHERES team provided NASA with flight software to be uploaded to the SPHERES satellites inside the ISS, for the purpose of performing various experiments. This flight software contains all of the core interfaces with the hardware (housekeeping, handling of the interrupts, interface with the sensors, actuators and communication system) as well as the GN&C software for each experiment.

Over the years, many ground-based experiments helped to support the development of the flight software (Chapters 4 and 5). A simulation written in the same programming language as the one used to program SPHERES (C programming language) was developed in parallel for the purpose of verifying compiled software for SPHERES (Chapter 3). Unfortunately, the simulation was not ready by the time flight operations started in May 2006. Therefore, the SPHERES team had to rely on its experience gained in previous ground experiments to develop a process that would allow flight software verification using the three satellites at the MIT SSL 2-D air table.

The resulting process is illustrated in Fig. 7-1. It is composed of three main gates, each being characterized by a series of questions that need to be answered affirmatively. The first gate uses 2-D laboratory experiments to determine if the expectations or requirements can be met with the implemented GN&C architecture. The second verifies if the code runs properly after changing the architecture parameters from 2-D operations to 3-D operations. Finally, the third gate is used to check, on orbit, if the code has been uploaded to the satellites properly and if the basic functions, such as thruster actuation and reading of the gyroscopes, are working nominally.

The process starts with the integration of the GN&C modules, that were validated through previous successful flight operations whenever possible, into the GN&C architecture. All modules are designed from the very beginning to accommodate 3-D operations. The software is designed such that all the physical parameters, that change when conducting tests in 2-D versus 3-D, are contained in one file that can be swapped easily prior to compiling the software. Such parameters include mass,

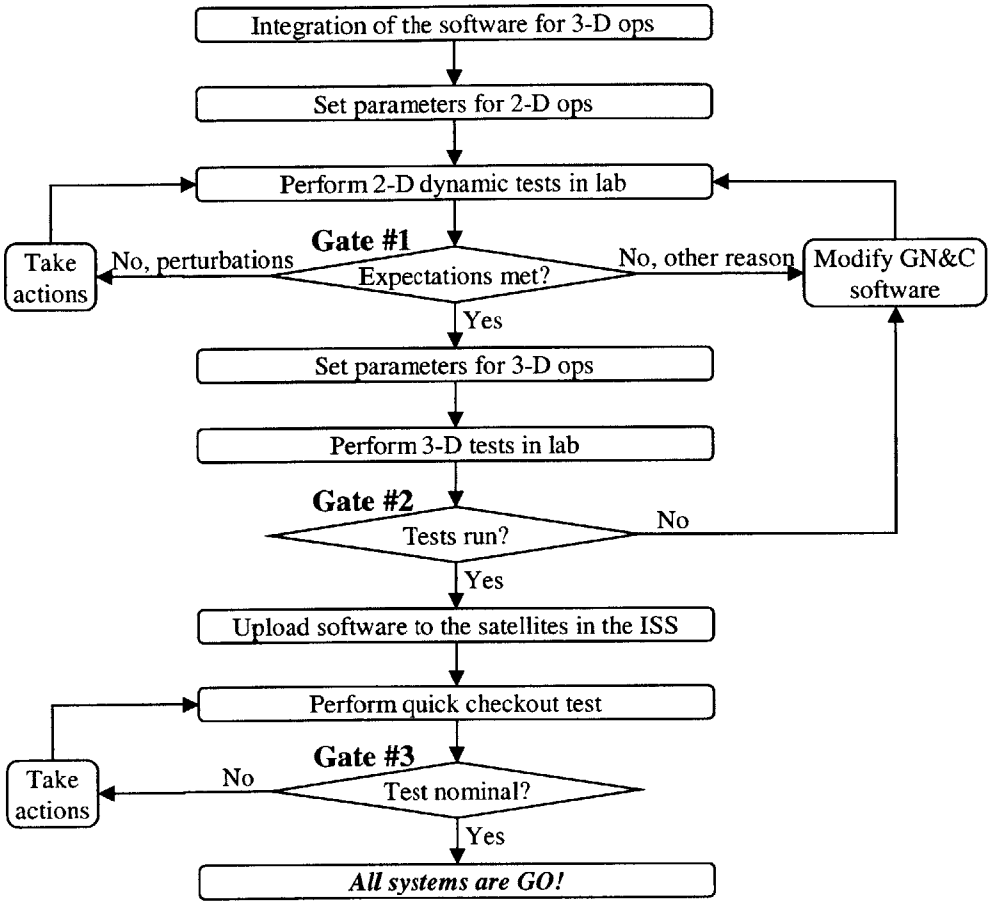


Figure 7-1: Process used to verify the software before uploading it to the ISS.

inertia, controller gains, etc.

Initially, the parameters are set for 2-D (three DOF) operations. This software is then tested in a 2-D environment on the MIT SSL air table. For each experiments, the user verifies that all expectations (requirements) related to the dynamic response of the system are met. This step involves performing the experiments with different sets of initial conditions, trying to capture the worst case scenarios. For a given set of initial conditions, an experiment often has to be repeated a couple of times because of table imperfections (friction, slopes, presence of dust). There are a couple of important questions, common to a wide range of experiments, that need to be answered affirmatively in order to proceed past Gate #1:

- Are the satellites following their expected trajectory?

- Are the satellites moving smoothly along their trajectory (as opposed to oscillating around it)?
- Are the transitions between the different maneuvers occurring in a timely manner?
- Are the final states of the satellites corresponding to the predicted ones?
- Is there a clear similarity between the telemetry and the observations throughout the experiment?
- Is the experiment repeatable?

A negative answer to any of these questions raises a red flag and forces the user to assess what is the likely source of the problem. If it is perturbations on the table (e.g., slope, dust, bumps and cracks), the user takes proper actions, and either cleans and levels the test area, or moves to a different test area. If the problems appears to be an error in the GN&C software, the user must modify the GN&C software accordingly, to solve the problem.

By experience, this gate is the most stringent in the process illustrated in Fig. 7-1. Since the MIT SSL 2-D air table constrains the movement of the satellites in a plane, the trajectory must be designed such that this testing plane corresponds to the most challenging one in terms of GN&C (e.g., for a planar trajectory, the most challenging testing plane is the plane that contains the trajectory).

The next step requires the user to replace the file containing the parameters for 2-D operation by the one for 3-D operation, to recompile and to repackage the final software in its flight configuration. Since the dynamic response has already been verified for the 2-D case, there is no need to verify it with the parameters set for 3-D operations. However, each experiment is still run once on the ground, with these parameters and the same Graphical User Interface (GUI) as the one used in the ISS (the flight GUI), to verify the correct software operation in its final configuration.[2] The following questions are addressed (Gate #2):

[2]Correct operation stands for performing all the basic functions like computing state estimates, sending telemetry, firing thrusters, etc.

- Are the tests initializing?
- Is the state estimator converging to a correct solution?
- Are the thrusters actuating as expected?
- Is telemetry being downloaded and sufficient for debugging purpose and reconstitution of the experiment through simulation?

Once again, a negative answer to any of these questions requires the user to make appropriate changes to the GN&C software. If every question is answered affirmatively, the software is sent as is to NASA and uploaded to the ISS.

The last step of this process consists in performing a quick checkout test after uploading the software to the satellites in ISS. It addresses the verification of the operational environment in ISS and the functionality of the hardware. This test is designed to make sure that the onboard computer, communication system, gyroscopes and thrusters are operating nominally. It returns a value, displayed in the GUI, indicating the level of IR noise in the U.S. Laboratory in the ISS. It also checks that the batteries are properly seated. The following questions must be answered affirmatively, either by the crew on ISS or the SPHERES team on the ground, prior to declaring all systems are GO (Gate #3):

- Are all the thrusters actuated sequentially?
- Are the satellites able to stop a tumble that is automatically induced?
- Are the satellites returning a termination code of (1) at the end of the test?

If any of these questions is answered negatively, the SPHERES team communicates to the crew appropriate actions to take. Once the test is performed successfully, all the systems are GO for nominal experimental procedures.

This process was refined during the first four test sessions in the ISS. By the fourth test session, after the resolution of a problem with the flight GUI, the combined success rate for the tests being performed in the ISS reached 82%. The remaining 18%

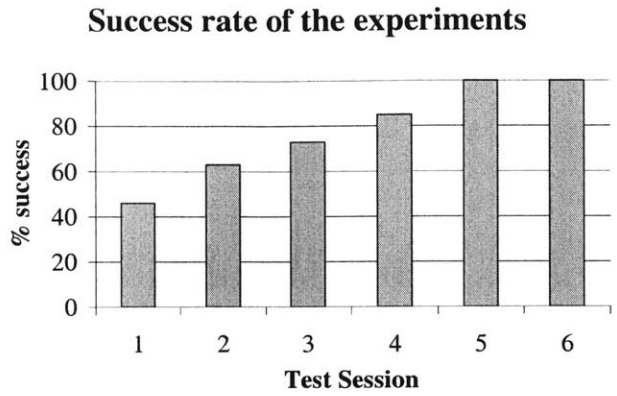


Figure 7-2: Increase in the success rate of the experiments.

exhibited unpredicted multi-path effects, which caused jumps in the state estimates. The success rate reached 100% in the fifth and sixth test session (Fig. 7-2).[3]

Even after performing such a V&V process, there is always a potential risk for failures, but at a level that is judged acceptable for the purpose of the experiments to be performed on SPHERES. Other than hardware failures (including low batteries and low fuel), crew actions and the presence of U/S or IR noise in the environment, some of the remaining risks not addressed by this procedure concern the presence of unmodeled effects like:

- the loss of communication between the satellites for an extended period of time;
- the sudden loss of line-of-sight by a satellite of more than 2 beacons because of body blockage;
- unmodeled estimator biases with spacecraft in close-proximity;
- the presence of multi-path during the convergence phase of the EKF.

These are rather addressed through the implementation of online FDIR.

This section summarized the process for the V&V of flight software prior to uploading it to the ISS. This process was used for the first six test sessions in the ISS. Following the development of the MATLAB® simulation, it was augmented to take

[3]Not included in the calculation of this success rate are the tests invalidated by low batteries, low fuel, crew actions or thruster anomalies.

full advantage of the added verification capabilities through the simulation. The updated process is covered in the following section.

7.3 V&V process using simulation tools

A MATLAB® simulation was developed to support flight algorithm implementation (Chapter 3). While this simulation is written in a different programming language than the one used on SPHERES, it possesses great visualization tools that can provide the user with accurate observations of the simulated results. Following the first ISS test sessions, the SPHERES team realized that this simulation could also be used as a verification tool prior to uploading flight software to the ISS, especially for complex 3-D experiments, like autonomous docking to a tumbling satellite, which are difficult to test in a 2-D environment.

A process was developed (Fig. 7-3) to take full advantage of the availability of a 3-D simulation tool based on the one shown in Fig. 7-1. In Fig. 7-3, the grey boxes correspond to the original V&V process, while the black boxes correspond to the additions made to it using the simulation tool. The left column concerns hardware-in-the-loop tests, while the right column concerns software-in-the-loop tests. For SPHERES, the hardware-in-the-loop testing code is written in C, while the simulation code for software-in-the-loop testing can be written in a different programming language (e.g., MATLAB®).

This augmented process is composed of five main gates. It starts with the user developing and integrating the simulation software using simulation modules and, whenever possible, a GN&C architecture that was already validated from previous experiments. All parameters are set for 3-D simulations in a module that is easily replaceable. The user then proceeds with 3-D simulations of the experiment. Because results from a 3-D dynamic simulation are now available for review, the first gate consists of the user verifying that *all* the requirements related to the experiment are met (Gate #1a). This might require multiple simulations to ensure that all worst case initial conditions are tested.

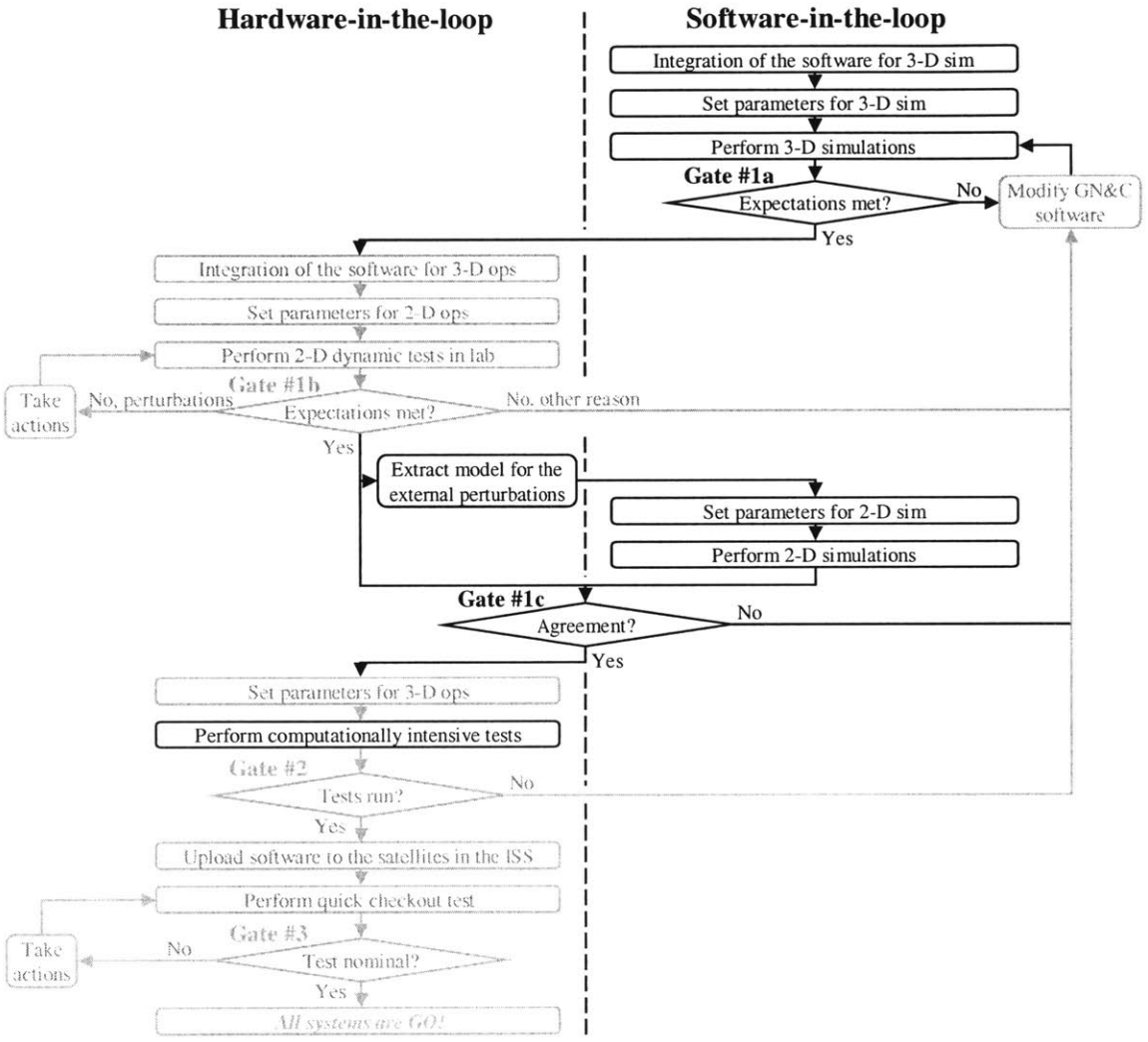


Figure 7-3: Process to verify the software using simulation tools.

After affirmatively answering the same questions as the ones related to the first gate in Section 7.2, the user proceeds with the development and integration of the flight software for hardware-in-the-loop testing. The parameters are set for 2-D operations, and the user initiates 2-D dynamic tests on the MIT SSL 2-D air table. The main objective of the 2-D testing, following successful 3-D simulations, is to provide credibility to the simulation software by validating it with data from dynamic testing. Eventually, for an experiment containing only validated simulation modules and a validated GN&C architecture, 2-D testing at the MIT SSL 2-D air table could be bypassed.

Following the performance of 2-D experiments, the results are analyzed to ensure that all expectations are met and the requirements are fulfilled. Once again, the same questions as in the first gate in Section 7.2 must be answered affirmatively before proceeding (Gate #1b). In case a question cannot be answered affirmatively, the user has to find the likely cause of the problem. If it is found to be external perturbations on the air table, (s)he takes appropriate actions. If it is a problem with the GN&C software, (s)he modifies it and resume simulations. For complex trajectories, involving motions in all six DOF, it might be desirable to also perform 2-D tests in a different plane to further reduce the risk associated with these maneuvers. To do so, a user could rotate the satellite on its air bearing (which requires modifications to the current air bearing). Another solution is to modify the beacon locations in the software, which are used to define the plane corresponding to the surface of the table.

After conclusive results are obtained in the 2-D tests, a model is derived to describe the dynamic motion on the air table, including friction parameters. This model is provided to the simulation along with the 2-D parameters used in the 2-D tests. It is used to represent the dynamic environment to increase the accuracy of the simulated results. The user then simulates the same experiments as the ones in the 2-D tests, with the same initial conditions, and compares the results. The following questions must be answered affirmatively prior to proceeding (Gate #1c):

- Do the trajectories, followed by each satellite during the 2-D tests, correspond to the simulated ones?

- Are the transitions between the different maneuvers occurring in approximately the same amount of time?
- Are the final states of the satellites in both the 2-D tests and the simulations similar?

A negative answer to any of these questions requires the user to modify the GN&C software and possibly add a friction model to the controller for increased performance on the air table (the parameters for 3-D operations would disable this model for ISS experiments).

Once the results from both the 2-D testing and the simulation agree, the parameters for the flight software are set for 3-D operations and the flight software is packaged in its flight configuration. Because enough confidence has been gathered that the flight GUI interfaces correctly with the satellites, there is no need to test every experiment in its final software configuration. However, at least one experiment (the most computationally intensive) from each group of similar experiments should be run. Particular attention should be placed on the proper real-time execution of the most computationally intensive process. The following questions have to be answered affirmatively for each satellite before sending the flight software to NASA and uploading it to the ISS:

- Are the tests initializing?
- Is the state estimator converging to a correct solution?
- Are the thrusters actuating as expected?
- Is telemetry being downloaded and sufficient for debugging purpose and reconstitution of the experiment through simulation?
- Is the most computationally intensive process being executed properly in real time?

Answering negatively to any of these questions is evidence that one process might not get executed properly in real time, requiring the user to modify the GN&C soft-

ware accordingly and resume simulations. Once that gate is successfully passed, the software is uploaded to the ISS and eventually to the satellites. At the start of a test session, the quick checkout test in the ISS occurs as explained in Section 7.2 to complete this process. Following each successful experiment, the corresponding GN&C module and their integration in a GN&C architecture is validated to TRL-6. Moreover, the corresponding simulation modules are also validated to TRL-6.

The main value added by this process is a full dynamic simulation of each experiment, which is used to verify the GN&C software prior to flight testing in the ISS. For complex 3-D trajectories, it is the only way to verify the GN&C software without having to make important compromises such as constraining the trajectory in a 2-D plane. It also reduces the risk associated with unmodeled 3-D perturbations, affecting the sequencing of the GN&C modes, by providing the user with more data to design proper objective functions for each mode.

7.4 Summary

This chapter proposes a role that a testbed like SPHERES can fulfill in the V&V of actual space missions, namely to validate *all software* simulation tools. It also synthesized two processes used for the V&V of flight software for on-orbit microgravity experiments onboard the ISS, which is a contribution as part of this research. The first process presented was used for the first six ISS test sessions, when simulation tools were not yet available. In resulted to a 100% success rate in Test Sessions 05 and 06, spanning a total of 15 experiments performed by seven investigators from MIT (details on these experiments can be found in Appendix E). The second process was developed after a simulation tool for the verification of flight software was made available. It was designed to take full advantage of that simulation tool for the V&V of flight software prior to flight testing, to further reduce the risk of failures.

Chapter 8

Conclusions and recommendations

8.1 Thesis summary

Traditionally, humans have played an active and significant role in coordinating close proximity spacecraft maneuvers such as docking. From the beginning of the space age, the Russians have adopted an approach that involves computers controlling docking. However, the computers are closely monitored by ground operators ready to take control in case of a problem. Such an approach has the advantage of standardizing docking operations, leading to the reduction of the cost associated with planning and operations.

With a vision for future space exploration, involving autonomous robotic assembly of spacecraft as well as robotic servicing of commercial satellites, the development of a capability to perform autonomous docking is needed. Multiple missions, entirely dedicated to autonomous docking and inspection, were flown over the past decade.

This thesis presents a novel approach to the development, verification, demonstration and validation through on-orbit experimentation of GN&C software, enabling autonomous docking to both cooperative and tumbling targets. The approach uses experimentation on the SPHERES testbed inside the ISS. The MIT Space Systems Laboratory created the SPHERES testbed to assist in the development of formation flight and autonomous docking algorithms, and to bring them to a level where they can be confidently used on future space platforms.

An investigation was first performed to find existing GN&C algorithms covering areas relevant to autonomous docking, including estimation, control, path planning, and FDIR (Chapter 2). Based on findings in the literature and on the experience of the author with each of them, a comparison between the algorithms was made to facilitate the selection of the ones to be implemented on the SPHERES testbed.

The implementation process is then covered (Chapter 3). Key subsystems of the SPHERES facility are described in detail. The software interface and simulation tools designed to facilitate the implementation of algorithms are introduced next, including a MATLAB® simulation developed as part of this research. The implementation of GN&C algorithms, selected mainly for their low computation requirement, is discussed in detail, including all the necessary assumptions and adjustments to bring the algorithms from equations to an embedded GN&C module for real-time operation.

The experimental validation of the GN&C modules is presented next (Chapter 4). Major difficulties were encountered with the state estimation algorithms. All were either related to the techniques used in the implementation of the algorithms, or to the sensors providing the navigation measurements. The solution to each of these problems is discussed in detail. Multiple experiments were performed on the MIT SSL 2-D air table and on NASA's KC-135 (reduced gravity research aircraft), resulting in the validation of the GN&C modules on hardware.

The integration of the modules into a GN&C software is then discussed (Chapter 5). Important considerations regarding the selection of the modules and the integration for real-time operation on hardware are provided. A series of ground experiments performed on the MIT SSL 2-D air table and on the NASA MSFC flat floor tested different combinations of algorithms in different docking scenarios. Later, simple formation flight experiments, performed in the ISS, demonstrated coordinated motion between multiple SPHERES satellites on orbit. The outcome of these experiments was a GN&C architecture enabling autonomous docking experimentation using the SPHERES testbed on orbit.

A series of autonomous docking experiments were performed on orbit to validate the GN&C architecture (Chapter 6). They included scenarios like docking to a coop-

erative and an uncooperative target. The experiments presented in Chapter 6 spanned two test sessions in the ISS (Test Session 04 and Test Session 05).[1] Improvements were made after the first session to allow successful docking, even in the presence of global positioning measurement errors. These experiments were concluded with the first autonomous docking to a tumbling target ever achieved in microgravity. The maneuver was successful despite measurement errors, which were detected and rejected online.

Finally, two processes for the V&V of GN&C software were realized (Chapter 7). The first one was used at MIT to verify the GN&C software prior to flight testing for the first six test sessions in the ISS. This process is largely responsible for the success of the experiments onboard the ISS, leading to a 100% success rate during the fifth and sixth test sessions (for a total of 15 experiments). A second process was also established to take advantage of the availability of software simulation tools like the MATLAB® simulation.

8.2 Contributions

The process of developing, verifying, demonstrating and validating a GN&C architecture for autonomous docking presented in this thesis involved the following: the implementation of key GN&C algorithms identified in the literature, their validation in an operational system, the establishment of the GN&C architecture, the integration in a GN&C software and finally the validation of the GN&C software through multiple autonomous docking scenarios. The contributions listed below were the result of these activities.

Developed a GN&C architecture for a nano-satellite, enabling on-orbit autonomous docking and formation flight experiments on a free flying platform in the ISS. Validated the GN&C architecture through

[1] Although the results of Test Session 06 were used in evaluating the V&V process presented in Chapter 7, no detailed analysis of the individual experiments could be achieved on time for this thesis. Nevertheless, preliminary results for Test Session 06 are provided in Table E.6.

11 successful autonomous docking experiments and three successful formation flight experiments in the ISS.

A GN&C architecture was developed as part of this research to enable on-orbit autonomous docking and formation flight experimentation on nano-satellites. The modularity in the architecture proved essential to ensuring flexibility in the use of GN&C modules and the capability to adapt to different scenarios. The architecture was demonstrated on orbit on the SPHERES testbed for both autonomous docking and formation flight experiments. The results from a total of 11 autonomous docking and three formation flight experiments allowed the validation of the GN&C architecture up to TRL-6 (System/subsystem model or prototype demonstration in a relevant environment) for the following increasingly complex scenarios:

- docking to a cooperative target;
- docking to a drifting target;
- docking to a cooperative target using a safe approach trajectory;
- docking to a tumbling target, where the docking axis approximately sweeps a plane at a rate not exceeding 2.25 deg/sec.

Since its implementation, the GN&C architecture has been used by 13 investigators and graduate students from MIT and abroad. Experience has shown that it takes approximately two weeks, for a scientist with experience in the implementation of GN&C systems, to obtain desired results using the proposed architecture on the SPHERES hardware.

Developed a series of flight qualified GN&C modules to populate the architecture. Validated the modules through two experiments in the KC-135 and 23 in the ISS.

A series of GN&C modules for autonomous docking and formation flight were developed as part of this research. Multiple estimation, control, path planning and FD modules were validated up to TRL-6 through the on-orbit experiments presented

in this thesis. These modules are also currently used by other investigators for research on autonomous reconfiguration and path planning.

Created and implemented a process for the verification of GN&C software prior to flight. Validated the process through testing on an experimental free flying platform in the ISS.

A process using laboratory experimentation on a 2-D air table was created and implemented for the verification of GN&C software prior to flight testing on SPHERES. It has been very successful during the preparation of the GN&C software for the first six test sessions in the ISS, leading to a 100% success rate in the experiments performed on the fifth and sixth test sessions, spanning a total of 15 successful experiments performed by seven investigators from MIT. A second version of the process, that takes advantage of newly available simulation tools, was also created.

Created simulation tools to support the implementation of GN&C algorithms for on-orbit formation flight and autonomous docking experiments.

A simulation software written in MATLAB® was created to allow closed-loop dynamic simulation using the same GN&C architecture and modules as the ones implemented on hardware. The simulation was calibrated using sensor and thruster models that were developed through experimentation on hardware over the years. It allows a scientist to obtain six DOF results prior to on-orbit testing. Recently, other investigators used the simulation to develop experiments involving complex 3-D motion.

Two important space firsts were also accomplished as part of this research:

First on-orbit autonomous docking to a tumbling target.

On November 11, 2006, the MIT SSL became the first to accomplish, using the SPHERES testbed in the ISS, an autonomous docking maneuver to a tumbling

target in microgravity. This was the result of an experiment performed for the purpose of this research. The experiment was further complicated by unexpected global positioning measurement errors that were successfully detected by a FD algorithm and rejected online. Moreover, the satellites involved in the experiment were of the same mass, which makes them more prone to perturbations caused by plume impingement as compared with a light satellite docking to a heavy target. This experiment represents a big leap forward, as no other space missions (including Orbital Express) are planning an autonomous docking maneuver to a tumbling target in the near future.

First autonomous formation flight performed inside of the ISS.

On August 19, 2006, the MIT SSL performed the first successful autonomous formation flight to occur using free flying satellites *inside* of the ISS. The three spacecraft formation included the two SPHERES satellites and the ISS.[2] The two satellites followed a trajectory consisting of three waypoints in a coordinate frame attached to the ISS. Centimeter-level precision on the global position of the satellites was maintained throughout the experiment.

8.3 Recommendations for future work

The recommendations for future work focus on five main topics: the recommendations for future autonomous docking experiments, the implementation of algorithms leading to an increase of the level of autonomy, the completion of the C simulation, the development of a realistic docking mechanism, and experiments pushing against the theoretical capabilities of the SPHERES satellites. Each are further described below.

- Regarding future autonomous docking experiments, either the tumbling pattern of the target can be made more complex (by simulating nutation or precession of the axis of rotation) or anomalies of different nature can be simulated. Unless it

[2]Navigation beacons were mounted on the walls of the ISS for this experiment.

is to prove that a computer could allow docking to a spacecraft where a human could not (e.g., the former mission for bringing Skylab back online in the late 1970s [17]), the author recommends following the second path. Safety has always been the primary factor in the planning of docking missions (more than fuel) [94]. Therefore, building confidence in algorithms that allow for greater safety should be a high priority.

- The addition of new GN&C modules made available to guest scientists through the GSP can allow for exciting new opportunities. Most of the algorithms implemented on SPHERES so far are low level GN&C algorithms. By implementing other solver, FDIR and MVM modules, the level of autonomy of the system can be significantly increased as compared to what is currently available. In return, this can allow for a wide range of experiments that could be greatly beneficial to the community.

  - *Implementation of a solver module.* It is the author's opinion that the addition of a LP solver to the library of modules should become a short term priority. Many path planners and event schedulers use LP-based solvers to provide a solution to a constrained optimization problem. Even for complex problems, it is sometime possible to make simplifying assumptions that allow the solution of the problem to be approximated using a LP-based solver [15]. The type of experiments enabled by the addition of a solver module include the ones involving autonomous docking after a random deployment, where the chaser has to compute online a safe and fuel optimal trajectory to dock to the target.

  - *Implementation of FDIR modules.* In this thesis, only one FD module was fully implemented to detect measurement errors from the U/S navigation system. The addition of other FDIR techniques addressing sensor, thruster and software failures, like the non-convergence of the estimator or the solver, can significantly increase the level of autonomy and overall safety of the system.

- *Implementation of MVM modules.* The MVM module used in this thesis for managing GN&C modes is very simple and does not allow for making decisions when multiple choices are available. There exist advanced techniques like the Mission Data System [61] that can allow a satellite to autonomously plan the sequence of activities leading to the completion of the mission. Although the implementation of such a technique can lead to giant leaps in terms of spacecraft autonomy, it is also the one that requires the most hardware and human resources for its completion.

- The current MATLAB® simulation developed as part of this research provides great visualization tools to verify a GN&C software through a six DOF dynamic simulation. However, because SPHERES uses an OS that reads code generated from a C compiler, the software has to be translated after its verification. Past experience has shown that software errors can be introduced during this translation process. Therefore, a six DOF dynamic simulation tool allowing the verification of the flight GN&C software in its native programming language can further increase the level of confidence prior to flight operations.

- Docking experimentation in the ISS has shown that although the two satellites are well aligned at contact, the Velcro docking mechanism did not latch properly. Moreover, it cannot perform a full latching sequence, which typically lasts for a couple of seconds and requires the spacecraft to remain together for that period of time. A new docking mechanism was developed in parallel to this research through the SWARM project [105, 106]. It interfaces with the SPHERES satellites through their expansion port. Its flight implementation can further increase the level of confidence gained through the results collected during ISS operations. This docking mechanism can be augmented with an optical-based navigation sensor [107] that provides six DOF relative state estimates. Such a sensor has the advantage of representing current technologies used in docking missions like the Advanced Video Guidance Sensor (AVGS) [132]. It can also operate even if the thrusters are being activated, as opposed to the current

U/S navigation system that requires the thruster to be turned OFF to avoid interference.

- The experiments performed in the ISS so far were relatively conservative in terms of the sensing and thrusting capabilities of the SPHERES facility. The docking experiments involving a tumbling target were limited to a tumbling rate of 2.25 deg/sec at a separation of 40 cm. Figure 5-3 shows that this is well within the theoretical capabilities of the satellite. It would be interesting to perform aggressive experiments involving circular trajectories to study the GN&C architecture when it is pushed to its limits. Such experiments could bring results that could be useful when planning for emergency situations (e.g., collision avoidance maneuvers).

8.4 Concluding remarks

The main objective of this thesis, which was to develop a GN&C architecture that enables safe and fuel efficient docking with a tumbling target, has been achieved. The architecture has also proven itself in the presence of anomalies or in situations where safety constraints on the trajectory are imposed. The approach used in this thesis to develop, verify, demonstrate and validate a GN&C architecture for autonomous docking has already been used by other investigators from MIT, recently allowing for complex formation flight experiments in the ISS.

Appendix A

Technology Readiness Levels

The Technology Readiness Level (TRL) scale is a metric used by NASA and the DoD for assessing the maturity of new technologies. The content of this appendix is meant to familiarize the reader with this metric. It has been taken as is from a report conducted by NASA in 1995 [78] and another one for Army CECOM in 2002 [50].

A.1 Introduction

TRLs follow a scale from 1 (lowest level of readiness) to 9 (mature development). For example, a technology assessed at TRL-1 is, by definition, at the lowest level of technology readiness. By the time the technology has reached TRL-9, the technology has progressed through formulation of an initial concept for application, proof of concept, demonstration in a laboratory environment and realistic environment, and integration into a system, and has been flight qualified and then flight proven. This last state of development, where the technology is operating under mission conditions, is TRL-9. The importance of achieving TRL-9 is that it is a proof that the system has fulfilled all of its requirements in an actual mission, meaning that the level of confidence for successful nominal operations is at its highest.

A.2 Definitions

This section contains important definitions that are used in the TRL description [50].

Breadboard: Integrated components that provide a representation of a system/subsystem and that can be used to determine concept feasibility and to develop technical data. Typically configured for laboratory use to demonstrate the technical principles of immediate interest. May resemble final system/subsystem in function only.

High fidelity: Addresses form, fit, and function. High-fidelity laboratory environment would involve testing with equipment that can simulate and validate all system specifications within a laboratory setting.

Low fidelity: A representative of the component or system that has limited ability to provide anything but first order information about the end product. Low-fidelity assessments are used to provide trend analysis.

Model: A functional form of a system, generally reduced in scale, near or at operational specification. Models will be sufficiently hardened to allow demonstration of the technical and operational capabilities required of the final system.

Operational environment: Environment that addresses all of the operational requirements and specifications required of the final system, including platform/packaging.

Prototype: A physical or virtual model used to evaluate the technical or manufacturing feasibility or military utility of a particular technology or process, concept, end item, or system.

Relevant environment: Testing environment that simulates the key aspects of the operational environment.

Simulated operational environment: Either (*a*) a real environment that can simulate all of the operational requirements and specifications required of the final system, or (*b*) a simulated environment that allows for testing of a virtual prototype. Used in either case to determine whether a developmental system meets the operational requirements and specifications of the final system.

A.3 TRL detailed description

This section contains the detailed description of each TRL from both the Army and NASA.

Table A.1: TRL description by the Army for software [50].

Technology readiness level	Detailed description
1. Basic principles observed and reported	Lowest level of software readiness. Basic research begins to be translated into applied research and development. Examples might include a concept that can be implemented in software or analytic studies of an algorithm's basic properties.
2. Technology concept and/or application formulated	Invention begins. Once basic principles are observed, practical applications can be invented. Applications are speculative and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies.
3. Analytical and experimental critical function and/or characteristic proof of concept	Active research and development is initiated. This includes analytical studies to produce code that validates analytical predictions of separate software elements of the technology. Examples include software components that are not yet integrated or representative but satisfy an operational need. Algorithms run on a surrogate processor in a laboratory environment.
4. Component and/or breadboard validation in laboratory environment	Basic software components are integrated to establish that they will work together. They are relatively primitive with regard to efficiency and reliability compared to the eventual system. System software architecture development initiated to include interoperability, reliability, maintainability, extensibility, scalability, and security issues. Software integrated with simulated current/legacy elements as appropriate.
5. Component and/or breadboard validation in relevant environment	Reliability of software ensemble increases significantly. The basic software components are integrated with reasonably realistic supporting elements so that it can be tested in a simulated environment. Examples include "high fidelity" laboratory integration of software components. System software architecture established. Algorithms run on a processor(s) with characteristics expected in the operational environment. Software releases are "Alpha" versions and configuration control is initiated. Verification, Validation, and Accreditation (VV&A) initiated.
6. System/subsystem model or prototype demonstration in a relevant environment	Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in software demonstrated readiness. Examples include testing a prototype in a live/virtual experiment or in a simulated operational environment. Algorithms run on processor of the operational environment are integrated with actual external entities. Software releases are "Beta" versions and configuration controlled. Software support structure is in development. VV&A is in process.
7. System prototype demonstration in an operational environment	Represents a major step up from TRL 6, requiring the demonstration of an actual system prototype in an operational environment, such as in a command post or air/ground vehicle. Algorithms run on processor of the operational environment are integrated with actual external entities. Software support structure is in place. Software releases are in distinct versions. Frequency and severity of software deficiency reports do not significantly degrade functionality or performance. VV&A completed.
8. Actual system completed and qualified through test and demonstration	Software has been demonstrated to work in its final form and under expected conditions. In most cases, this TRL represents the end of system development. Examples include test and evaluation of the software in its intended system to determine if it meets design specifications. Software releases are production versions and configuration controlled, in a secure environment. Software deficiencies are rapidly resolved through support infrastructure.
9. Actual system proven through successful mission operations	Actual application of the software in its final form and under mission conditions, such as those encountered in operational test and evaluation. In almost all cases, this is the end of the last "bug fixing" aspects of the system development. Examples include using the system under operational mission conditions. Software releases are production versions and configuration controlled. Frequency and severity of software deficiencies are at a minimum.

Table A.2: TRL description by NASA [78].

Technology readiness level	Detailed description
1. Basic principles observed and reported	This is the lowest "level" of technology maturation. At this level, scientific research begins to be translated into applied research and development. Examples might include studies of basic properties of materials (e.g., tensile strength as a function of temperature for a new fiber).
2. Technology concept and/or application formulated	Once basic physical principles are observed, then at the next level of maturation, practical applications of those characteristics can be 'invented' or identified. For example, following the observation of high critical temperature (Htc) superconductivity, potential applications of the new material for thin film devices (e.g., SIS mixers) and in instrument systems (e.g., telescope sensors) can be defined. At this level, the application is still speculative: there is not experimental proof or detailed analysis to support the conjecture.
3. Analytical and experimental critical function and/or characteristic proof of concept	At this step in the maturation process, active research and development (R&D) is initiated. This must include both analytical studies to set the technology into an appropriate context and laboratory-based studies to physically validate that the analytical predictions are correct. These studies and experiments should constitute "proof-of-concept" validation of the applications/concepts formulated at TRL 2. For example, a concept for High Energy Density Matter (HEDM) propulsion might depend on slush or super-cooled hydrogen as a propellant: TRL 3 might be attained when the concept-enabling phase/temperature/pressure for the fluid was achieved in a laboratory.
4. Component and/or breadboard validation in laboratory environment	Following successful "proof-of-concept" work, basic technological elements must be integrated to establish that the "pieces" will work together to achieve concept-enabling levels of performance for a component and/or breadboard. This validation must devised to support the concept that was formulated earlier, and should also be consistent with the requirements of potential system applications. The validation is relatively "low-fidelity" compared to the eventual system: it could be composed of ad hoc discrete components in a laboratory. For example, a TRL 4 demonstration of a new 'fuzzy logic' approach to avionics might consist of testing the algorithms in a partially computer-based, partially bench-top component (e.g., fiber optic gyros) demonstration in a controls lab using simulated vehicle inputs.
5. Component and/or breadboard validation in relevant environment	At this, the fidelity of the component and/or breadboard being tested has to increase significantly. The basic technological elements must be integrated with reasonably realistic supporting elements so that the total applications (component-level, sub-system level, or system-level) can be tested in a 'simulated' or somewhat realistic environment. From one-to-several new technologies might be involved in the demonstration. For example, a new type of solar photovoltaic material promising higher efficiencies would at this level be used in an actual fabricated solar array 'blanket' that would be integrated with power supplies, supporting structure, etc., and tested in a thermal vacuum chamber with solar simulation capability.
6. System/subsystem model or prototype demonstration in a relevant environment (ground or space)	A major step in the level of fidelity of the technology demonstration follows the completion of TRL 5. At TRL 6, a representative model or prototype system or system — which would go well beyond ad hoc, 'patch-cord' or discrete component level breadboarding — would be tested in a relevant environment. At this level, if the only 'relevant environment' is the environment of space, then the model/prototype must be demonstrated in space. Of course, the demonstration should be successful to represent a true TRL 6. Not all technologies will undergo a TRL 6 demonstration: at this point the maturation step is driven more by assuring management confidence than by R&D requirements. The demonstration might represent an actual system application, or it might only be similar to the planned application, but using the same technologies. At this level, several-to-many new technologies might be integrated into the demonstration. For example, a innovative approach to high temperature/low mass radiators, involving liquid droplets and composite materials, would be demonstrated to TRL 6 by actually flying a working, sub-scale (but scaleable) model of the system on a Space Shuttle or International Space Station 'pallet'. In this example, the reason space is the 'relevant' environment is that microgravity plus vacuum plus thermal environment effects will dictate the success/failure of the system — and the only way to validate the technology is in space.
7. System prototype demonstration in a space environment	TRL 7 is a significant step beyond TRL 6, requiring an actual system prototype demonstration in a space environment. It has not always been implemented in the past. In this case, the prototype should be near or at the scale of the planned operational system and the demonstration must take place in space. The driving purposes for achieving this level of maturity are to assure system engineering and development management confidence (more than for purposes of technology R&D). Therefore, the demonstration must be of a prototype of that application. Not all technologies in all systems will go to this level. TRL 7 would normally only be performed in cases where the technology and/or subsystem application is mission critical and relatively high risk. Example: the Mars Pathfinder Rover is a TRL 7 technology demonstration for future Mars micro-rovers based on that system design. Example: X-vehicles are TRL 7, as are the demonstration projects planned in the New Millennium spacecraft program.
8. Actual system completed and "flight qualified" through test and demonstration (ground or space)	By definition, all technologies being applied in actual systems go through TRL 8. In almost all cases, this level is the end of true 'system development' for most technology elements. Example: this would include DDT&E through Theoretical First Unit (TFU) for a new reusable launch vehicle. This might include integration of new technology into an existing system. Example: loading and testing successfully a new control algorithm into the onboard computer on Hubble Space Telescope while in orbit.
9. Actual system "flight proven" through successful mission operations	By definition, all technologies being applied in actual systems go through TRL 9. In almost all cases, the end of last 'bug fixing' aspects of true 'system development'. For example, small fixes/changes to address problems found following launch (through '30 days' or some related date). This might include integration of new technology into an existing system (such operating a new artificial intelligence tool into operational mission control at JSC). This TRL does not include planned product improvement of ongoing or reusable systems. For example, a new engine for an existing RLV would not start at TRL 9: such 'technology' upgrades would start over at the appropriate level in the TRL system.

Appendix B

Navigation sensor specifications and models

The purpose of this appendix is to provide relevant information about the hardware used by the navigation subsystem on SPHERES. Experience has shown that a good understanding of the capabilities and the limits of the hardware is necessary to design an accurate and robust state estimator. Specifications are given on the gyroscopes as well as the accelerometers. The results of an extensive calibration of the U/S system is also provided. The information provided in this section is general to all the satellites. Information specific to each satellite (like biases and scaling factors) is provided in Appendix D.

B.1 Gyroscope specifications

This section includes relevant information on the gyroscopes. Their general specifications are provided in Figs. B-1 and B-2 (model number QRS14-00050-102).

The gyroscopes output a 0 to +5 Vdc signal. To remove the electrical noise, this signal goes through an anti-aliasing electronic filter (a first order low-pass filter with a cutoff frequency set at 300 Hz). Since this filter is built in the circuit, it cannot be altered or bypass. The signal is then input into a 12 bit analog to digital (A/D) converter and routed through a Field Programmable Gate Array (FPGA). The FPGA

Table B.1: Gyroscope anti-aliasing FIR filter coefficients

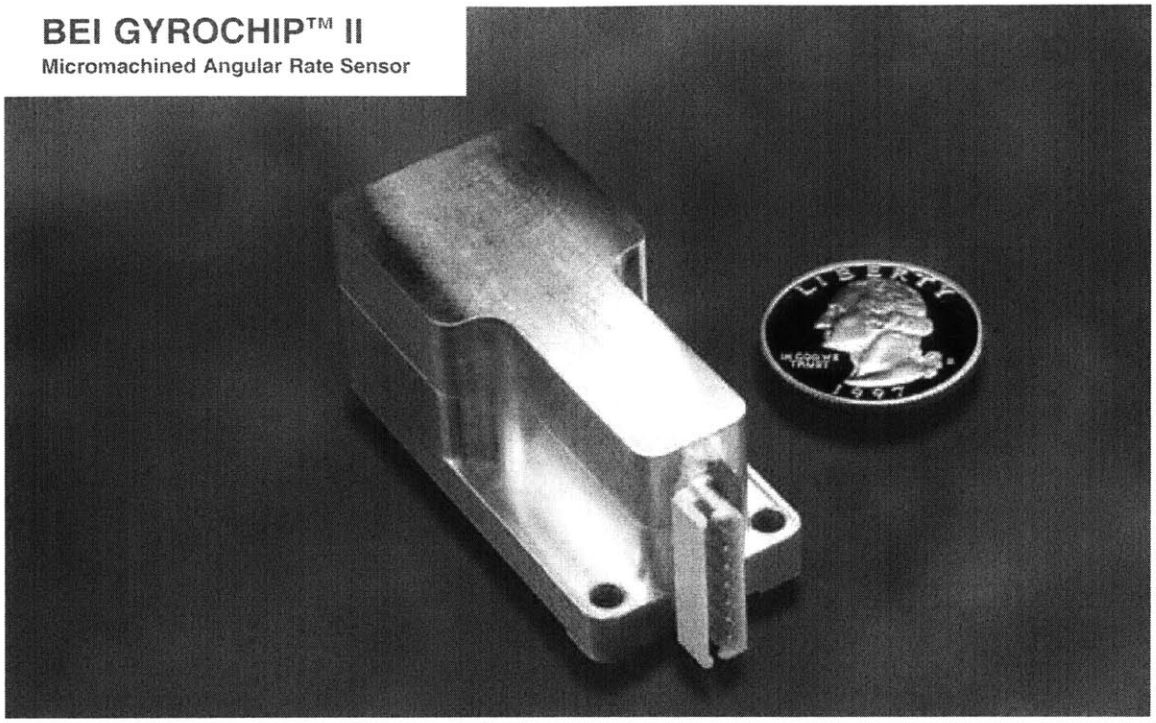
0.0003922365431
0.002028151182
0.003867307911
-0.0001111030724
-0.01577296667
-0.03245829791
-0.01553749759
0.06641919911
0.195257172
0.2959139645
0.2959139645
0.195257172
0.06641919911
-0.01553749759
-0.03245829791
-0.01577296667
-0.0001111030724
0.003867307911
0.002028151182
0.0003922365431

reads the signal and sends the readings at a regular interval to the main computer for processing.

Figure B-3 illustrates the frequency response of the gyroscopes, which was sent directly by the manufacturer. Looking closely at the plots, they appear to have a cutoff frequency at approximately 50 Hz. However, a calibration performed at MIT has revealed an internal resonance at a frequency of 338 Hz as mentioned in this thesis. Figure B-4 shows the result of a Fast Fourier Transform performed on raw gyroscope data collected at 1 kHz with a floating satellite on the MIT SSL 2-D air table. The 338 Hz resonance is clearly visible. Similar results were obtained with an unmounted gyroscope, confirming that the resonance is internal to the gyroscope as opposed to be caused by vibrations in the satellite. This 338 Hz resonance is not affected by the 300 Hz built-in low-pass filter and requires an additional low-pass filter in the software to be attenuated. To attenuate that resonance, a discrete low-pass filter has been implemented with a cutoff frequency set at 50 Hz (finite impulse response filter of order 19 with the coefficients shown in Table B.1). The implementation uses a direct form II transposed filter [91].

BEI GYROCHIP™ II

Micromachined Angular Rate Sensor



Applications

- Platform Stabilization
- Short Term Navigation
- GPS Augmentation
- Camera Stabilization
- Instrumentation
- Robotics
- Autonomous Vehicle Control

Description

The BEI GyroChip II is a compact, rugged, solid-state inertial sensor used to measure angular rotation rates. It features a monolithic quartz sensing element, internal power regulation and DC input/high-level DC output operation. Two versions are available. The +12 Vdc version features a high-level 0 to +5 Vdc output, integral POWER-SAVE mode, and operation from standard battery power. The ±15 Vdc version provides a high-level bipolar output of ±5 Vdc, and is designed for use with conventional double-sided power supplies.

Features

- Solid-State
- Compact, Lightweight Design
- Wide Temperature Range
- High Reliability
- DC Input/High-Level DC Output
- Internal Power Regulation
- POWER SAVE Mode (+12 Vdc Version)

Operation

The BEI GyroChip™ II utilizes a one piece, micromachined, vibrating quartz tuning fork sensing element. Applying the Coriolis effect, a rotational motion about the sensor's input axis produces a DC voltage proportional to the rate of rotation. Use of piezoelectric quartz material simplifies the active element, resulting in exceptional stability over temperature and product life.

BEI GyroChip

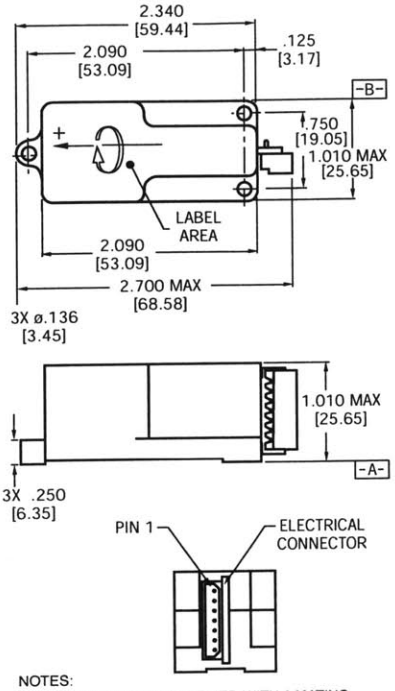
BEI SYSTRON DONNER INERTIAL DIVISION
BEI TECHNOLOGIES, INC.

For applications assistance or more information on any of Systron Donner Inertial Division's micromachined inertial sensors, Call 1-800-227-1625.

Figure B-1: General information on the SPHERES gyroscopes [3].

BEI GYROCHIP™ II

Micromachined Angular Rate Sensor



NOTES:
1. GYROCHIP™ II IS SUPPLIED WITH A MATING CONNECTOR (MOLEX P/N 5264-7 OR EQUIV.).
2. ANGULAR RATE APPLIED AS SHOWN WILL PRODUCE A MORE POSITIVE OUTPUT.
3. UNIT OF MEASURE IN INCHES/[MM].
4. POWER IS DISCONNECTED FROM INTERNAL CIRCUITS APPLYING 5 Vdc ±1 V TO POWER SAVE INPUT.
5. BUILT-IN-TEST ACTIVATED BY GROUNDING PIN 7 CAUSES AN INCREASE IN RATE OUTPUT (PIN 5) OF 0.5 Vdc NOMINAL.
6. BUILT-IN-TEST ACTIVATED BY GROUNDING PIN 7 CAUSES AN INCREASE IN RATE OUTPUT (PIN 5) OF 1.0 Vdc NOMINAL.

QRS14-00XXX-102	
Connector Pin	**Assignment**
1	Power and Signal Ground
2	+Vdc Input
3	POWER SAVE[4]
4	No Connection, Leave Open
5	Rate Output
6	No Connection, Leave Open
7	Built-in-Test[5]

QRS14-00XXX-103	
Connector Pin	**Assignment**
1	-Vdc Input
2	+Vdc Input
3	Power Ground
4	Signal Ground
5	Rate Output
6	No Connection, Leave Open
7	Built-in Test[6]

PARAMETER	**SUMMARY SPECIFICATIONS**	
Part Number	**QRS14-0XXXX-102****	**QRS14-0XXXX-103****
Power Requirements		
Input Voltage	+9 to +18 Vdc	±9 to ±18 Vdc
Input Current	<20 mA	<40 mA (each supply)
Performance		
Standard Ranges	±50, 100, 200, 500, 1000°/sec	
Full Range Output (Nominal)	0 to +5 Vdc	±5 Vdc
Scale Factor Calibration (at 22°C)	±2% of value	
Scale Factor over Temperature (Dev. from 22°C)	≤0.06%/°C	
Bias Calibration (at 22°C)	+2.5 ±0.045 Vdc	0.0 ±0.075 Vdc*
Bias Variation over Temperature (Dev. from 22°C)	<3.0°/sec*	
Short Term Bias Stability (100 sec at const. temp)	≤0.05°/sec	
Long Term Bias Stability (1 year)	≤1.0°/sec	
G Sensitivity	≤0.06°/sec/g	
Start-Up Time	<1.0 sec	
Bandwidth (-90°)	>50 Hz	
Non-Linearity	≤0.05% of F.R.	
Threshold/Resolution	≤0.004°/sec*	
Output Noise (DC to 100Hz)	≤0.05°/sec/√Hz*	≤0.02°/sec/√Hz*
Operating Life	10 years, typical	
Environments		
Operating Temperature	-40°C to +85°C	
Storage Temperature	-55°C to +100°C	
Vibration Operating	4 g_{rms} 20 Hz to 2 kHz random	
Vibration Survival	10 g_{rms} 20 Hz to 2 kHz random	
Shock	200 g	
Weight	≤50 grams	

*Values indicated for ±100°/sec range. **"XXXX" designates ± range.

BEI SYSTRON DONNER INERTIAL DIVISION
BEI TECHNOLOGIES INC.

DIVISION HEADQUARTERS
Systron Donner Inertial Division
2700 Systron Drive, Concord, CA 94518-1399
Tel: 1-925-671-6400 or 1-800-227-1625
Fax: 1-925-671-6590
E-mail: service@systron.com
World Wide Web: http://www.systron.com

EUROPEAN HEADQUARTERS
Systron Donner Inertial Division
Evegate Business Centre, Evegate Park Barn
Smeeth Ashford, Kent, England TN25 6SX
Tel: ++44 (0) 1303 812778
Fax: ++44 (0) 1303 812708
E-mail: systron@easynet.co.uk

DS02-2 9/98

Figure B-2: SPHERES gyroscopes specifications [3].

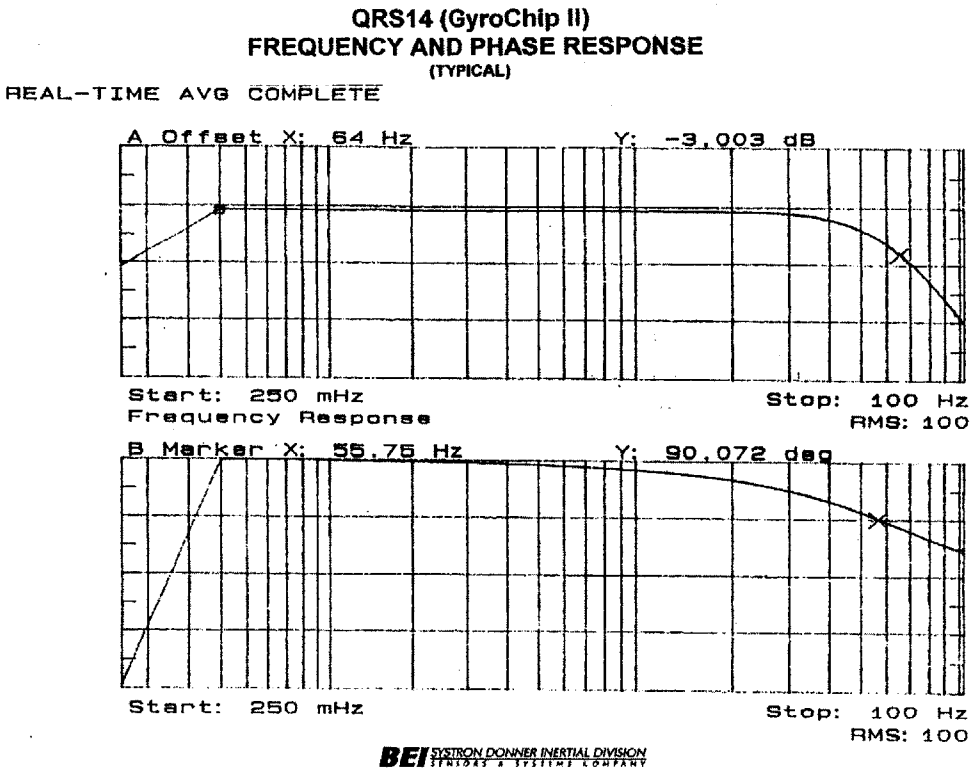


Figure B-3: SPHERES gyroscopes frequency response.

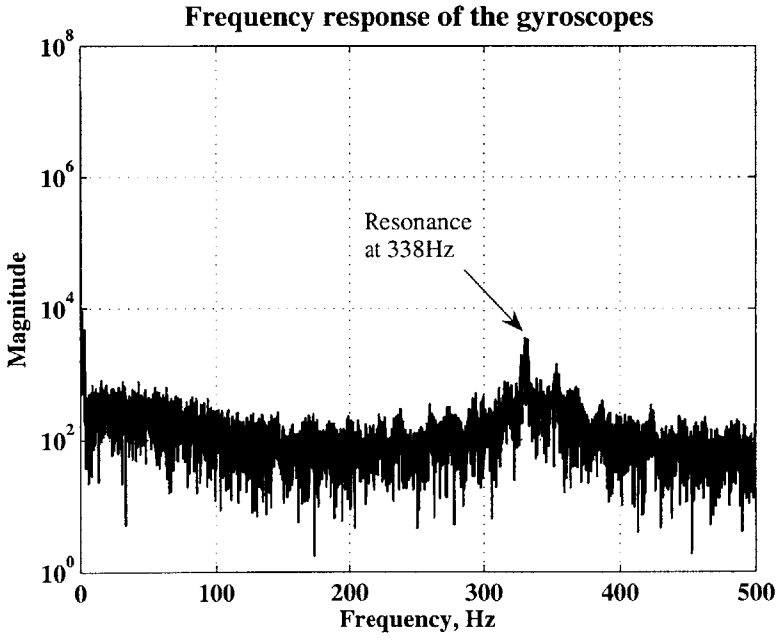


Figure B-4: SPHERES gyroscopes resonance at 338 Hz.

B.2 Accelerometers specifications

General specifications on the accelerometers used on SPHERES are provided in Figs. B-5 and B-6. The frequency response indicated by the manufacturer is also shown in Figs. B-7 and B-8.

The accelerometers are of DC-coupled, meaning that they act as a low pass filter (Fig. B-7). They output a current with an intensity that varies with the acceleration. Because the accelerations induced by the thrusters are much less than the total operational range of the accelerometers, an amplification circuit was built to increase the effective resolution. This circuit has an electronic low-pass filter built-in with a 300 Hz cutoff frequency (anti-aliasing filter for a 1 kHz sampling frequency). It also transforms the current signal to a voltage signal that can be read by an A/D converter. Like for the gyroscopes, the output of the A/D converter is routed through the FPGA, which sends readings to the main computer.

Because the accelerometers are not located at the center of mass of the satellites, the component of the acceleration induced by the angular rate must be subtracted to obtain the linear acceleration in the body frame. The center of acceleration of each accelerometer necessary for this operation is provided in Table B.2.

Table B.2: Center of acceleration for each accelerometer in the body frame.

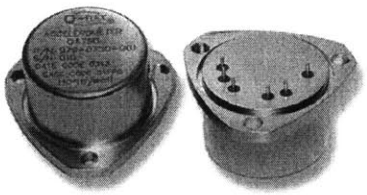
	X-accel.	**Y-accel.**	**Z-accel.**
Center of acc., X-axis (in)	1.681	-1.048	1.237
Center of acc., Y-axis (in)	0.856	0.935*	-1.720
Center of acc., Z-axis (in)	1.421	1.421	1.862

* The Y-accelerometer for satellite S/N 2 is mounted backward. Its center of acceleration along the Y-axis is therefore 1.28 in.

Honeywell

Q-Flex® QA-750 Accelerometer

Cost-effective inertial-grade sensor



For Q-Flex technology in an economical package, Honeywell produces the QA750 for a broad array of moderate performance applications.

As with the entire Q-Flex family of accelerometers, the QA750 features patented Q-Flex® etched-quartz-flexure seismic system. An amorphous quartz proof-mass structure provides excellent bias, scale factor, and axis alignment stability.

The integral electronics develops an acceleration proportional output current providing both static and dynamic acceleration measurements. By use of a customer supplied output load resistor, appropriately scaled for the acceleration range of the application, the output current can be converted into a voltage.

As an option, the QA750 can be provided with the temperature-compensating algorithm where bias, scale factor, and axis misalignment performance are dramatically improved.

Robust design and quality assurance provides superior reliability.

Features

- High value
- Environmentally rugged
- Analog output
- Compact design
- Field-adjustable range
- Built-in test
- Optional thermal compensation

Configuration Drawings

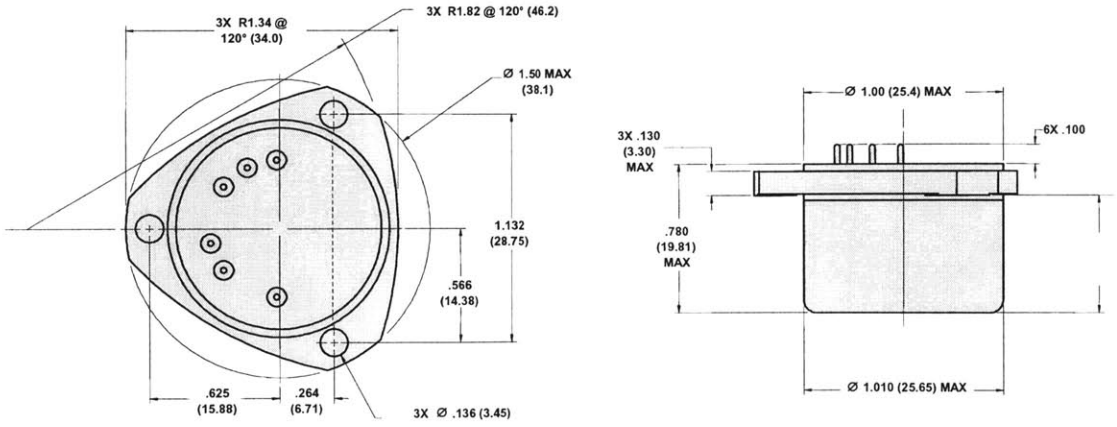


Figure B-5: General information on the SPHERES accelerometers [2].

Performance Characteristics

Additional product specifications, outline drawings and block diagrams, and test data are available on request.

Performance	
Input Range [g]	±30
Bias [mg]	<8
One-year Composite Repeatability [µg]	<1000 (w/o model data)
Temperature Sensitivity [µg/°C]	<60
Scale Factor [mA/g]	1.20 to 1.46
One-year Colmposite Repeatability [ppm]	<1000 (w/o model data)
Temperature Sensitivity [ppm/°C]	<190
Axis Misalignment [µrad]	<7000
One-year Composite Repeatability [µrad]	<300
Vibration Rectification [µg/g^{2}rms]	<60 (50-500 Hz) <200 (500-2000 Hz)
Intrinsic Noise [µg-rms]	<7 (0-10 Hz) <70 (10-500 Hz) <1500 (500-10,000 Hz)
Environment	
Operating Temperature Range [°C]	-55 to +95
Shock [g]	200
Vibration Peak Sine [g]	20 @ 30-500 Hz
Resolution/Threshold [µg]	<1
Bandwidth [Hz]	>300
Thermal Modeling	
	-010 NO -020 YES
Electrical	
Quiescent Current per Supply [mA]	<16
Quiescent Power [mW] @ ±15 VDC	<480
Electrical Interface	Temp Sensor
	Voltage Self Test
	Power / Signal Ground
Input Voltage	±13 to ±18
Physical	
Weight [grams]	52.5 ±4
Diameter below mounting surface [inches]	Ø1.07 ±0.01
Height - bottom to mounting surface [inches]	.600 Max
Case Material	300 Series Stainless Steel

ISO-9001 Certification Since 1995

Find out more:
www.inertialsensor.com

Defense & Space Redmond

Honeywell International, Inc.
MAIL ADDRESS: P.O. Box 97001
15001 N.E. 36th Street
Redmond, Washington 98073-9701
PHONE: 888 206 1667
FAX: 425 883 2104
www.honeywell.com

EXP037, August 2005

Honeywell

Figure B-6: SPHERES accelerometers specifications [2].

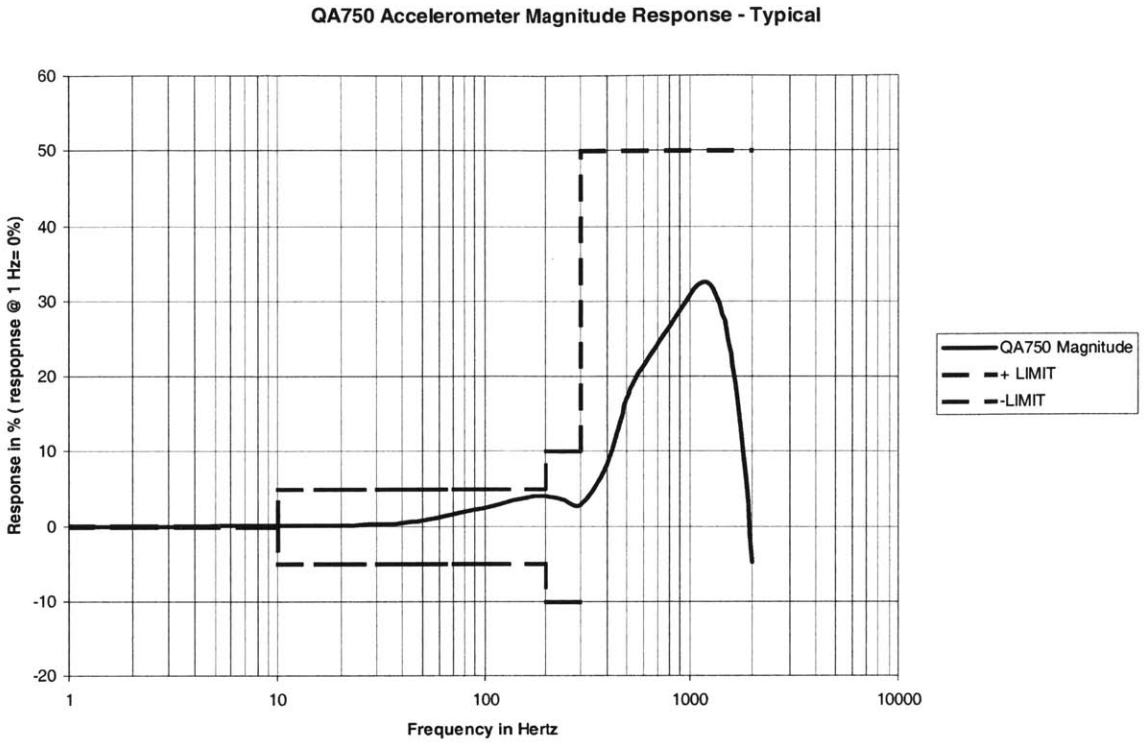


Figure B-7: SPHERES accelerometers frequency response (magnitude).

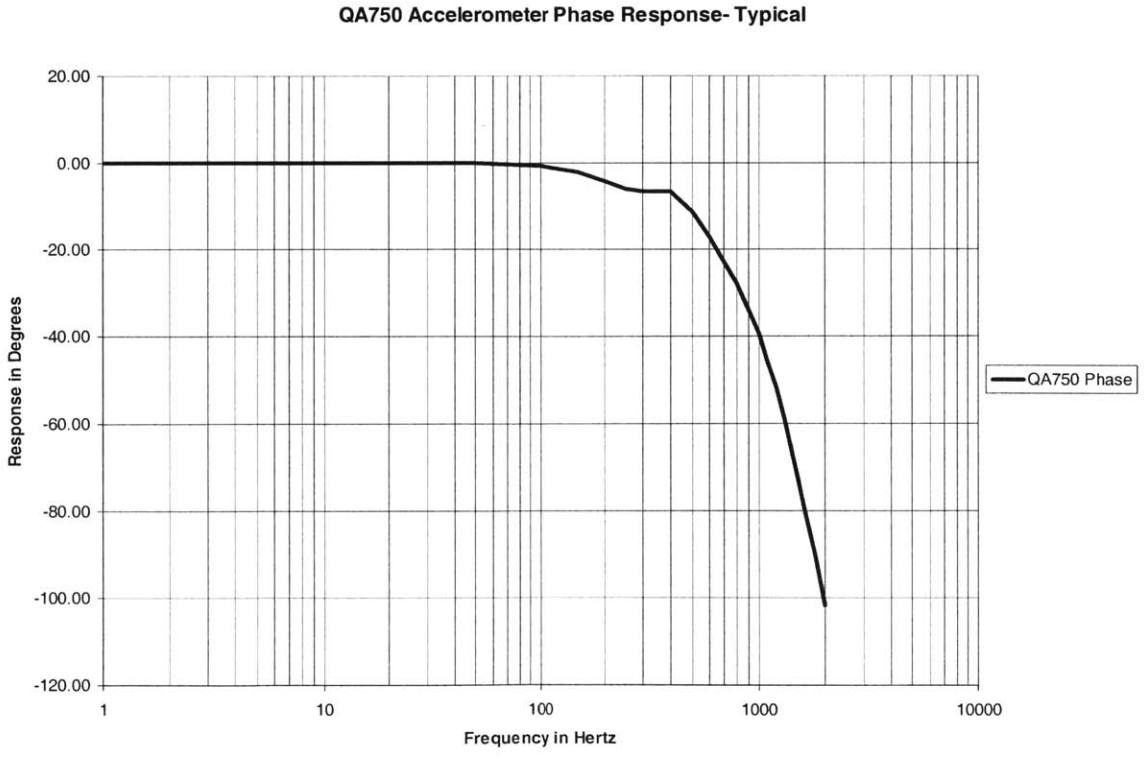


Figure B-8: SPHERES accelerometers frequency response (phase).

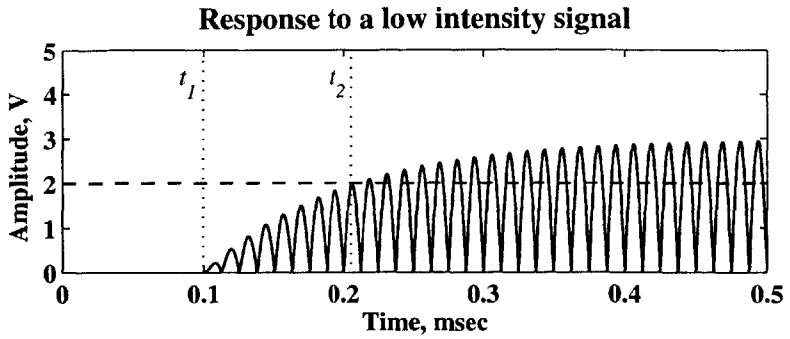


Figure B-9: Response of the U/S receiver to a low intensity sonic wave.

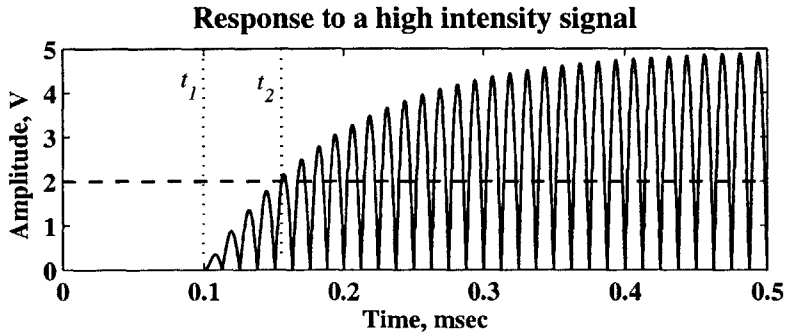
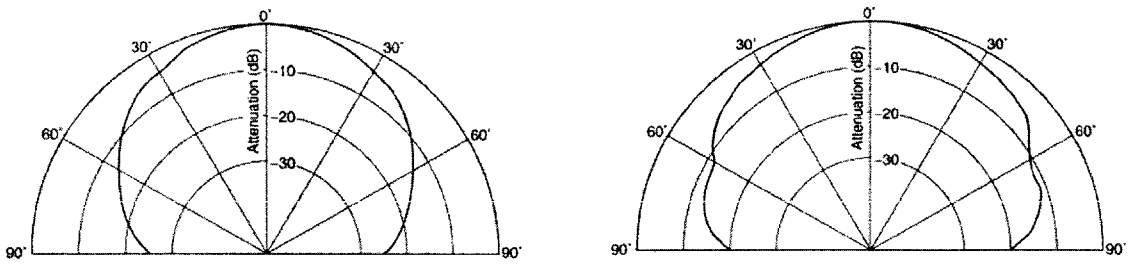


Figure B-10: Response of the U/S receiver to a high intensity sonic wave.

B.3 Ultrasound metrology calibration

The U/S navigation system uses receivers (microphones) composed of a membrane that resonates when excited at a particular frequency (≈40 kHz). The receivers output a voltage signal that fluctuates according to the vibration amplitude of the membrane. Like any resonator, the response to an excitation is not instantaneous. Figures B-9 and B-10 show a typical response (rectified) to a low intensity and a high intensity sonic wave. The reception of the wave is triggered when the voltage of the signal crosses a fixed threshold. By comparing the two plots in Figs. B-9 and B-10, it is observed that although the signal is physically received at time t_1 on both plots, the computer gets the information of the signal only at time t_2, which occurs later for a low intensity wave as compared to a high intensity one.

There are many parameters that affect the intensity of a sonic wave. The traveling distance is an important one (the intensity of the signal diminishes with the square



(a) U/S receiver.

(b) U/S transmitter.

Figure B-11: Gain lobes for the U/S receivers and transmitters.

of the traveling distance). The gain lobes of the transmitters and the receivers also affect the perceived intensity of the signal (Fig. B-11). For example, a receiver angle of 60 degrees causes an attenuation of ≈15 dB. A calibration is necessary to better understand the influence of each parameter and to obtain a good estimate of the actual time the signal reaches the receiver (t_1 in Figs. B-9 and B-10).

A series of experiments was performed at MIT to compare distances measured by an U/S receiver and a high precision linear track activated by a motor-driven threaded shaft. For each experiment, only one of the following parameter was changed:

- the distance between the transmitter and the receiver;
- the receiver angle;
- the transmitter angle;
- the transmitter address (pinging delay after the IR flash);
- the face of the satellite;
- the satellite itself.

The receiver angle is defined as the angle between the vector normal to the receiver and the line-of-sight to the transmitter, while the transmitter angle is the angle between the vector normal to the transmitter and the line-of-sight to the receiver (Fig. B-12). In the list above, only the first three parameters are found to influence the raw measurements. The most important one is the distance between the transmitter

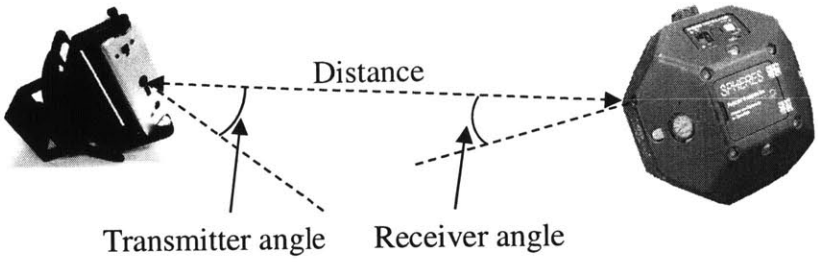


Figure B-12: Varying parameters in the U/S system calibration.

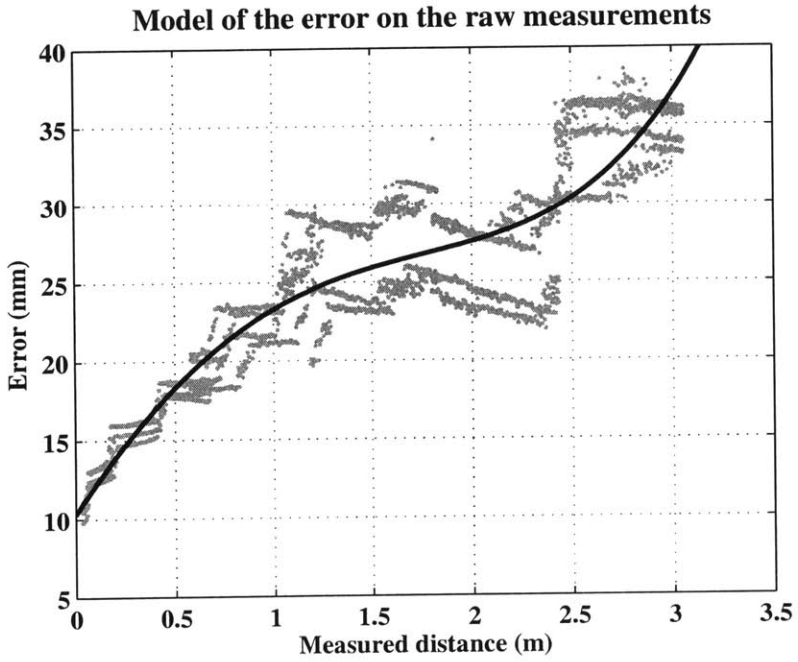


Figure B-13: Measurement errors at different measured distances.

and the receiver. Figure B-13 illustrates the error obtained for measured distances varying between 0 and 3.1 meters. The gray points represent measurement errors from a series of experiments, each using different receiver. A curve is fit through the points to obtain the following measurement error model:

$$z = \varsigma - \left(0.00046\varsigma^4 - 0.0004487\varsigma^3 - 0.006221\varsigma^2 + 0.01919\varsigma + 0.01029\right) \qquad \text{(B.1)}$$

where ς is the measured distance and z is the calibrated distance. This model is accurate to within ± 8 mm throughout the operational range.

Other experiments were performed with only the transmitter angle varying. The

Table B.3: Measurement errors at different transmitter angles.

Transmitter angle (deg)	0-15	15-25	25-35	35+
Measurement error (mm)	+0	+1.5	+5	+11
Uncertainty on the error (mm)	≈0	±1	±1	±2

Table B.4: Measurement errors at different receiver angles.

Receiver angle (deg)	0-15	15-25	25-35	35+
Measurement error (mm)	+0	+0	+4	+7
Uncertainty on the error (mm)	≈0	≈0	±0.5	±1

results are shown in Table B.3. Finally, the influence of the receiver angle is modeled in Table B.4. For both models, the measurement error becomes significant only when the angle exceeds 25 degrees.

All three models presented in this section are currently used to correct the raw measurements before they are passed to the EKF for processing. Because they were obtained by varying only one parameter at a time, there is a chance that combining the corrections from multiple models does not provide an accurate representation of the measurement error. However, although no formal experiment was performed to prove it, it is assumed that the models can be combined when necessary.

Appendix C

Propulsion subsystem theoretical performance

This appendix provides more details on the theoretical performance of the SPHERES CO_2-based propulsion system. The thrust per thruster, the specific impulse, the tank lifetime and the ΔV capability are calculated. The results of a five years long experiment to measure the leak rate in the tank pin valve are also given.

C.1 Theoretical thrust

The theoretical thrust $\mathcal{F}$ can be determined using the following equation [56]:

$$\mathcal{F} = \dot{m} \cdot u_e + (P_e - P_a) A_e \tag{C.1}$$

where $\dot{m}$ is the CO_2 mass flow, u_e is its exit velocity, P_e is its exit pressure, P_a is the atmospheric pressure far from the exit (14.69 psi) and A_e is the average exit area (5.55E-7 m^2, computed from the average exit diameter in Table D.1). To compute P_e, u_e and $\dot{m}$, the state of the flow at the exit is required. Assuming the flow is accelerated in an isoenergetic-isentropic process, it is choked if the following condition is true [47]:

$$\frac{P_a}{P_0} \leq \left(\frac{2}{\alpha + 1}\right)^{\frac{\alpha}{\alpha - 1}} \tag{C.2}$$

where P_0 is the stagnation pressure downstream of the regulator and α is the adiabatic index (1.289 for CO_2). If the velocity of the flow just downstream of the regulator is assumed to be negligible (no dynamic pressure), P_0 is equal to the absolute regulated pressure (static pressure). Equation (C.2) allows to determine a minimum absolute pressure of 26.8 psi required for a choked flow. Consequently, **the minimum regulator pressure that ensures a choked flow is 12.1 psi** and the flow is choked for typical regulator settings of 25, 35 and 55 psi. Also, the propulsion system is designed such that the smallest cross-section area encountered by the flow is at the thruster nozzle exit. Therefore, if the flow is choked, it is choked at the nozzle exit. The exit pressure, P_e, can be determined using the following equation [47]:

$$P_e = P^* = P_0 \left(\frac{2}{\alpha + 1} \right)^{\frac{\alpha}{\alpha - 1}} \tag{C.3}$$

where * means choked flow (Mach 1). For a choked flow, the exit velocity u_e is equal to the speed of sound c [47]:

$$u_e = c = \sqrt{\alpha R T_e} \tag{C.4}$$

where the exit temperature, T_e, is determined using:

$$T_e = T^* = T_0 \left(\frac{2}{\alpha + 1} \right) \tag{C.5}$$

with T_0 being the room temperature (295 K). Finally, the mass flow is computed with:

$$\dot{m} = \varphi_e \cdot c \cdot A_e \tag{C.6}$$

where the density at the exit, φ_e, is calculated using [47]:

$$\varphi_e = \varphi^* = \varphi_0 \left(\frac{2}{\alpha + 1} \right)^{\frac{1}{\alpha - 1}} \tag{C.7}$$

The stagnation density φ_0 is determined using the ideal gas law:

$$\varphi_0 = \frac{P_0}{R T_0} \tag{C.8}$$

where R is the ideal gas constant (188.92 J/(kg.K)). The theoretical thrust is calculated by combining the results from Eqs. (C.1) to (C.8). Table C.1 provides detailed results for typical regulator settings.

Table C.1: Theoretical thrust determination.

Regulator setting (psi)			**25**	**35**	**55**
P_e	(psi),	Eq. (C.3)	21.74	27.22	38.17
T_e	(K),	Eq. (C.5)	257.75	257.75	257.75
u_e	(m/sec),	Eq. (C.4)	250.54	250.54	250.54
φ_0	(kg/m^3),	Eq. (C.8)	4.91	6.15	8.62
φ_e	(kg/m^3),	Eq. (C.7)	3.08	3.85	5.40
$\dot{m}$	(kg/sec),	Eq. (C.6)	4.28E-04	5.36E-04	7.51E-04
$\mathcal{F}$	**(N),**	**Eq. (C.1)**	**0.134**	**0.182**	**0.278**

It is interesting to note that the thrust measured using a load cell is approximately 40% inferior to the thrust calculated using Eq. (C.1). The calculations in this section are based on the simplifying assumption that the CO_2 is accelerated in an isoenergetic-isentropic process, which is an approximation that is not representative of the reality. More details on other possible reasons explaining the difference between the theoretical prediction and the experimental measurements can be found in Ref. [19, 12].

C.2 Specific impulse

The specific impulse ($\mathcal{I}_{sp}$) is a way to describe the efficiency of rocket and jet engines. It represents the impulse (change in momentum) per unit weight of propellant. It is usually expressed in units of seconds. The higher the specific impulse, the less propellant is needed to gain a given amount of momentum. The $\mathcal{I}_{sp}$ is calculated as follows:

$$\mathcal{I}_{sp} = \frac{1}{g}\left(u_e + \frac{(P_e - P_a)}{\dot{m}} \cdot A_e\right) \tag{C.9}$$

where g is the Earth's gravitational constant (9.81 m/sec^2). Table C.2 provides the $\mathcal{I}_{sp}$ of the SPHERES propulsion system, which was calculated using the results shown in Table C.1 for u_e, P_e and $\dot{m}$.

Table C.2: Specific impulse for typical regulator settings.

Regulator setting (psi)			**25**	**35**	**55**
$\mathcal{I}_{sp}$	(sec),	Eq. (C.9)	32.0	34.7	37.7

As expected, the higher the regulated pressure, the more efficient is the propulsion system. The $\mathcal{I}_{sp}$ shown in Table C.2 is comparable to the $\mathcal{I}_{sp}$ of traditional cold gas propulsion systems, which typically ranges between 50 and 70 seconds [116].

C.3 Tank lifetime

The SPHERES propulsion system uses a CO_2 tank with a capacity C of 0.172 kg. Using the mass flow $\dot{m}$ shown in Table C.1, the tank lifetime t_{tank} can be determined as follows:

$$t_{tank} = \frac{C}{\dot{m} \cdot N \cdot D} \tag{C.10}$$

where N is the number of thrusters turned ON and D is the duty cycle. Table C.3 shows the resulting lifetimes. The first case assumes one thruster turned ON with a duty cycle of 100%, while the second case assumes four thrusters turned ON with a duty cycle of 5%, which is representative of typical ISS experiments.

Table C.3: Tank lifetime for typical regulator settings.

Regulator setting (psi)			**25**	**35**	**55**
t_{tank}, N=1 and D=100%	(sec),	Eq. (C.10)	403	322	229
t_{tank}, N=4 and D=5%	(sec),	Eq. (C.10)	2014	1608	1147

With a typical regulator pressure of 25 psi used for experiments in the ISS, Table C.3 supports the approximation of one tank per satellite per half hour of test operation.

C.4 ΔV capability

The ΔV capability is a measure of the total change in velocity that can be performed using the onboard fuel. It is analog to the total range for airplanes or cars. The higher

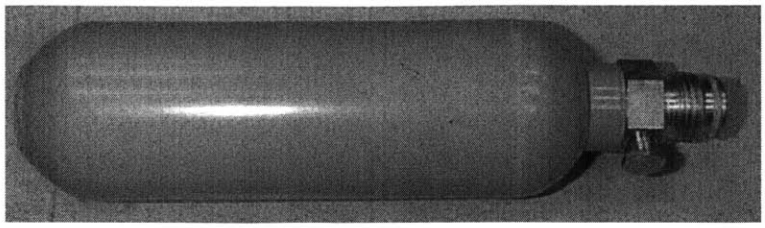


Figure C-1: The SPHERES CO_2 tank assembly (tank and pin valve).

it is, the more trajectory corrections can be applied over the satellite's lifetime. It is calculated as follows:

$$\Delta V = \frac{\mathcal{F}}{m} \cdot t_{tank} \tag{C.11}$$

where m is the mass of the satellite and t_{tank} is the tank lifetime (assuming one thruster opened and a duty cycle of 100%). The results for one tank and for the total number of tanks sent to the ISS (96) are shown in Table C.4.

Table C.4: ΔV capability for typical regulator settings.

Regulator setting (psi)		**25**	**35**	**55**
ΔV for 1 tank	(m/sec), Eq. (C.11)	12.6	13.6	14.8
ΔV for 96 tanks	(m/sec)	1207	1308	1424

As expected, the ΔV increases with the regulated pressure, like the $\mathcal{I}_{sp}$.

C.5 Tank leak rate

An experiment was initiated in 2002 to measure the leak rate of the SPHERES CO_2 tank assembly. Three tanks, using three different pin valve brands, were filled and put aside. The leakage was measured by weighting the tanks on a precision scale.

Figure C-2 shows the leakage over time for all three pin valves. The brand used with the ISS tanks is CrossFire, which shows a leak rate of approximately 5.2 mg/day.

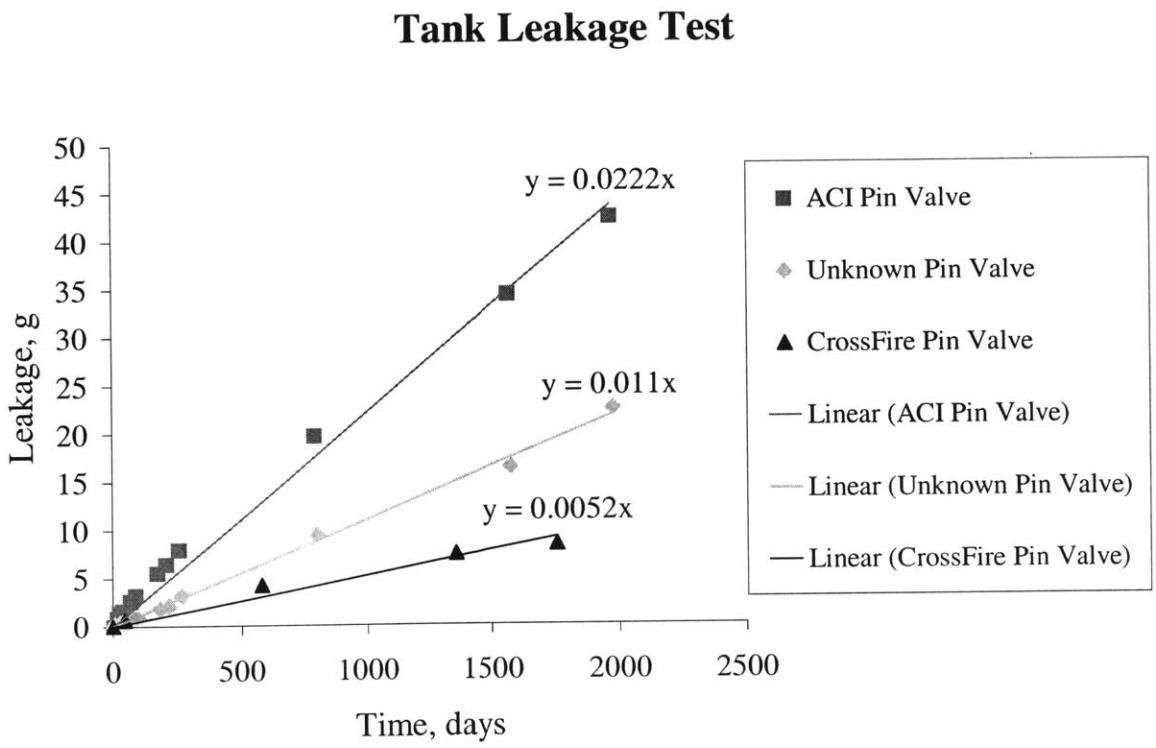


Figure C-2: The tank leakage for three different pin valves.

Appendix D

SPHERES fact sheets

The purpose of this appendix is to provide information specific to each satellite. This information is relevant to the implementation of GN&C algorithms on the SPHERES hardware. Most of it was collected through calibrations and experiments at the MIT SSL and in the KC-135. It also represents what is used by the MIT SPHERES team as of April 2007. It does not include the results of the mass property identification tests performed by Dr. Edward Wilson from Intellization during the first eight ISS test sessions. For each satellite, the INS biases and scaling factors that are currently used are bold.

Table D.1: Satellite S/N 1 fact sheet.

Spheres SN 1 Fact Sheet

Last Revision Date:	**16-Sep-05**					
Item	**Units**					
SPHERES UNIT		*Dry*				
Mass	kg					
Source						
CM offset from GC (Wet)	mm	0.82	-0.99	1.07		
CM offset from GC (Dry)	mm	0.37	-1.52	3.32		
Source			FU #1 ID Excel/03Feb03/Dustin			
CM offset from GC (Wet)	mm	0.97	-2.68	0.55		
Source			KC2003/28Feb03/Dustin			
		Relative to CM		*Relative to CM*		
Inertia - Relative to CM		*Wet*	*Dry*	*Wet*	*Dry*	
I_{xx}	kg m^{-2}	2.58E-02	2.45E-02	2.84E-02	n/a	
I_{yy}	kg m^{-2}	2.25E-02	2.15E-02	2.68E-02	n/a	
I_{zz}	kg m^{-2}	2.03E-02	2.03E-02	2.30E-02	n/a	
I_{xy}	kg m^{-2}	n/a	n/a	-8.37E-05	n/a	
I_{xz}	kg m^{-2}	n/a	n/a	1.40E-05	n/a	
I_{yz}	kg m^{-2}	n/a	n/a	-2.90E-04	n/a	
Source		FU #1 ID Excel/03Feb03/Dustin		KC2003/28Feb03/Dustin		
Sensors						
IMU - Accelerometers	*SN*	*Specified Bias (mg)*	*Ampifier Gain (V/amp)*	*Resolution (mA/g)*		
Accel_X	209	-4.35	80000	**1.30**		
Accel_Y	205	-4.10	80000	**1.31**		
Accel_Z	212	-4.03	80000	**1.30**		
Source		Honeywell QA-750		Honeywell QA-750		
Theoretical Specification		< 8		1.2 to 1.46		
IMU - Gyroscopes	*SN*	*Alignment*	*Specified Bias (mV)*	*Meas. Spec Res (mV/deg/s)*	*Measured Bias*	*Spec Res (mV/deg/s)*
Gyro X	33684	Negative	2538	29.94	**Cnts: 2026.6**	**30.1**
Gyro Y	33685	Positive	2509	30.04	**Cnts: 2043.6**	**30.1**
Gyro Z	33686	Positive	2503	30.46	**Cnts: 2054.5**	**30.5**
Source			Systron Data Sheet	Lab Test/16 Sept 05/Edmund	Lab Test/27 March 06/Simon	Systron Data Sheet
Theoretical Specs			2500 ± 45			30 ± 0.6
Thrusters				Average Thrust Direction - unit vector components		
Thruster Properties	*SN*	*Exit Diameter (mm)*	Thrust Strength (N)	*X*	*Y*	*Z*
Thruster 1: +Z face, -X direction		0.844	0.1313	0.998	-0.022	0.025
Thruster 2: -Z face, -X direction		0.844	0.1302	0.999	-0.001	-0.025
Thruster 3: +X face, -Y direction		0.840	0.1338	0.059	0.996	0.032
Thruster 4: -X face, -Y direction		0.840	0.1339	-0.018	0.999	0.004
Thruster 5: +Y face, -Z direction		0.838	0.1263	0.011	0.034	0.999
Thruster 6: -Y face, -Z direction		0.838	0.1318	-0.022	-0.012	0.999
Thruster 7: +Z face, +X direction		0.844	0.1340	-0.999	-0.008	0.044
Thruster 8: - Z face, +X direction		0.844	0.1321	-0.998	-0.009	-0.037
Thruster 9: +X face, +Y direction		0.840	0.1320	0.042	-0.999	0.009
Thruster 10: -X face, +Y direction		0.840	0.1340	-0.019	-1.000	0.004
Thruster 11: +Y face, +Z direction		0.838	0.1241	0.004	0.060	-0.998
Thruster 12: -Y face, +Z direction		0.838	0.1244	-0.002	-0.030	-0.999
Source		Thruster Groups/21Nov02/Dustin	FU #1 ID Excel/03Feb03/Dustin			

Table D.2: Satellite S/N 2 fact sheet.

Spheres SN 2 Fact Sheet

Last Revision Date:	**16-Sep-05**					
Item	**Units**					
SPHERES UNIT		*Dry*				
Mass	kg					
Source						
CM offset from GC (Wet)	mm					
CM offset from GC (Dry)	mm					
Source						
CM offset from GC (Wet)	mm					
Source						
		Relative to CM				
Inertia - Relative to CM		*Wet*	*Dry*			
I_{xx}	kg m^{-2}					
I_{yy}	kg m^{-2}					
I_{zz}	kg m^{-2}					
I_{xy}	kg m^{-2}					
I_{xz}	kg m^{-2}					
I_{yz}	kg m^{-2}					
Source						
Sensors						
IMU - Accelerometers	*SN*	*Specified Bias (mg)*	*Ampifier Gain (V/amp)*	*Resolution (mA/g)*		
Accel_X	202	-2.81	80000	**1.29**		
Accel_Y	204	-4.84	80000	**1.31**		
Accel_Z	210	-5.31	80000	**1.30**		
Source		Honeywell QA-750		Honeywell QA-750		
Theoretical Specification		< 8		1.2 to 1.46		
IMU - Gyroscopes	*SN*	*Alignment*	*Specified Bias (mV)*	*Meas. Spec Res (mV/deg/s)*	*Measured Bias (mV)*	*Spec Res (mV/deg/s)*
Gyro X	33244	Negative	2535	29.63	**2475.7 (Cnts: 2027.6)**	**30**
Gyro Y	33242	Positive	2527	29.74	**2510.0 (Cnts:2055.7)**	**30.1**
Gyro Z	33240	Positive	2512	29.76	**2500.2 (Cnts: 2047.7)**	**30.1**
Source			Systron Data Sheet	Lab Test/16 Sept 05/Edmund	Lab Test/07July03/Edmund	Systron Data Sheet
Theoretical Specs			2500 ± 45			30 ± 0.6
Thrusters				Average Thrust Direction - unit vector components		
Thruster Properties	*SN*	*Exit Diameter (mm)*	Thrust Strength (N)	*X*	*Y*	*Z*
Thruster 1: +Z face, -X direction						
Thruster 2: -Z face, -X direction						
Thruster 3: +X face, -Y direction						
Thruster 4: -X face, -Y direction						
Thruster 5: +Y face, -Z direction						
Thruster 6: -Y face, -Z direction						
Thruster 7: +Z face, +X direction						
Thruster 8: - Z face, +X direction						
Thruster 9: +X face, +Y direction						
Thruster 10: -X face, +Y direction						
Thruster 11: +Y face, +Z direction						
Thruster 12: -Y face, +Z direction						
Source						

Table D.3: Satellite S/N 3 fact sheet.

Spheres SN 3 Fact Sheet

Last Revision Date:	**26-Aug-05**					
Item	**Units**					
SPHERES UNIT		*Dry*				
Mass	kg					
Source						
CM offset from GC (Wet)	mm					
CM offset from GC (Dry)	mm					
Source						
CM offset from GC (Wet)	mm					
Source						
		Relative to CM				
Inertia - Relative to CM		*Wet*	*Dry*			
I_{xx}	$kg\ m^{-2}$					
I_{yy}	$kg\ m^{-2}$					
I_{zz}	$kg\ m^{-2}$					
I_{xy}	$kg\ m^{-2}$					
I_{xz}	$kg\ m^{-2}$					
I_{yz}	$kg\ m^{-2}$					
Source						
Sensors						
IMU - Accelerometers	*SN*	*Specified Bias (mg)*	*Ampifier Gain (V/amp)*	*Resolution (mA/g)*		
Accel_X	203	-4.71	80000	**1.30**		
Accel_Y	206	-4.78	80000	**1.31**		
Accel_Z	208	-3.47	80000	**1.31**		
Source		Honeywell QA-750		Honeywell QA-750		
Theoretical Specification		< 8		1.2 to 1.46		
IMU - Gyroscopes	*SN*	*Alignment*	*Specified Bias (mV)*	*Measured Bias (mV)*	*Spec Res (mV/deg/s)*	
Gyro X	33688	Negative	2508	**2508.7 (Cnts: 2054.6)**	**30.1**	
Gyro Y	33689	Positive	2516	**2497.9 (Cnts: 2047.8)**	**30.1**	
Gyro Z	33687	Positive	2506	**2483.4 (Cnts: 2033.9)**	**30.1**	
Source			Systron Data Sheet	Lab Test/24July03/Edmund	Systron Data Sheet	
Theoretical Specs			2500 ± 45		30 ± 0.6	
Thrusters				Average Thrust Direction - unit vector components		
Thruster Properties	*SN*	*Exit Diameter (mm)*	Thrust Strength (N)	*X*	*Y*	*Z*
Thruster 1: +Z face, -X direction						
Thruster 2: -Z face, -X direction						
Thruster 3: +X face, -Y direction						
Thruster 4: -X face, -Y direction						
Thruster 5: +Y face, -Z direction						
Thruster 6: -Y face, -Z direction						
Thruster 7: +Z face, +X direction						
Thruster 8: - Z face, +X direction						
Thruster 9: +X face, +Y direction						
Thruster 10: -X face, +Y direction						
Thruster 11: +Y face, +Z direction						
Thruster 12: -Y face, +Z direction						
Source						

Table D.4: Satellite S/N 4 fact sheet.

Spheres SN 4 Fact Sheet

Last Revision Date:	**16-Sep-05**					
Item	**Units**					
SPHERES UNIT		*Dry*				
Mass	kg					
Source						
CM offset from GC (Wet)	mm					
CM offset from GC (Dry)	mm					
Source						
CM offset from GC (Wet)	mm					
Source						
		Relative to CM		*Relative to CM*		
Inertia - Relative to CM		*Wet*	*Dry*	*Wet*		*Dry*
I_{xx}	kg m^2					
I_{yy}	kg m^2					
I_{zz}	kg m^2					
I_{xy}	kg m^2					
I_{xz}	kg m^2					
I_{yz}	kg m^2					
Source						
Sensors						
IMU - Accelerometers	*SN*	*Specified Bias (mg)*	*Amplifier Gain (V/amp)*	*Resolution (mA/g)*		
Accel_X	235	-2.60	80000	**1.31**		
Accel_Y	234	-0.68	80000	**1.30**		
Accel_Z	233	-1.87	80000	**1.30**		
Source		Honeywell QA-750		Honeywell QA-750		
Theoretical Specification		< 8		1.2 to 1.46		
IMU - Gyroscopes	*SN*	*Alignment*	*Specified Bias (mV)*	*Meas. Spec Res (mV/deg/s)*	*Measured Bias (mV)*	*Spec Res (mV/deg/s)*
Gyro X	33690	Negative	2510	28.89	**2502.2 (Cnts: 2049.3)**	**30.1**
Gyro Y	35759	Positive	2511	29.7	**2494.6 (Cnts: 2043.1)**	**30.1**
Gyro Z	35760	Positive	2521	29.65	**2502.0 (Cnts: 2049.1)**	**30.1**
Source			Systron Data Sheet	Lab Test/16 Sept 05/Edmund	Lab Test/24July03/Edmund	Systron Data Sheet
Theoretical Specs			2500 ± 45			30 ± 0.6
Thrusters				Average Thrust Direction - unit vector components		
Thruster Properties	*SN*	*Exit Diameter (mm)*	Thrust Strength (N)	*X*	*Y*	*Z*
Thruster 1: +Z face, -X direction						
Thruster 2: -Z face, -X direction						
Thruster 3: +X face, -Y direction						
Thruster 4: -X face, -Y direction						
Thruster 5: +Y face, -Z direction						
Thruster 6: -Y face, -Z direction						
Thruster 7: +Z face, +X direction						
Thruster 8: - Z face, +X direction						
Thruster 9: +X face, +Y direction						
Thruster 10: -X face, +Y direction						
Thruster 11: +Y face, +Z direction						
Thruster 12: -Y face, +Z direction						
Source						

Table D.5: Satellite S/N 5 fact sheet.

Spheres SN 5 Fact Sheet

Last Revision Date:	**16-Sep-05**					
Item	Units					
SPHERES UNIT		*Dry*				
Mass	**kg**					
Source						
CM offset from GC (Wet)	mm					
CM offset from GC (Dry)	mm	-0.02	-0.82	3.08		
Source		AMES JANNAF Paper (Now 2003 KC Test)				
CM offset from GC (Wet)	mm					
Source						
		Relative to CM		*Relative to CM*		
Inertia - Relative to CM		*Wet*	*Dry*	*Wet*	*Dry*	
I_{xx}	kg m^{-2}				2.20E-02	
I_{yy}	kg m^{-2}				1.97E-02	
I_{zz}	kg m^{-2}				1.82E-02	
I_{xy}	kg m^{-2}				1.96E-04	
I_{xz}	kg m^{-2}				-5.50E-05	
I_{yz}	kg m^{-2}				-2.15E-04	
Source				AMES JANNAF Paper (Now 2003 KC Test)		
Sensors						
IMU - Accelerometers	*SN*	*Specified Bias (mg)*	*Ampifier Gain (V/amp)*	*Resolution (mA/g)*		
Accel_X	207	-4.58	80000	**1.30**		
Accel_Y	201	-4.03	80000	**1.31**		
Accel_Z	250	-0.67	80000	**1.29**		
Source		Honeywell QA-750		Honeywell QA-750		
Theoretical Specification		< 8		1.2 to 1.46		
IMU - Gyroscopes	*SN*	*Alignment*	*Specified Bias (mV)*	*Meas. Spec Res (mV/deg/s)*	*Measured Bias (mV)*	*Spec Res (mV/deg/s)*
Gyro X	33241	Negative	2517	29.69	**2497.8 (Cnts: 2045.9)**	**30**
Gyro Y	33243	Positive	2512	29.76	**2510.3 (Cnts: 2055.9)**	**30.1**
Gyro Z	35761	Positive	2510	29.71	**2492.9 (Cnts: 2041.7)**	**30**
Source			Systron Data Sheet	Lab Test/16 Sept 05/Edmund	Lab Test/07July03/Edmund	Systron Data Sheet
Theoretical Specs			2500 ± 45			30 ± 0.6
Thrusters				Average Thrust Direction - unit vector components		
Thruster Properties	*SN*	*Exit Diameter (mm)*	Thrust Strength (N)	*X*	*Y*	*Z*
Thruster 1: +Z face, -X direction						
Thruster 2: -Z face, -X direction						
Thruster 3: +X face, -Y direction						
Thruster 4: -X face, -Y direction						
Thruster 5: +Y face, -Z direction						
Thruster 6: -Y face, -Z direction						
Thruster 7: +Z face, +X direction						
Thruster 8: - Z face, +X direction						
Thruster 9: +X face, +Y direction						
Thruster 10: -X face, +Y direction						
Thruster 11: +Y face, +Z direction						
Thruster 12: -Y face, +Z direction						
Source						

Table D.6: Satellite S/N 6 fact sheet.

Spheres SN 6 Fact Sheet

Last Revision Date:	**6-Sep-05**					
Item	Units					
SPHERES UNIT		*Dry*				
Mass	kg					
Source						
CM offset from GC (Wet)	mm					
CM offset from GC (Dry)	mm					
Source						
CM offset from GC (Wet)	mm					
Source						
Inertia - Relative to CM						
I_{xx}	kg m^{-2}					
I_{yy}	kg m^{-2}					
I_{zz}	kg m^{-2}					
I_{xy}	kg m^{-2}					
I_{xz}	kg m^{-2}					
I_{yz}	kg m^{-2}					
Source						
Sensors						
IMU - Accelerometers	*SN*	*Specified Bias (mg)*	*Ampifier Gain (V/amp)*	*Resolution (mA/g)*	*Zero g (Counts)*	
Accel_X	1255	0.82	61	**1.32**	2069	
Accel_Y	1256	-0.99	61	**1.31**	2058	
Accel_Z	1254	0.28	61	**1.31**	2037	
Source		Honeywell QA-750		Honeywell QA-750		
Theoretical Specification		< 8		1.2 to 1.46		
IMU - Gyroscopes	*SN*	*Alignment*	*Specified Bias (mV)*	*Meas. Spec Res (mV/deg/s)*	*Measured Bias (mV)*	*Spec Res (mV/deg/s)*
Gyro X	40333	Negative	2517	30.02	**2497.8 (Cnts: 2045.7)**	**30.05**
Gyro Y	40331	Positive	2512	30.1	**2490.3 (Cnts: 2039.6)**	**30.07**
Gyro Z	40332	Positive	2510	30.02	**2509 (Cnts: 2054.9)**	**30.04**
Source			Systron Data Sheet	Lab Test/06Sep05/Edmund	Lab Test/25Aug05/Edmund	Systron Data Sheet
Theoretical Specs			2500 ± 45			30 ± 0.6
Thrusters				Average Thrust Direction - unit vector components		
Thruster Properties	*SN*	*Exit Diameter (mm)*	Thrust Strength (N)	*X*	*Y*	*Z*
Thruster 1: +Z face, -X direction	49					
Thruster 2: -Z face, -X direction	10					
Thruster 3: +X face, -Y direction	45					
Thruster 4: -X face, -Y direction	80					
Thruster 5: +Y face, -Z direction	81					
Thruster 6: -Y face, -Z direction	31					
Thruster 7: +Z face, +X direction	84					
Thruster 8: - Z face, +X direction	86					
Thruster 9: +X face, +Y direction	71					
Thruster 10: -X face, +Y direction	74					
Thruster 11: +Y face, +Z direction	83					
Thruster 12: -Y face, +Z direction	25					
Source						

Table D.7: Master fact sheet, 1 of 3.

Spheres Fact Sheet					SPHERES
Last Revision Date:	**21-Nov-02**				
Item	**Units**		**Value**		**Measured Info (Source/Date/Who)**
SPHERES UNIT		*Wet*		*Dry*	
Mass	kg	4.38		4.21	CAD/19Nov02/Edison
Dimension		X	Y	Z	
Maximum Span	cm	20	20	25	CDR/15Feb02/???
CM offset from GC (Wet)	mm	0.48	-1.19	1.08	CAD/19Nov02/Edison
CM offset from GC (Dry)	mm	0.49	-1.24	3.98	CAD/19Nov02/Edison
Inertia - Relative to CM		*Wet*		*Dry*	
I_{xx}	kg m^{-2}	2.30E-02		2.19E-02	CAD/19Nov02/Edison
I_{yy}	kg m^{-2}	2.42E-02		2.31E-02	CAD/19Nov02/Edison
I_{zz}	kg m^{-2}	2.14E-02		2.13E-02	CAD/19Nov02/Edison
I_{xy}	kg m^{-2}	9.90E-05		9.90E-05	CAD/19Nov02/Edison
I_{xz}	kg m^{-2}	-2.95E-04		-2.95E-04	CAD/19Nov02/Edison
I_{yz}	kg m^{-2}	-2.54E-05		-2.54E-05	CAD/19Nov02/Edison
Inertia - relative to GC		*Wet*		*Dry*	
I_{xx}	kg m^{-2}	2.29E-02		2.18E-02	CAD/19Nov02/Edison
I_{yy}	kg m^{-2}	2.42E-02		2.31E-02	CAD/19Nov02/Edison
I_{zz}	kg m^{-2}	2.14E-02		2.13E-02	CAD/19Nov02/Edison
I_{xy}	kg m^{-2}	9.65E-05		9.64E-05	CAD/19Nov02/Edison
I_{xz}	kg m^{-2}	-2.93E-04		-2.87E-04	CAD/19Nov02/Edison
I_{yz}	kg m^{-2}	-3.11E-05		-4.61E-05	CAD/19Nov02/Edison
Inertia - Principal Axis	kg m^{-2}				
I_{xx}	kg m^{-2}	2.13E-02		2.12E+05	CAD/19Nov02/Edison
I_{yy}	kg m^{-2}	2.30E-02		2.19E+05	CAD/19Nov02/Edison
I_{zz}	kg m^{-2}	2.42E-02		2.31E+05	CAD/19Nov02/Edison
Rotate Angles - GC to Principal Axis					
About x	deg	-91.08		-92.26	CAD/19Nov02/Edison
About y	deg	4.50		4.74	CAD/19Nov02/Edison
About z	deg	-79.72		-64.70	CAD/19Nov02/Edison
Radii of Gyration wrt Principal Axis					
R1	cm	6.98		7.09	CAD/19Nov02/Edison
R2	cm	7.25		7.22	CAD/19Nov02/Edison
R3	cm	7.44		7.41	CAD/19Nov02/Edison
Sensors					
Global Metrology					
Ultrasonic (Receiver)	No.	24			
Frequency	kHz	40			Measured/Jan02/Edmund
IR Receiver	No.	12			
Wavelength	nm	880			Vishay Data Sheet
IR Transmitter	No.	24			
Wavelength	nm	880			Photonic Data Sheet
Max Update Rate	Hz	6.7			Calculated/15Sept02/Edmund
Nominal Update Rate	Hz	1 to 2			CDR/15Feb02/???
Max Range	m	4			CDR/15Feb02/???
Position Accuracy	cm				
Position Variability	cm				
Angular Accuracy	deg				
Angular Variability	deg				
Locations (GC)		*X*	*Y*	*Z*	
US1: +X Face, +Z board, -Y US	cm	10.23	-3.92	3.94	CAD Drawing/05Nov02/Edison
US2: +X Face, +Z board, +Y US	cm	10.23	3.92	3.94	CAD Drawing/05Nov02/Edison
US3: +X Face, -Z board, +Y US	cm	10.23	3.92	-3.94	CAD Drawing/05Nov02/Edison
US1: +X Face, -Z board, -Y US	cm	10.23	-3.92	-3.94	CAD Drawing/05Nov02/Edison
US5: +Y Face, +X board, -Z US	cm	3.94	10.23	-3.92	CAD Drawing/05Nov02/Edison
US6: +Y Face, +X board, +Z US	cm	3.94	10.23	3.92	CAD Drawing/05Nov02/Edison
US7: +Y Face, -X board, +Z US	cm	-3.94	10.23	3.92	CAD Drawing/05Nov02/Edison
US8: +Y Face, -X board, -Z US	cm	-3.94	10.23	-3.92	CAD Drawing/05Nov02/Edison
US9: +Z Face, +Y board, -X US	cm	-3.92	3.94	10.26	CAD Drawing/05Nov02/Edison
US10: +Z Face, +Y board, +X US	cm	3.92	3.94	10.26	CAD Drawing/05Nov02/Edison
US11: +Z Face, -Y board, +X US	cm	3.92	-3.94	10.26	CAD Drawing/05Nov02/Edison
US12: +Z Face, -Y board, -X US	cm	-3.92	-3.94	10.26	CAD Drawing/05Nov02/Edison
US13: -X Face, +Y board, -Z US	cm	-10.23	3.92	-3.94	CAD Drawing/05Nov02/Edison
US14: -X Face, +Y board, +Z US	cm	-10.23	3.92	3.94	CAD Drawing/05Nov02/Edison
US15: -X Face, -Y board, +Z US	cm	-10.23	-3.92	3.94	CAD Drawing/05Nov02/Edison
US16: -X Face, -Y board, -Z US	cm	-10.23	-3.92	-3.94	CAD Drawing/05Nov02/Edison
US17: -Y Face, +Z board, -X US	cm	-3.94	-10.23	3.92	CAD Drawing/05Nov02/Edison
US18: -Y Face, +Z board, +X US	cm	3.94	-10.23	3.92	CAD Drawing/05Nov02/Edison
US19: -Y Face, -Z board, +X US	cm	3.94	-10.23	-3.92	CAD Drawing/05Nov02/Edison
US20: -Y Face, -Z board, -X US	cm	-3.94	-10.23	-3.92	CAD Drawing/05Nov02/Edison
US21: -Z Face, +X board, -Y US	cm	3.92	-3.94	-10.23	CAD Drawing/05Nov02/Edison
US22: -Z Face, +X board, +Y US	cm	3.92	3.94	-10.23	CAD Drawing/05Nov02/Edison
US23: -Z Face, -X board, +Y US	cm	-3.92	3.94	-10.23	CAD Drawing/05Nov02/Edison
US24: -Z Face, -X board, -Y US	cm	-3.92	-3.94	-10.23	CAD Drawing/05Nov02/Edison
OnBoard Beacon Location (GC)	cm	10.7	0	0	CAD Drawing/05Nov02/Edison

Table D.8: Master fact sheet, 2 of 3.

Spheres Fact Sheet					**SPHERES**	
Last Revision Date:	**21-Nov-02**					
Item	**Units**		**Value**		**Measured Info (Source/Date/Who)**	**Theoretical Specification (Source/Date/Who)**
IMU - Accelerometers	No.	3				
Max Update Rate	Hz	1000				Interrupt Update Rate
Accel Range	g	±30	Accelerometer			Honeywell/05Nov02/Simon
Designed Range	mg	±25.6				Conditioning Circuit/05Nov02/Simon
	counts	0-4096				Maxim A/D Spec/05Nov02/Simon
Bias	mg	< 8				Honeywell/05Nov02/Simon
	counts	2048				05Nov02/Simon
Scale Factor	mA/g	1.2 to 1.46				Honeywell/05Nov02/Simon
Resolution	mg/count	12.5				Conditioning Circuit/05Nov02/Simon
Axis Misalignment	mrad	< 7				Honeywell/05Nov02/Simon
Bandwidth	Hz	300				Honeywell/05Nov02/Simon
Low Pass Filter # 1	Hz	300				Conditioning Circuit/05Nov02/Simon
Low Pass Filter # 1	Hz	300				Anti Aliasing RC/05Nov02/Simon
Noise (0 to 10 Hz) - 1 s	mg rms	< 7				Honeywell/05Nov02/Simon
	counts	< 1	excluding electrical noise in circuits			= 7E-3 mg / 0.0125 mg/count
Noise (10 to 500 Hz) - 1 s	mg rms	< 70				Honeywell/05Nov02/Simon
	counts	< 6	excluding electrical noise in circuits			= 70E-3 mg / 0.0125 mg/count
Noise (500 to 10 kHz) - 1 s	mg rms	< 1.5				Honeywell/05Nov02/Simon
	counts	0	damped in low pass filters			Conditioning Circuit/05Nov02/Simon
Locations (CM)		*X*	*Y*	*Z*		
Accel_X	cm	5.19	2.17	3.27	CAD Drawing/05Nov02/Edison	
Accel_Y	cm	-2.66	3.35	3.30	CAD Drawing/05Nov02/Edison	
Accel_Z	cm	3.28	-4.37	3.35	CAD Drawing/05Nov02/Edison	
IMU - Gyroscopes	No.	3				
Max Update Rate	Hz	1000				Interrupt Update Rate
Range	deg/s	±83				= 5000 mV / 30 mV/(deg/s) (Simon/05Nov02)
	counts	0 - 4096				Maxim A/D Spec/05Nov02/Simon
Bias Calibration (22°C)	V	2.5 ± 0.045				Systron/Edmund
	counts	2048 ± 37				= xxx / 5V * 4096 counts (Simon/05Nov02)
Bias Variation over Temperature	deg/s	< 3.0				Systron/Edmund
	counts	< 74			Simon/05Nov02	= 3 deg/s / 0.0407 deg/s/count (Simon/05Nov02)
Short Term Bias Stability (100 secs)	deg/s	< 0.05				Systron/Edmund
	counts	< 2			Simon/05Nov02	= 0.05 deg/s / 0.0407 deg/s/count (Simon/05Nov02)
Long Term Bias Stability (1 Year)	deg/s	< 1.0				Systron/Edmund
Resolution	mV/(deg/s)	30				Systron/05Nov02/Simon
	deg/s/count	0.0407				= 166 deg/s / 4096 counts (Simon/05Nov/02)
Bandwidth (-90°)	Hz	50				Systron/Edmund
Low pass filter	Hz	300				Anti Aliasing RC/05Nov02/Simon
Noise (0 to 100 Hz) - 1 s	deg/s/(Hz)$^{1/2}$	< 0.05				Systron/Edmund
	deg/s rms	< 0.71			Simon/05Nov02	= sqrt(0.05^2 * 200) (Simon/05Nov02)
	counts	< 17			Simon/05Nov02	= 0.71 deg/s / 0.0407 deg/s/count (Simon/05Nov02)
Locations (CM)		*X*	*Y*	*Z*		
Gyro_X	cm	(-2.93 to 2.4)	3.10	6.39	CAD Drawing/05Nov02/Edison	
Gyro_Y	cm	-5.49	(-3.75 to 1.59)	-3.24	CAD Drawing/05Nov02/Edison	
Gyro_Z	cm	-5.49	3.24	(1.08 to 4.26)	CAD Drawing/05Nov02/Edison	
Thrusters	No.	12				
Thrust/Thruster	kgms^{-2}	< 0.2			Lab Tests (Al's Thesis/05Nov02/Simon)	
Thrust variability	kgms^{-2}	0.01			Lab Tests (Al's Thesis/06Nov02/Simon)	
Thruster Locations (GC)	cm	*X*	*Y*	*Z*		
Thruster 1: +Z face, -X direction	cm	-5.16	0.00	9.65	CAD Drawing/05Nov02/Edison	
Thruster 2: -Z face, -X direction	cm	-5.16	0.00	-9.65	CAD Drawing/05Nov02/Edison	
Thruster 3: +X face, -Y direction	cm	9.65	-5.16	0.00	CAD Drawing/05Nov02/Edison	
Thruster 4: -X face, -Y direction	cm	-9.65	-5.16	0.00	CAD Drawing/05Nov02/Edison	
Thruster 5: +Y face, -Z direction	cm	0.00	9.65	-5.16	CAD Drawing/05Nov02/Edison	
Thruster 6: -Y face, -Z direction	cm	0.00	-9.65	-5.16	CAD Drawing/05Nov02/Edison	
Thruster 7: +Z face, +X direction	cm	5.16	0.00	9.65	CAD Drawing/05Nov02/Edison	
Thruster 8: - Z face, +X direction	cm	5.16	0.00	-9.65	CAD Drawing/05Nov02/Edison	
Thruster 9: +X face, +Y direction	cm	9.65	5.16	0.00	CAD Drawing/05Nov02/Edison	
Thruster 10: -X face, +Y direction	cm	-9.65	5.16	0.00	CAD Drawing/05Nov02/Edison	
Thruster 11: +Y face, +Z direction	cm	0.00	9.65	5.16	CAD Drawing/05Nov02/Edison	
Thruster 12: -Y face, +Z direction	cm	0.00	-9.65	5.16	CAD Drawing/05Nov02/Edison	
CO_2 Tank						
Tank Mass (with pin valve)	g	440			Lab Measurement/05Nov02/Simon	
Capacity	g	172				Manufacturer Spec (Simon)
High Pressure	psi	860				Vapor pressure of liquid CO2 at 72 °F
Low Pressure	psi	0 - 55				Regulator limits
Nominal Operating Pressure	psi	35				Group decision

Table D.9: Master fact sheet, 3 of 3.

Spheres Fact Sheet					SPHERES	
Last Revision Date:	**21-Nov-02**					
Item	**Units**	**Value**			**Measured Info (Source/Date/Who)**	**Description**
Processors						
DSP	No.	1			CDR/15Feb02/Alvar	Sundance SMT375: TI TMS320C6701
Speed	Mhz	167			CDR/15Feb02/Alvar	Sundance SMT375
FLOPS (Peak)	G	1			CDR/15Feb02/Alvar	Sundance SMT375
RAM	MB	16			CDR/15Feb02/Alvar	Sundance SMT375
Cache	kB	512			CDR/15Feb02/Alvar	Sundance SMT375
ROM - available	kB	224			Design/11Nov02/Alvar	Sundance SMT375
ComPorts	No.	6			CDR/15Feb02/Alvar	Sundance SMT375
CommPort Rate	Mbps	20			CDR/15Feb02/Alvar	Sundance SMT375
FPGA	No.	1			CDR/15Feb02/???	Xilinx Spartan II FPGA (XC2S200)
Clock Speed	Mhz	25			CDR/15Feb02/???	ECS-3953C-250
6 Channel 12 Bit A/D	No.	1			CDR/15Feb02/???	MAX1294
Communications						
Spacecraft to Spacecraft						
Frequency	Mhz	916.5			RFM/Sept02/Alvar	RFM DR-2000
Data Rate	kbps	18			Design/11Nov02/John	
	packets/s	70			Design/11Nov02/John	
Spacecraft to Laptop						
Frequency	Mhz	868.35			RFM/Sept02/Alvar	RFM DR-2001
Data Rate	kbps	18			Design/11Nov02/John	
	packets/s	70			Design/11Nov02/John	
Communication Scheme		TDMA			Design/11Nov02/John	
Frame Length	ms	150 (Nom)	User Changeable		Design/11Nov02/John	
Data Packets	bytes	32			Design/11Nov02/John	
POWER						
Battery Packs	No.	2			CDR/15Feb02/???	
Batteries per pack	No.	8			CDR/15Feb02/???	AA Batteries
3A Schottky Diodes per pack	No.	1			CDR/15Feb02/???	Charge Prevention
Average Power	W	15			CDR/15Feb02/???	
Stand-by Power	W	13.75			CDR/15Feb02/???	
Maximum Power	W	16.25			CDR/15Feb02/???	
3.3 V Regulator	-	-			CDR/15Feb02/???	Traco TSI3.3S2ROSH
5V Regulator	-	-			CDR/15Feb02/???	Traco TSI5.0S2ROSH
15 V DC-DC	-	-			CDR/15Feb02/???	MAXIM MAX772
-15V DC-DC	-	-			CDR/15Feb02/???	MAXIM MAX776
22 V Supply	-	-			CDR/15Feb02/???	MAXIM MAX668 (Propulsion Board)
EXPANSION PORT						
Global data bus	No.	1	Shared with internal components		Design/11Nov02/Alvar	
Addressable Components	No.	3			Design/11Nov02/Alvar	
Data width	bits	32			Design/11Nov02/Alvar	
Address lines	bits	31			Design/11Nov02/Alvar	
RS232 Serial line - wired	No.	1			Design/11Nov02/Alvar	
Data rate	kbps	115			Design/11Nov02/Alvar	
BEACONS						
Fixed Beacons	No.	5				
Mass	g	113			CDR/15Feb02/???	
Dimensions	cm	3.3 x 10 x 4.3			CDR/15Feb02/???	
Onboard Beacons/Sphere Unit	No.	1				
Timing from IR						
- Beacon No. 1	ms	10			Measured/Jun02/Edmund	
- Beacon No. 2	ms	30	Hardware Programmable	10, 30, 50, 70, 90, 110, 130, 150, 170	Measured/Jun02/Edmund	
- Beacon No. 3	ms	50			Measured/Jun02/Edmund	
- Beacon No. 4	ms	70			Measured/Jun02/Edmund	
- Beacon No. 5	ms	90			Measured/Jun02/Edmund	
- Onboard Beacon #1	ms	110 (Nom - Off)	Software Programmable	10, 30, 50, 70, 90, 110, 130, 150, 170, OFF		
- Onboard Beacon #2	ms	130 (Nom - Off)				
- Onboard Beacon #3	ms	150 (Nom - Off)				
OnBoard Beacon Location (GC)		X	Y	Z		
	cm	10.7	0	0	CAD Drawing/05Nov02/Edison	

Appendix E

ISS experiments overview

This appendix presents an overview of every ISS test session covered by this research (Test Sessions 01 to 06). Each experiment in each test session is assessed. The duration of the experiment is provided, along with the fuel consumption. The crew comments are shown and a brief analysis from video observations and the telemetry is also provided. Since a detailed analysis of the results of Test Session 06 could not be provided on time for this thesis, the results shown in Table E.6 are preliminary.

Table E.1: ISS test session 01 overview.

First SPHERES test session in the ISS								
Program	**Test**	**Description**	**Duration**	**GUI result**	**Result**	**Crew notes**	**Notes**	**% tank**
P101	T01	Quick checkout with IMU data download	06:01	1 = success	success		test proceded as expected	0.9
	T02	Open loop rotations, old mixer	07:34	3 = stopped	success	1. First 2 rotations were less than 180 deg (~150-160) 2. Test was manually stopped after ~3 minutes of "running" on GUI.	stopped by crew during data download, but did get data tank not inserted properly	0.8
	T03	Open loop rotations, new mixer	09:23	1 = success	success	2nd axis appeared to be 20-30 deg less than 180 3rd axis appeared to be 20-30 deg greater than 180	test proceded as expected tank not inserted properly	0.8
	T04	Beacon track attitude PD	07:05	0 = lost comm	success	Satellite performed a series of ~10deg changes in one axis instead of the expected response	although beacon tracking was not achieved, the data collected validated the use of the ultrasonic based navigation system flash corruption (bad gyro scaling factors) led bad rate estimates	2.7
	T06	Close-loop xyz rotation	02:53	0 = lost comm	failed	No jet firings at all during test.	flash corruption (bad gyro scaling factors) led bad rate estimates	0
	T06	Close-loop xyz rotation	05:41	1 = success	failed	No thruster firings. Comm light confirmed green during test.	flash corruption (bad gyro scaling factors) led bad rate estimates	0
	T08	Dock Free Short S#1 PD	04:17	3 = stopped	failed	Beacon was off	no beacon data (beacon off) prevented the ultrasonic navigation system from converging crew inadvertently purged tank before the test	24
	T08	Dock Free Short S#1 PD	03:52	3 = stopped	success	satellite performed plus/minus ~10 deg rotations in Y and drifted in other axis.	although beacon docking was not achieved, the data collected validated the use of the ultrasonic based navigation system flash corruption (bad gyro scaling factors) led bad rate estimates	7.8
	T06	Close-loop xyz rotation	02:48	4 = lost comm	failed	no response from the satellite. Enable, Comm, Bat confirmed green on GUI.	flash corruption (bad gyro scaling factors) led bad rate estimates	0
	T06	Close-loop xyz rotation	02:46	1 = success	failed		flash corruption (bad gyro scaling factors) led bad rate estimates	0
	T02	Open loop rotations, old mixer	05:43	1 = success	success	But first braking pulse was insufficient to stop the rotation	confirmed with data, possible stickiness with that thruster tank was inserted properly	0.8
	T03	Open loop rotations, new mixer	04:12	no results	failed		no crew comment file saved, seems that test never started crew inadvertently purged tank before the test	15.1
	T03	Open loop rotations, new mixer	03:58	0 = lost comm	failed	Satellite performed 1st rotation with a 20-30 overshoot and then initiated the 2nd rotation. Test ended without any other jet firings.	test proceded normally, but lost comm before data download	0.4
	T46	Dock Range only S#1	02:31	1 = success	success	No response from the satellite. Beacon green light flashed during test.	although beacon docking was not achieved, raw data collected help validating the use of the ultrasonic based navigation system flash corruption (bad gyro scaling factors) led bad rate estimates	0
	T01	Quick checkout with IMU data download	03:00	1 = success	success	2nd part was a series of rapid fire jet firings with very little rotation	test proceded as expected	0.8
# tests	15						total gas consumption	54.084
total duration	1:11:44						total gas used in tests	15.0096
avg duration/test	04:47					The battery level was good throughout the test session	total gas lost in purge	39.0744

Table E.2: ISS test session 02 overview.

Second SPHERES test session in the ISS									
Program	**Test**	**Description**	**Duration**	**GUI result**	**Result**	**Crew notes**	**Notes**	**% tank used**	**Low batt**
P101	T84	Flash Memory Test	03:07	no results	failed		no crew feedback file, comm lost too early	0	
	T84	Flash Memory Test	03:04	no results	failed		no crew feedback file, comm lost too early	0	
	T84	Flash Memory Test	00:51	no results	failed		no crew feedback file, comm lost too early	0	
	T06	Close-loop xyz rotation	01:17	no results	failed		no crew feedback file, comm lost too early	0	
	T06	Close-loop xyz rotation	08:19	no results	failed		no crew feedback file, comm lost too early	0	
	T84	Flash Memory Test	09:20	no results	failed		no crew feedback file, comm lost too early	0	
	T84	Flash Memory Test	01:34	0 = lost comm	failed	rtn'd value: 0	test started, but lost comm	0	
	T84	Flash Memory Test	02:08	9 = not enabled	failed	rtn value: 9	sphere lost comm right after being enabled before starting the test	0	
	T06	Close-loop xyz rotation	04:16	9 = not enabled	failed	no response, rtn value: 9	sphere lost comm right after being enabled before starting the test	0	
	T06	Close-loop xyz rotation	02:50	3 = stopped by crew	failed	Enable light went out before "run test" No response. Rtn value: 3	sphere lost comm right after being enabled before starting the test	0	
	T84	Flash Memory Test	01:07	11 = flash problem detected and fixed	success	rtn value: 11	flash problem detected and fixed on the first attempt	0	
	T06	Close-loop xyz rotation	05:23	1 = success	success	1st rotation may have overshot by ~20deg. Appeared to trim in other axis during maneuver as expected.	test proceded as expected	0.7	
	T14	De-Tumble, Track, and Dock	03:22	1 = success	failed	may have gotten out of range	estimator crashed becaused of a bad initialization the data collected allow to find the exact source of the problem	0.3	
	T14	De-Tumble, Track, and Dock	03:20	1 = success	failed	No response after stabilizing. Rtn value: 1 Range ~0.5 m	estimator crashed becaused of a bad initialization the data collected allow to find the exact source of the problem	0.3	
	T16b	Dock Fixed Long S#2 PD	08:00	1 = success	success	approached the satellite, but uncontrolled.	was released too close to beacon (expected long approach) did not dock, but got useful data	0.3	
P112	T01	Failed-on thruster FDI	01:52	9 = not enabled	failed	no jet firings. Rtn value: 9	sphere lost comm right after being enabled before starting the test	0	
(Ames)	T01	Failed-on thruster FDI	01:20	1 = success	success	As expected after reset.	Test proceded as expected	0.1	
	T02	Failed-off thruster FDI	08:42	1 = success	success	several pairs of sequential firings. Rtn value: 1	Test proceded as expected	0.2	
	T03	Multiple thruster FDI	01:58	3 = stopped by crew	failed	no response. Rtn value: 3	No response, stopped by crew	0	
	T03	Multiple thruster FDI	01:01	9 = not enabled	failed	no response. Rtn value:9	sphere lost comm right after being enabled before starting the test	0	
	T03	Multiple thruster FDI	01:43	0 = lost comm	success	appeared to be as described. Rtn value: 0	Test succeeded, although might have lost comm right before ending	0.3	
	T04	Closed loop attitude control	03:22	1 = success	success	rotations were highly coupled	Test proceded as expected	1.6	
	T05	FDI with attitude control	02:57	1 = success	success	rtn value: 1	Test proceded as expected	0.6	
P113	T01	Quick checkout	01:29	no results	failed		no crew feedback file, comm lost too early	0	
	T02	Basic Position Hold	01:00	0 = lost comm	failed	Immediate rtn value of zero	lost comm right after test started	0	
	T02	Basic Position Hold	06:21	1 = success	success	rtn value: 1	started very close to beacon, hold position successfully low battery for most of the test	0.7	x
	T15	Attitude path following	03:49	1 = success	success	multi-axis. rtn value: 1	test proceded as expected low battery during the whole test	0.1	x
	T03	Stationkeeping 3D - 1	02:24	0 = lost comm	success	rtn value: 1	perturbation occurred too early and too many perturbations by crew reset due to low battery in the middle of the test but got good data	1	x
	T08	De-tumble, Track, & Dock	01:28	0 = lost comm	failed	Satellite stopped test after initiating tumble.	reset due to low battery in the middle of the test	0.2	x
	T08	De-tumble, Track, & Dock	02:49	0 = lost comm	failed	test terminated after satellite induced tumble. Rtn value: 0	reset due to low battery in the middle of the test	0.2	x
	T18	De-tumble, Track, & Dock	01:00	no results	failed		no crew feedback file, comm lost too early low battery	0	x
# tests	31						total gas consumption for the second test session	6.542	
total duration	1:41:13						current tank status	60.626	
avg duration/test	03:16								
low batt duration	09:17								

Table E.3: ISS test session 03 overview.

SPHERES Test Session TS003 Results Summary											
Program	Test	Description	Start time	Interval	Tank SN2	Result SN2	Tank SN3	Result SN3	Evaluation	Crew notes	Notes
P124	T1	Quick Checkout	16:24:10	02:24	0.92%	1 (normal)	n/a	n/a	Good		Short IR noise between t=3-4s
MIT Docking	T2	3D Position Hold with Disturbance	16:26:34	06:20	5.21%	1 (normal)	n/a	n/a	IR Noise		Substantial IR during convergence
	T2	3D Position Hold with Disturbance	16:32:54	05:53	0.53%	1 (normal)	n/a	n/a	OK		Good convergence, IR noise during control: caused drift
	T3	Docking PD (1.5m)	16:38:47	04:30	2.44%	1 (normal)	n/a	n/a	OK		Converged well. Substantial IR once the overhead wall reached; had to re-converge after IR spikes
	T4	De-tumble, Track, & Dock Set 1	16:43:17	06:09	6.43%	1 (normal)	n/a	n/a	Good		Converged well. Some IR noise present once the overhead wall reached, but it recovered quickly and made contact with the beacon. Great initial "pointing" rotation.
	T5	Trajectory 3 (Safety w/rotation	16:49:26	04:05	5.11%	1 (normal)	n/a	n/a	Good		Very good control, no IR noise
	T6	3D Position Hold (Robust)	16:53:31	17:33	4.30%	1 (normal)	n/a	n/a	Good		Converged well originally. At some point the satellite lost control: must research if it was the estimator or the controller. No IR noise during test (some during download).
P126	T1	ID all axes	17:11:04	02:15	3.84%	1 (normal)	n/a	n/a	OK		Full data download. It appears some thrusters did not operate as expected. Minimal IR noise.
Mass ID	T2	ID all axes, proof mass	17:13:19	02:21	3.84%	1 (normal)	n/a	n/a	OK		Full data download. It appears some thrusters did not operate as expected. Minimal IR noise. Repeated since no proof mass was attached.
	T2	ID all axes, proof mass	17:15:40	02:22	3.80%	1 (normal)	n/a	n/a	OK	Proof mass (battery) attached on -X face	Full data download. It appears some thrusters did not operate as expected. Minimal IR noise.
	T3	Single-thruster, proof mass	17:18:02	02:54	0.02%	1 (normal)	n/a	n/a	IR Noise	proof mass attached	Substantial IR noise prevented any thruster firings
	T4	Single-thruster firings	17:20:56	02:33	3.70%	1 (normal)	n/a	n/a	Unclear	w/o mass, made contact with structure prior to completion of firings. Will repeat test	IR noise and thrusters not opening throughout the test.
	T4	Single-thruster firings	17:23:29	02:02	2.73%	1 (normal)	n/a	n/a	IR Noise		Obtained some good data. IR noise at the start of the test, then noise subdued.
	T5	Fuel slosh	17:25:31	01:20	0.36%	1 (normal)	n/a	n/a	Unclear	truster firings induced mostly a rotation, not a translation. Will repeat test.	Obtained some data. IR noise at end of test and thrusers did not open as expected.
	T5	Fuel slosh	17:26:51	02:24	0.64%	1 (normal)	n/a	n/a	Unclear	Again, more rotation than translation. (performed reset prior to this test)	Obtained some data. IR noise at end of test and thrusers did not open as expected.
	T6	Roll-Pitch-Axis spin	17:29:15	01:18	0.64%	1 (normal)	n/a	n/a	Good	as expected	All thrusters worked; no IR noise
	T7	Pitch-Yaw-Axis spin	17:30:33	01:09	0.64%	1 (normal)	n/a	n/a	OK	as expected	One thruster did not operate as expected
	T8	Yaw-Roll-Axis spin	17:31:42	10:03	0.53%	1 (normal)	n/a	n/a	OK	as expected	Obtained some good data. IR noise during the test delayed one thruster; one thruster did not operate as expected
P125	T1	Quick Checkout	17:41:45	08:18	0.13%	1 (normal)	0.00%	1 (normal)	IR Noise		Substantial IR noise during deployment caused thrusters to open at wrong times
MIT 2 Sat Initial	T2	Twin Rotations: Independent	17:50:03	10:09	0.81%	1 (normal)	0.27%	255 (reset)	Good / Low Battery	changed batteries in Blue (follower)	Test did not complete due to low battery on SN3; SN2 performed test succesfully
	T2	Twin Rotations: Independent	18:00:12	03:52	0.79%	1 (normal)	1.01%	1 (normal)	Good		SN2 performed rotations successfully. SN3 had substantial IR noise, but satellite recovered and performed rotations. Satellites rotates in opposite directions since these rotations are independent.
	T3	Twin Rotations: Formation	18:04:04	04:28	1.27%	1 (normal)	1.41%	1 (normal)	Good		SN2 had substantial IR noise, but it performed the rotations as commanded (with some delay). SN3 (also had IR noise) followed with some lag as expected (IR noise increased lag). Both satellites responded to Y-axis exitation by crew as expected
	T4	Twin Position Hold: Formation	18:08:32	09:48	21.60%	1 (normal)	25.55%	6 (timeout)	IR Noise	Many bounces off structure	SN2 had substantial IR noise during convergce region; estimator was not able to overcome the problem and the satellite moved randomly around work area. SN3's estimator was never able to converge either, and moved randomly around the area.
	T5	Two Satellite Docking - Set 1	18:18:20	07:41	0.00%	1 (normal)	6.30%	6 (timeout)	Not possible to deploy	Red did not respond.	SN2 had too much rotational drift during deployment - this was somewhat expected, as the satellite had no control whatsoever. Because the satellite rotated too much, the SN3 estimator was never able to converge.
# tests	24		*18:26:01*								
total duration	2:01:51										
avg duration/test	05:05			Total Tank	70.27%		34.53%				

Table E.4: ISS test session 04 overview.

SPHERES Test Session TS004 Results Summary											
Program	**Test**	**Description**	**Start time**	**Interval**	**Tank SN2**	**Result SN2**	**Tank SN3**	**Result SN3**	**Evaluation**	**Crew notes**	**Notes**
P131	T1	Quick Checkout	13:30:23	03:31	0.53%	255 (reset)	n/a	n/a	Incomplete		Test reset because of IR floading after running for 3.6 sec
MIT 4: Global Metrology Sys-ID	T2	Global Sys ID: Center	13:33:54	03:38	0.00%	101 (normal+IR)	n/a	n/a	success, got good data	return value 101	IR interference during data download only. Beacon location not received, preventing the estimator from converging
	T3	Global Sys ID: Beacon 1	13:37:32	03:30	0.00%	101 (normal+IR)	n/a	n/a	success, got good data	rtn 101	IR interference mostly during data download. Beacon location not received.
	T4	Global Sys ID: Beacon 3	13:41:02	03:27	0.00%	101 (normal+IR)	n/a	n/a	success, got good data	rtn 101	Data indicated a reset toward the end of data download caused by IR floading. Beacon location not received
	T5	Global Sys ID: Beacon 5	13:44:29	03:36	0.00%	1 (normal)	n/a	n/a	success, got good data	rtn 1	Beacon location not received
	T6	Global Sys ID: No beacon corner 1	13:48:05	03:26	0.00%	1 (normal)	n/a	n/a	success, got good data	rtn 1	Beacon location not received
	T7	Global Sys ID: No beacon corner 2	13:51:31	03:18	0.00%	1 (normal)	n/a	n/a	success, got good data	rtn 1	Some IR interference toward the middle of the test, but not enough to trigger the IR warning. Beacon location not received.
	T2	Global Sys ID: Center	13:54:49	03:42	0.00%	1 (normal)	n/a	n/a	success, got good data	rtn 1	Beacon location not received
	T8	3D Position Hold with Disturbance	13:58:31	04:10	13.96%	255 (reset)	n/a	n/a	Incomplete, reset occurred prior sending raw data	rtn 255	Just prior the reset, there is a noticeable IR noise increased, although it is lower than previously. The Sphere could not hold position and attitude because the estimator did not converge (beacon location not received)
	T3	Global Sys ID: Beacon 1	14:02:41	03:27	0.00%	1 (normal)	n/a	n/a	success, got good data	rtn 1	Beacon location not received
	T4	Global Sys ID: Beacon 3	14:06:08	03:21	0.00%	1 (normal)	n/a	n/a	success, got good data	rtn 1	Beacon location not received
	T8	3D Position Hold with Disturbance	14:09:29	10:36	17.37%	1 (normal)	n/a	n/a	partial success, got good sensor data	Steady volley of 5-6 jet firings at about 1 Hz with no apparent change to rotational or translational drift. Rtn value: 1	The Sphere could not hold position and attitude because the estimator did not converge (beacon location not received). Few low battery warnings.
	T9	Trajectory: 3D Avoidance	14:20:05	07:52	12.54%	1 (normal)	n/a	n/a	partial success, got good sensor data	low CO2	The Sphere could not maneuver accordingly because the estimator did not converge (beacon location not received). More low battery warnings.
	T9	Trajectory: 3D Avoidance	14:27:57	00:35	0.00%	n/a	n/a	n/a	n/a		Test not started
	T9	Trajectory: 3D Avoidance	14:28:32	10:02	14.06%	101 (normal+IR)	n/a	n/a	partial success, got good sensor data	low batt. Rtn 101	The Sphere could not maneuver accordingly because the estimator did not converge (beacon location not received). Data indicated a reset toward the end of data download caused by low batteries.
	T9	Trajectory: 3D Avoidance	14:38:34	09:44	11.94%	1 (normal)	n/a	n/a	partial success, got good sensor data	rtn value 1	The Sphere could not maneuver accordingly because the estimator did not converge (beacon location not received).
	T10	Trajectory: Avoidance with Rotation	14:48:18	04:42	20.25%	3 (stopped by crew)	n/a	n/a	incomplete, interrupted before the end	rtn value: 3	The Sphere could not maneuver accordingly because the estimator did not converge (beacon location not received).
			14:53:00	Approximation							
P132	T1	Quick Checkout	15:26:33	04:05	1.03%	101 (normal+IR)	1.05%	101 (normal+IR)	success	Red pressure ~23 psi	
MIT 4: Global Metrology Multi Sat	T2	Ultrasound Shadow: 1.5m	15:30:38	04:44	0.00%	1 (normal)	0.00%	1 (normal)	success	rtn 1	
	T3	Ultrasound Shadow: 0.5m	15:35:22	05:04	0.00%	101 (normal+IR)	0.00%	1 (normal)	success	red 101, blue 21	
	T4	Ultrasound Shadow: 0.2m	15:40:26	04:55	0.00%	1 (normal)	0.00%	1 (normal)	success	both sat rtn: 1	
	T5	Gyroscope Calibration	15:45:21	08:37	0.78%	101 (normal+IR)	1.74%	101 (normal+IR)	success	both returned 101	
	T6	2 Sat. Position Hold - Independent	15:53:58	07:33	3.85%	1 (normal)	1.64%	101 (normal+IR)	success	Blue 101, Red 1	
	T7	2 Sat. Leader, Follower	16:01:31	07:56	1.76%	1 (normal)	2.26%	1 (normal)	success	rtn value 1 for both	
	T8	2 Sat. Docking: Target Hold	16:09:27	06:16	11.29%	255 (reset)	8.29%	1 (normal)	partial success, got good sensor data		Estimator diverges at time t=65 sec, possibly after contact. Red resets during data download, likely because of IR floading
	T8	2 Sat. Docking: Target Hold	16:15:43	05:32	2.21%	1 (normal)	2.02%	1 (normal)	success	rtn values: 1 for both	
	T9	3D Formation	16:21:15	07:32	3.32%	101 (normal+IR)	2.85%	101 (normal+IR)	success	rtn value 101 for both	
	T10	2 Sat. De-tumble, track, & dock	16:28:47	06:05	3.17%	101 (normal+IR)	1.82%	101 (normal+IR)	success	rtn 101 for both	
	T11	2 Sat. Docking: Plume Impingement Check	16:34:52	09:10	1.83%	1 (normal)	0.81%	101 (normal+IR)	success	red rtn value: 1, blue rtn value: 101	
# tests	29		*16:44:02*	Approximation from data in the Notes file							
total duration	2:40:06										
avg duration/test	05:31			**Total Tank**	119.89%		22.48%				

Table E.5: ISS test session 05 overview.

SPHERES Test Session TS005 Results Summary											
Program	**Test**	**Description**	**Start time**	**Interval**	**Tank SN2**	**Result SN2**	**Tank SN3**	**Result SN3**	**Evaluation**	**Crew notes**	**Notes**
P142	T1	ID all axes	20:26:58	10:57						Satellite moved to starboard while rotating slowly, gently contacted LAB1S2 rack, then returned toward center of work area.	
"Mass ID 2 (NASA Ames)"	T2	ID all axes, proof mass	20:37:55	00:19							
	T2	ID all axes, proof mass	20:38:14	02:43						Satellite exited the work area aft just at the end of thruster activity. Did not abort test.	
	T3	Single-thruster, proof mass	20:40:57	03:44						Satellite exited work area forward starboard, contacting starboard wall of LAB1 forward alcove just after last thruster firing.	
	T4	Single-thruster firings	20:44:41	02:56						Satellite exited work area aft.	
	T5	Fuel slosh	20:47:37	03:39						Downward firing thrusters inop?	
	T6	Roll-Pitch-axis spin	20:51:16	05:12						Yaw rate was very slow; approx 1 revolution per 30 seconds. Other axes around 8 seconds per rev.	
	T7	Pitch-Yaw-axis spin	20:56:28	01:28							
	T8	Yaw-Roll-axis spin	20:57:56	01:18							
	T9	Manual calibrations	20:59:14	01:41							
	T10	Roll-axis spin	21:00:55	01:26							
	T11	Pitch-axis spin	21:02:21	01:07							
	T12	Yaw-axis spin	21:03:28	01:18							
	T1	ID all axes	21:04:46	08:56							
	T1	ID all axes	21:13:42	03:06							
	T1	ID all axes	21:16:48	02:08							
		Bootload P141	*21:18:56*	*09:10*							
P141	T1	Quick Checkout	21:28:06	04:29	0.78%	1 (normal)	0.53%	255 (reset)	partial success	Gauge readings: Red 27; Blue 25. Firings were synchronous except for final, long firing. Blue fired one ~1.5 second firing; Red firing was stocatto.	bad battery contact suspected to have caused the blue satellite to reset in the middle of the test (bad insertion or bad battery)
MIT 5: Docking Multi Sat	T1	Quick Checkout	21:32:35	14:26	0.77%	1 (normal)	0.53%	255 (reset)	partial success	Red and Blue gauges read 26. Same results as previous test. Red returned 1; Blue returned 255.	bad battery contact suspected to have caused the blue satellite to reset in the middle of the test (bad insertion or bad battery)
	T1	Quick Checkout	21:47:01	07:59	0.82%	1 (normal)	0.55%	255 (reset)	partial success	Red 28; Blue 27	bad battery contact suspected to have caused the blue satellite to reset in the middle of the test (bad insertion or bad battery)
	T1	Quick Checkout	21:55:00	02:11	0.76%	1 (normal)	0.74%	1 (normal)	success	Red 26; Blue 27	
	T2	Docking to Fixed Target	21:57:11	10:59	1.52%	1 (normal)	0.76%	1 (normal)	success	Docking would have been successful with better docking mechanism!	
	T6	Docking to Tumbling Target	22:08:10	04:59	2.13%	103 (stopped by crew + IR noise)	1.03%	103 (stopped by crew + IR noise)	success, got good data		very little IR noise (did not affect the estimator)
	T6	Docking to Tumbling Target	22:13:09	07:37	2.13%	103 (stopped by crew + IR noise)	0.94%	103 (stopped by crew + IR noise)	success		very little IR noise (did not affect the estimator)
	T5	Safe Docking w/fault	22:20:46	07:23	1.43%	103 (stopped by crew + IR noise)	0.92%	3 (stopped by crew)	success, got good data	There was motion perpendicular to docking axis at contact, approximately equal in magnitude to motion along docking axis.	very little IR noise (did not affect the estimator)
	T4	Safe Docking	22:28:09	06:49	1.85%	103 (stopped by crew + IR noise)	1.16%	103 (stopped by crew + IR noise)	success		very little IR noise (did not affect the estimator)
	T12	Circular Formation Flight	22:34:58	06:25	2.69%	1 (normal)	3.47%	1 (normal)	success, got good data		
# tests	26	*Test ends*	*22:41:23*								
total duration to run tests	2:14:25										
avg duration/test	05:10			**Total Tank**	14.88%		10.63%				

Table E.6: ISS test session 06 overview.

SPHERES Test Session TS006 Results Summary													
Program	**Test**	**Description**	**Start time**	**Interval**	**Tank SN1**	**Result SN1**	**Tank SN2**	**Result SN2**	**Tank SN3**	**Result SN3**	**Evaluation**	**Crew notes**	**Notes**
P141	T1	Quick Checkout	20:03:19	06:57			0.75%	1 (normal)	0.72%	1 (normal)	success		
MIT 5: Docking & Reconfiguration	T7	Docking to Tumbling Target w/ fault	20:10:16	09:41			0.79%	7 (error)	0.71%	101 (normal + IR noise)	success		
	T3	Docking to Fixed Target w/ fault	20:19:57	06:09			1.39%	1 (normal)	0.73%	1 (normal)	success		
	T8	Reconfiguration: 2 Satellites Attached	20:26:06	04:47			4.64%	1 (normal)	0.00%	1 (normal)	success		
	T11	Trajectory 1	20:30:53	03:27			1.74%	103 (stopped by crew + IR noise)	5.37%	3 (stopped by crew)	partial success		IR spike caused blue sat to reset at time 61s
	T11	Trajectory 1	20:34:20	05:24			2.88%	255 (reset)	1.16%	255 (reset)	failed		low battery level suspected to have caused the red satellite to reset in the middle of the test
	T11	Trajectory 1	20:39:44	10:56			1.37%	255 (reset)	0.92%	255 (reset)	failed	255 was the number for both. We ran this test a couple of times and it seems like we always got bad numbers. Have battery low light, so going to change the batteries and rerun	low battery level suspected to have caused the red satellite to reset in the middle of the test
	T11	Trajectory 1	20:50:40	07:11			2.29%	1 (normal)	2.05%	1 (normal)	success	Good Test	
	T9	Safe Docking to Tumbling Target	20:57:51	09:44			1.58%	1 (normal)	1.20%	101 (normal + IR noise)	success		
	T10	Safe Docking to Tumbling Target w/ fault	21:07:35	10:52			1.54%	101 (normal + IR noise)	1.51%	101 (normal + IR noise)	success		very little IR noise (did not affect the estimator)
	T14	Trajectory 2	21:18:27	02:59			0.91%	3 (stopped by crew)	0.00%	3 (stopped by crew)	failed	faced wrong direction	not deployed properly
	T14	Trajectory 2	21:21:26	07:44			1.93%	1 (normal)	0.00%	1 (normal)	success		
	T13	Reconfiguration: Satellite plus proof mass	21:29:10	03:18			5.69%	101 (normal + IR noise)	0.00%	1 (normal)	success		
		Bootload P131	*21:32:28*	*08:59*									very little IR noise (did not affect the estimator)
P131	T1	Quick Checkout	21:41:27	02:03	0.14%	255 (reset)					failed		bad battery contact suspected to have caused the orange satellite to reset in the middle of the test (bad insertion or bad battery)
MIT 4: Global Metrology Sys-ID	T1	Quick Checkout	21:43:30	03:38	0.20%	255 (reset)					failed		bad battery contact suspected to have caused the orange satellite to reset in the middle of the test (bad insertion or bad battery)
	T1	Quick Checkout	21:47:08	02:10	0.73%	1 (normal)					success		
	T2	Global Sys ID: Center	21:49:18	04:07	0.00%	1 (normal)					success		
	T8	3D Position Hold with Disturbance	21:53:25	02:41	1.41%	255 (reset)					failed		bad battery contact suspected to have caused the orange satellite to reset in the middle of the test (bad insertion or bad battery)
		Bootload P142	*21:56:06*	*10:27*									
P142	T1	ID all axes	22:06:33	02:32									
Mass ID 2 (NASA Ames)	T1	ID all axes	22:09:05	02:23									
	T1	ID all axes	22:11:28	02:19									
	T2	ID all axes, proof mass	22:13:47	02:08									
	T3	Single-thruster, proof mass	22:15:55	02:36									
	T4	Single-thruster firings	22:18:31	02:17									
	T5	Fuel slosh	22:20:48	02:04								22 psi	
	T6	Manual calibrations	22:22:52	03:40								need to repeat	
	T6	Manual calibrations	22:26:32	09:01								22 psi	
			22:35:33										
# tests	27												
total duration to run tests	2:32:14												
avg duration/test	05:38		**Total Tank**		2.48%		27.49%		14.37%				

Bibliography

[1] www.mathworks.com. MATLAB is a registered trademark of The Mathworks, Inc., ©1994-2007.

[2] QA750 Q-Flex Accelerometer. Available at http://www.inertialsensor.com/qa750.shtml as of April 2007.

[3] QRS14 (GyroChip II) Micromachined Angular Rate Sensor. Available at http://www.systron.com/pro_QRS14.asp as of April 2007.

[4] A brief description of the radar approach equipment of the Soyuz-type spacecraft ("Kratkoye opisaniye radioapparatury sblisheniya kosmicheskikh korabley tipa Soyuz"). Available at http://history.nasa.gov/astp/APSYZ-descharac3.PDF as of April 2007, November 1970. NASA Technical Translation.

[5] Guide for the verification and validation of computational fluid dynamics simulations. AIAA G-077-1998, January 1998. Guide.

[6] JSC Reduced Gravity Program Users Guide. Available at http://jsc-aircraft-ops.jsc.nasa.gov/ReducedApril 2007, August 2005. AOD 33899 Rev A PCN 1.

[7] Overview of the DART mishap investigation results. Available at http://www.nasa.gov/pdf/148072main_DART_mishap_overview.pdf as of May 2007, May 2006. For public release.

[8] David L. Akin, J. Corde Lane, Brian J. Roberts, and Steven R. Weisman. Robotic capabilities for complex space operations. In *AIAA Space 2001 - Con-*

ference and Exposition, number AIAA-2001-4538, Albuquerque, New Mexico, August 2001.

[9] Edwin E. Aldrin, Jr. *Line-of-sight guidance techniques for manned orbital rendezvous.* Sc. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, February 1963.

[10] M. Athans, D. Castanon, K. Dunn, C. Greene, Wing Lee, N. Sandell, Jr., and A. Willsky. The stochastic control of the F-8C aircraft using a multiple model adaptive control (MMAC) method-Part I: Equilibrium flight. *IEEE Transactions on Automatic Control*, Vol. 22(Num. 5):pp 768–780, October 1977.

[11] Alberto Bemporad, Francesco Borrelli, and Manfred Morari. Model predictive control based on linear programming - the explicit solution. Technical Report Tech. Report AUT01-06, Automatic Control Laboratory, ETH Zentrum, ETL I 26, CH-8092 Zurich, Switzerland, January 2001.

[12] Pierre Blanc-Paques. Thruster calibration for the SPHERES spacecraft formation flight testbed. Internal research report, June 2005.

[13] F. Bonfatti, G. Gadda, and P.D. Monari. PLC software modularity and cooperative development. In *Proceedings of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, volume 2, pages 775–780, July 2001.

[14] Louis Breger. *Control of Spacecraft in Proximity Orbits.* Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, February 2007.

[15] Louis Breger and Jonathan P. How. Safe trajectories for autonomous rendezvous of spacecraft. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, number AIAA-2006-6584, Keystone, Colorado, August 2006.

[16] Robert G. Brown and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley & Sons, New York, NY, third edition, 1997.

[17] H. J. Buchanan, M. S. Hopkins, and Z. J. Galaboff. Uncontrolled dynamics of the Skylab vehicle. *Journal of Guidance and Control*, Vol. 4(Num. 3):pp 277–283, May-June 1981.

[18] Joel D. Burcham, Michael K. Balch, and Stephen R. Granade. A correlator-based video sensor for docking with the Hubble Space Telescope. In Peter Tchoryk, Jr. and Brian Holz, editors, *Spaceborne Sensors II*, volume 5798, pages 130–138. SPIE, 2005.

[19] Allen Chen. Propulsion system characterization for the SPHERES formation flight and docking testbed. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2002.

[20] Allen Chen, Alvar Saenz-Otero, Mark Hilstad, and David Miller. Development of formation flight and docking algorithms using the SPHERES testbed. In *15th Annual USU Conference on Small Satellites*, number SSC01-VIIIa-2. AIAA/USU, August 2001.

[21] Wei Chen, Lusine Baghdasaryan, Thaweepat Buranathiti, and Jian Cao. Model validation via uncertainty propagation and data transformations. *AIAA Journal*, Vol. 42(Num. 7):pp 1406–1415, July 2004.

[22] Soon-Jo Chung. *Nonlinear Control and Synchronization of Multiple Lagrangian Systems with Application to Tethered Formation Flight Spacecraft.* Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2007.

[23] Soon-Jo Chung, Jean-Jacques E. Slotine, and David W. Miller. Nonlinear model reduction and decentralized control of tethered formation flight. *Journal of Guidance, Control and Dynamics*, Vol. 30(Num. 2):pp 390–400, March-April 2007.

[24] John L. Crassidis. Sigma-point Kalman filtering for integrated GPS and inertial navigation. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, number AIAA 2005-6052, San Francisco, California, August 2005.

[25] John L. Crassidis and F. Landis Markley. Unscented filtering for spacecraft attitude estimation. *Journal of Guidance, Control and Dynamics*, Vol. 26(Num. 4):pp 536–542, July-August 2003.

[26] Edward Crawley, Olivier de Weck, Steven Eppinger, Christopher Magee, Joel Moses, Warren Seering, Joel Schindall, David Wallace, and Daniel Whitney. The influence of architecture in engineering systems. Technical report, Massachusetts Institute of Technology, Cambridge, MA, March 2004. Available at http://esd.mit.edu/symposium/pdfs/monograph/architecture-b.pdf as of April 2007.

[27] Glenn Creamer, Frank Pipitone, Charmaine Gilbreath, Dexter Bird, and Sam Hollander. NRL technologies for autonomous inter-spacecraft rendezvous and proximity operations. In *The John L. Junkins Astrodynamics Symposium, AAS/AIAA Space Flight Mechanics Meeting*, number AAS-03-272 in Advances in the Astronautical Sciences, pages 233–250, College Station, Texas, May 2003.

[28] K. C. Daly, E. Gai, and J. V. Harrison. Generalized liklihood test for FDI in redundant sensor configurations. *Journal of Guidance and Control*, Vol. 2:pp 9–17, January-February 1979.

[29] Thomas M. Davis and David Melanson. XSS-10 micro-satellite flight demonstration program results. In Peter Tchoryk, Jr. and Melissa Wright, editors, *Spacecraft Platforms and Infrastructure*, volume 5419, pages 16–25. SPIE, 2004.

[30] John Van de Vegte. *Feedback Control Systems.* Prentice-Hall, third edition, 1994.

[31] R. Dearden, T. Willeke, R. Simmons, V. Verma, F. Hutter, and S. Thrun. Real-time fault detection and situational awareness for rovers: report on the Mars technology program task. In *Proceedings of the 2004 IEEE Aerospace Conference*, volume 2, pages 826–840, March 2004.

[32] R. Delage, N. Durante, JJ. Wasbauer, JF. Goester, and D. Cornier. An overview of ATV integrated mission analysis and mission preparation. In *54th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, number IAC-03-V.2.09, Bremen, Germany, September 2003.

[33] John J. Deyst and James C. Deckert. Maximum likelihood failure detection techniques applied to the Shuttle RCS jets. *Journal of Spacecraft and Rockets*, Vol. 13(Num. 2):pp 65–74, February 1976.

[34] G.A. Dorais and Y. Gawdiak. The personal satellite assistant: an internal spacecraft autonomous mobile monitor. In *Proceedings of the 2003 IEEE Aerospace Conference*, volume 1, pages 1–348, March 2003.

[35] Michael A. Dornheim. Rendezvous trials. *Aviation Week & Space Technology*, Vol. 162:pp 35–36, April 2005.

[36] Riley M. Duren. Validation and verification of deep-space missions. *Journal of Spacecraft and Rockets*, Vol. 41(Num. 4):pp 651–658, July-August 2004.

[37] D. Dvorak, R. Rasmussen, G. Reeves, and A. Sacks. Software architecture themes in JPL's Mission Data System. In *Proceedings of the 2000 IEEE Aerospace Conference*, volume 7, pages 259–268, March 2000.

[38] J. Enright, M. Hilstad, A. Saenz-Otero, and D. Miller. The SPHERES Guest Scientist Program: collaborative science on the ISS. In *Proceedings of the 2004 IEEE Aerospace Conference*, volume 1, March 2004.

[39] R. L. Farrenkopf. Analytic steady-state accuracy solutions for two common spacecraft attitude estimators. *Journal of Guidance and Control*, Vol. 1(Num. 4):pp 282–284, July-August 1978.

[40] Wigbert Fehse. *Automated Rendezvous and Docking of Spacecraft.* Number 16 in Cambridge Aerospace Series. Cambridge University Press, Cambridge, United Kingdom, November 2003.

[41] E. Frazzoli, M.A. Dahleh, and E. Feron. A hybrid control architecture for aggressive maneuvering of autonomous helicopters. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 3, pages 2471–2476, December 1999.

[42] Steven E. Fredrickson, Steve Duran, Nathan Howard, and Jennifer D. Wagenknecht. Application of the mini AERCam free flyer for orbital inspection. In Peter Tchoryk, Jr. and Melissa Wright, editors, *Spacecraft Platforms and Infrastructure*, volume 5419, pages 26–35. SPIE, 2004.

[43] Steven E. Fredrickson, Steve G. Duran, Angela N. Braun, Timothy M. Straube, and Jennifer D. Mitchell. AERCam autonomy: Intelligent software architecture for robotic free flying nanosatellite inspection vehicles. In *Space 2006*, number AIAA-2006-7392, San Jose, California, September 2006.

[44] M. Fujita, Y. Kuroki, T. Ishida, and T.T. Doi. Autonomous behavior control architecture of entertainment humanoid robot SDR-4X. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 1, pages 960–967, October 2003.

[45] T. Fujiwara, T. Nakamura, Y. Tsuda, and S. Nakasuka. Motion estimation and capturing of tumbling object in non-gravitational field. In *Proceedings of IFAC Automatic Control in Aerospace*, pages 77–82, Seoul, Korea, 1998.

[46] H.C. Gelderloos, S.I. Sheikh, and B.W. Schipper. GPS attitude system integrated with an inertial system for space applications. In *Proceedings of the 16th DASC. AIAA/IEEE Digital Avionics Systems Conference*, volume 2, page 8.1, October 1997.

[47] Philip M. Gerhart, Richard J. Gross, and John I. Hochstein. *Fundamentals of Fluid Mechanics.* Addison-Wesley Publishing Company, second edition, September 1993.

[48] Robert Godwin, editor. *Rocket and Space Corporation Energia.* Apogee Books, Burlington, Ontario, Canada, 2001.

[49] P. Goel, G. Dedeoglu, S.I. Roumeliotis, and G.S. Sukhatme. Fault detection and identification in a mobile robot using multiple model estimation and neural network. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, volume 3, pages 2302–2309, April 2000.

[50] Caroline P. Graettinger, Suzanne Garcia, Jeannine Siviy, Robert J. Schenk, and Peter J. Van Syckle. Using the technology readiness levels scale to support technology management in the DoDs ATD/STO environments. A Findings and Recommendations, Report Conducted for Army CECOM CMU/SEI-2002-SR-027, Carnegie Mellon Software Engineering Institute, Pittsburgh, PA, September 2002. http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02sr027.pdf.

[51] Hari B. Hablani, Myron Tapper, and David Dana-Bashian. Guidance algorithms for autonomous rendezvous of spacecraft with a target vehicle in circular orbit. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, number AIAA-2001-4393, Montreal, Canada, August 2001.

[52] S. R. Hall. Parity vector compensation for FDI. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, February 1982.

[53] S. R. Hall, P. Motyka, E. Gai, and J. J. Deyst, Jr. In-flight parity vector compensation for FDI. In *Proceedings of the 1982 IEEE National Aerospace and Electronics Conference (NAECON 1982)*, Dayton, OH, May 1982.

[54] Steven R. Hall, Edward F. Crawley, and Jonathan P. How. Hierarchic control architecture for intelligent structures. *Journal of Guidance, Control, and Dynamics*, Vol. 14(Num. 3):pp 503–512, May-June 1991.

[55] G. Heimbold, J.J.M. Prins, and W. Fehse. Epos: European proximity operations simulation. In *Proceedings of the 1st European In-Orbit Operations Technology Symposium*, number N88-19484 12-12, pages 273–279, 1987.

[56] Philip G. Hill and Carl R. Peterson. *Mechanics and Thermodynamics of Propulsion.* Addison-Wesley Publishing Company, second edition, June 1992.

[57] Mark O. Hilstad. A multi-vehicle testbed and interface framework for the development and verification of separated spacecraft control algorithms. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2002.

[58] Mark O. Hilstad, John P. Enright, and Arthur G. Richards. The SPHERES Guest Scientist Program. http://ssl.mit.edu/spheres/gsp/spheres-gsp.pdf, October 2003.

[59] Jonathan P. How and Steven R. Hall. Local control design methodologies for a hierarchic control architecture. *Journal of Guidance, Control, and Dynamics*, Vol. 15(Num. 3):pp 654–663, May-June 1992.

[60] G. Hunyadi, J. Ganley, A. Peffer, and M. Kumashiro. The University Nanosat Program: an adaptable, responsive and realistic capability demonstration vehicle. In *Proceedings of the 2004 IEEE Aerospace Conference*, volume 5, March 2004.

[61] Michel D. Ingham, Robert D. Rasmussen, Matthew B. Bennett, and Alex C. Moncada. Engineering complex embedded systems with state analysis and the Mission Data System. *Journal of Aerospace Computing, Information, and Communication*, Vol. 2(Num. 12):pp 507–536, December 2005.

[62] Stephen A. Jacklin, Michael R. Lowry, Johann M. Schumann, Pramod P. Gupta, John T. Bosworth, Eddie Zavala, John W. Kelly, Kelly J. Hayhurst, Celeste M. Belcastro, and Christine M. Belcastro. Verification, validation, and certification challenges for adaptive flight-critical control system software. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, number AIAA 2004-5258, Providence, Rhode Island, August 2004.

[63] S. Jacobsen, C. Lee, C. Zhu, and S. Dubowsky. Planning of safe kinematic trajectories for free flying robots approaching an uncontrolled spinning satellite. In *DETC2002/MECH*, Montreal, Canada, 2002.

[64] S. Julier, J. Uhlmann, and H.F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, Vol. 45(Num. 3):pp 477–482, March 2000.

[65] Isao Kawano, Masaaki Mokuno, Toru Kasai, and Takashi Suzuki. Result and evaluation of autonomous rendezvous docking experiments of ETS-VII. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, number AIAA-1999-4073, Portland, Oregon, August 1999.

[66] O. Kawasaki, T. Imada, K. Yamanaka, and T. Tanaka. On-orbit demonstration and operations plan of the H-II Transfer Vehicle (HTV). In *51st International Astronautical Congress IAF*, number IAF-00-T208, Rio de Janeiro, Brazil, October 2000.

[67] Brian S. Keller, Sivakumara S.K. Tadikonda, and Jalal Mapar. Autonomous acquisition, rendezvous, & docking using a video guidance sensor: Experimental testbed results. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, number AIAA-2002-4846, Monterey, CA, August 2002.

[68] Edmund M. C. Kong, Mark O. Hilstad, Simon Nolet, and David W. Miller. Development and verification of algorithms for spacecraft formation flight using the SPHERES testbed: application to TPF. In Wesley A. Traub, editor, *New Frontiers in Stellar Interferometry*, volume 5491, pages 308–319. SPIE, 2004.

[69] Edmund M. C. Kong, Alvar S. Otero, Simon Nolet, Dustin S. Berkovitz, David W. Miller, and Steve W. Sell. SPHERES as a formation flight algorithm development and validation testbed: Current progress and beyond. In *2nd International Symposium on Formation Flying Missions and Technologies*, Washington DC, September 2004.

[70] Deok-Jin Lee and Kyle T. Alfriend. Sigma point filtering for sequential orbit estimation and prediction. *Journal of Spacecraft and Rockets*, Vol. 44(Num. 2):pp 388–398, March-April 2007.

[71] E. J. Lefferts, F. L. Markley, and M. D. Shuster. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Dynamics and Control*, Vol. 5(Num. 5):pp 417–429, September-October 1982. AIAA-1982-0070.

[72] V. P. Legostaev and B. V. Raushenbach. Automatic assembly in space. *Cosmic Res*, Vol. 7(Num. 6):pp 723–731, Nov-Dec 1969.

[73] Edward A. LeMaster, David B. Schaechter, and Connie K. Carrington. Experimental demonstration of technologies for autonomous on-orbit robotic assembly. In *Space 2006*, number AIAA-2006-7428, San Jose, California, September 2006.

[74] Jean-François Lévesque. Second-order simplex sigma points for nonlinear estimation. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, number AIAA 2006-6093, Keystone, Colorado, August 2006.

[75] Roger W. Logan and Cynthia K. Nitta. Validation, uncertainty, and quantitative reliability at confidence (QRC). In *41st Aerospace Sciences Meeting and Exhibit*, number AIAA 2003-1337, Reno, Nevada, January 2003.

[76] Roger W. Logan, Cynthia K. Nitta, and Steven K. Chidester. Risk reduction as the product of model assessed reliability, confidence, and consequence. In *42nd AIAA Aerospace Sciences Meeting and Exhibit*, number AIAA 2004-1175, Reno, Nevada, January 2004.

[77] D. Magill. Optimal adaptive estimation of sampled stochastic processes. *IEEE Transactions on Automatic Control*, Vol. 10(Num. 4):pp 434–439, October 1965.

[78] John C. Mankins. Technology Readiness Levels. White paper, Advanced Concepts Office, Office of Space Access and Technology, NASA, April 1995. Available at http://www.hq.nasa.gov/office/codeq/trl/index.htm as of April 2007.

[79] F. Landis Markley, John L. Crassidis, and Yang Cheng. Nonlinear attitude filtering methods. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, San Francisco, California, August 2005.

[80] S. Matsumoto, Y. Ohkami, Y. Wakabayashi, M. Oda, and H. Ueno. Satellite capturing strategy using agile Orbital Servicing Vehicle, Hyper-OSV. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, volume 3, pages 2309–2314, May 2002.

[81] Shuichi Matsumoto, Steven Dubowsky, Stephen Jacobsen, and Yoshiaki Ohkami. Fly-by approach and guidance for uncontrolled rotating satellite capture. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, number AIAA-2003-5745, Austin, Texas, August 2003.

[82] David W. Miller and Gregory J. W. Mallory. Control testbeds and flight demonstrations: Transitioning theory to application. In *Proceedings of the 1998 American Control Conference*, pages 873–878, Philadelphia, Pennsylvania, June 1998.

[83] M. Nimelman, J. Tripp, A. Allen, D. M. Hiemstra, and S. A. McDonald. Spaceborne scanning lidar system (SSLS) upgrade path. In Edward M. Carapezza, editor, *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, volume 6201, page 62011V. SPIE, 2006.

[84] M. Nimelman, J. Tripp, G. Bailak, and J. Bolger. Spaceborne scanning lidar system (SSLS). In Peter Tchoryk, Jr. and Brian Holz, editors, *Spaceborne Sensors II*, volume 5798, pages 73–82. SPIE, 2005.

[85] Simon Nolet. The SPHERES navigation system: from early development to on-orbit testing. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, South Carolina, August 2007. Accepted for publication.

[86] Simon Nolet, Edmund Kong, and David W. Miller. Autonomous docking algorithm development and experimentation using the SPHERES testbed. In Peter

Tchoryk, Jr. and Melissa Wright, editors, *Spacecraft Platforms and Infrastructure*, volume 5419, pages 1–15. SPIE, 2004.

[87] Simon Nolet, Alvar Saenz-Otero, David W. Miller, and Amer Fejzic. SPHERES operations aboard the ISS: Maturation of GN&C algorithms in microgravity. In *30th Annual AAS Guidance and Control Conference*, number AAS 07-042, Breckenridge, Colorado, February 2007.

[88] William L. Oberkampf and Matthew F. Barone. Measures of agreement between computation and experiment: Validation metrics. In *34th AIAA Fluid Dynamics Conference and Exhibit*, number AIAA 2004-2626, Portland, Oregon, June-July 2004.

[89] Mitsushige Oda. Experiences and lessons learned from the ETS-VII robot satellite. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, volume 1, pages 914–919, April 2000.

[90] Eric A Olsen, Chan-Woo Park, and Jonathan P How. 3D formation flight using differential carrier-phase GPS sensors. *Navigation*, Vol. 46(Num. 1):pp 35–48, Spring 1999.

[91] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-Time Signal Processing.* Prentice Hall, second edition, February 1999.

[92] Jung-Min Park, Insup Song, Young-Jo Cho, and Sang-Rok Oh. A hybrid control architecture using a reactive sequencing strategy for mobile robot navigation. In *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99)*, volume 3, pages 1279–1284, October 1999.

[93] Sanghyuk Park, John J. Deyst, and Jonathan P. How. A new nonlinear guidance logic for trajectory tracking. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, number AIAA-2004-4900, Providence, Rhode Island, August 2004.

[94] D. J. Pearson. The glideslope approach. *Advances in the Astronautical Sciences*, Vol. 69:pp 109–123, 1989.

[95] Nelson Pedreiro. Spacecraft architecture for disturbance-free payload. *Journal of Guidance, Control, and Dynamics*, Vol. 26(Num. 5):pp 794–804, September-October 2003.

[96] A. J. Pejsa. Optimum orientation and accuracy of redundant sensor arrays. In *AIAA 9th Aerospace Sciences Meeting*, New York, NY, January 1971.

[97] M. E. Polites. An assessment of the technology of automated rendezvous and capture in space. Technical Report NASA / TP–1998 -208528, NASA Marshall Space Flight Center, July 1998.

[98] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, second edition, October 1992.

[99] Ilia Rapoport and Yaakov Oshman. Fault-tolerant particle filtering by using interacting multiple model-based rao-blackwellization. *Journal of Guidance, Control and Dynamics*, Vol. 28(Num. 6):pp 1171–1177, November-December 2005.

[100] Boris V. Rauschenbakh, Michael Yu. Ovchinnikov, and Susan McKenna-Lawlor. *Essential Spaceflight Dynamics and Magnetospherics.* The Space Technology Library, Kluwer Academic Publishers and Microcosm Press, Boston, MA, 2003.

[101] M.W. Regehr, A.B. Acikmese, A. Ahmed, M. Aung, K.C. Clark, P. MacNeal, J. Shields, G. Singh, R. Bailey, C. Bushnell, A. Hicke, B. Lytle, and R.E. Rasmussen. The formation control testbed. In *Proceedings of the 2004 IEEE Aerospace Conference*, volume 1, March 2004.

[102] Arthur Richards, Tom Schouwenaars, Jonathan P. How, and Eric Feron. Spacecraft trajectory planning with collision and plume avoidance using mixed-integer

linear programming. *Journal of Guidance, Control and Dynamics*, Vol. 25(Num. 4):pp 755–764, July-August 2002.

[103] Arthur George Richards. Trajectory optimization using mixed-integer linear programming. Master of science thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 2002.

[104] Arthur George Richards. *Robust Constrained Model Predictive Control.* Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, February 2005.

[105] L. Rodgers, N. Hoff, E. Jordan, M. Heiman, and D. W. Miller. A universal interface for modular spacecraft. In *Proceedings of the 19th Annual AIAA/USU Conference on Small Satellites*, Logan, Utah, August 2005.

[106] Lennon Rodgers. Concepts and technology development for the autonomous assembly and reconfiguration of modular space systems. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, February 2006.

[107] Lennon Rodgers, Simon Nolet, and David W. Miller. Development of the miniature video docking sensor. In Pejmun Motaghedi, editor, *Modeling, Simulation, and Verification of Space-based Systems III*, volume 6221, page 62210E. SPIE, 2006.

[108] Marcello Romano, David A. Friedman, and Tracy J. Shay. Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, number AIAA-2006-6798, Keystone, Colorado, August 2006.

[109] David S. Rubinstein and David W. Carter. Attitude control system design for return of the Kistler K1 orbital vehicle. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, number AIAA-1998-4420. Charles Stark Draper Lab., Inc., Cambridge, MA, August 1998.

[110] Timothy E. Rumford. Demonstration of autonomous rendezvous technology (DART) project summary. In Peter Tchoryk, Jr. and James Shoemaker, editors, *Space Systems Technology and Operations*, volume 5088, pages 10–19. SPIE, 2003.

[111] Michael Ruth and Chisholm Tracy. Video-guidance design for the DART rendezvous mission. In Peter Tchoryk, Jr. and Melissa Wright, editors, *Spacecraft Platforms and Infrastructure*, volume 5419, pages 92–106. SPIE, 2004.

[112] Alvar Saenz-Otero. The SPHERES satellite formation flight testbed: Design and initial control. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, August 2000.

[113] Alvar Saenz-Otero. *Design Principles for the Development of Space Technology Maturation Laboratories Aboard the International Space Station.* Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2005.

[114] W. T. Scofield and J. R. Mellin. A feasibility study of unmanned rendezvous and docking in Mars orbit. Technical Report MCR 74-244, Martin Marietta, September 1974. JPL contract 953746.

[115] James Shoemaker and Melissa Wright. Orbital Express space operations architecture program. In Peter Tchoryk, Jr. and Melissa Wright, editors, *Spacecraft Platforms and Infrastructure*, volume 5419, pages 57–65. SPIE, 2004.

[116] Marcel J. Sidi. *Spacecraft Dynamics and Control, A Practical Engineering Approach.* Number 7 in Cambridge Aerospace Series. Cambridge University Press, Cambridge, 1997.

[117] Robert F. Stengel. *Optimal Control and Estimation.* Dover Publications, Inc., New York, NY, Dover edition, 1994.

[118] Ben H. Thacker. The role of nondeterminism in computational model verification and validation. In *46th AIAA/ASME/ASCE/AHS/ASC Structures,*

Structural Dynamics & Materials Conference, number AIAA 2005-1902, Austin, Texas, April 2005.

[119] Patrick A. Tobbe, Marlin J. Williamson, and John R. Glaese. The Flight Robotics Laboratory. Technical Report 19890003224, NASA, NASA Marshall Space Flight Center, January 1988.

[120] Yuichi Tsuda and Shinichi Nakasuka. New attitude motion following control algorithm for capturing tumbling object in space. *Acta Astronautica*, Vol. 53(Num. 11):pp 847–861, December 2003.

[121] US Army Corps of Engineers. NAVSTAR Global Positioning System surveying. Engineer manual 1110-1-1003, Department of the Army, Washington, DC, July 2003. Available at http://www.usace.army.mil/inet/usace-docs/eng-manuals/em1110-1-1003/entire.pdf as of May 2007.

[122] R. Van der Merwe and E.A. Wan. The square-root unscented Kalman filter for state and parameter-estimation. In *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, volume 6, pages 3461–3464, May 2001.

[123] Mark Wade. Encyclopedia Astronautica. Available at http://www.astronautix.com/details/cos88662.htm as of May 2007.

[124] B. Wie and P. M. Barba. Quaternion feedback for spacecraft large angle maneuvers. *Journal of Guidance, Control, and Dynamics*, Vol. 8(Num. 3):pp 360–365, 1985.

[125] B. Wie, H. Weiss, and A. Arapostathis. Quaternion feedback regulator for spacecraft eigenaxis rotations. *Journal of Guidance, Control, and Dynamics*, Vol. 12(Num. 3):pp 375–380, 1989.

[126] Bong Wie. *Space Vehicle Dynamics and Control.* AIAA Education Series, Reston, VA, 1998.

[127] Marlin Williamson, Nick Johnston, Richard T. Howard, Drew P. Hall, Joseph Gaines, and Katherine Chavis. Automated rendezvous and docking operations evaluations. In Pejmun Motaghedi, editor, *Modeling, Simulation, and Verification of Space-based Systems III*, volume 6221, page 62210C. SPIE, 2006.

[128] E. Wilson, C. Lages, and R. Mah. Gyro-based maximum-likelihood thruster fault detection and identification. In *Proceedings of the 2002 American Control Conference*, volume 6, pages 4525–4530, May 2002.

[129] E. Wilson, C. Lages, and R. Mah. On-line gyro-based, mass-property identification for thruster-controlled spacecraft using recursive least squares. In *The 2002 45th Midwest Symposium on Circuits and Systems (MWSCAS-2002)*, volume 2, pages 334–337, August 2002.

[130] Edward Wilson, David W. Sutter, and Robert W. Mah. Motion-based, mass- and thruster-property identification for thruster-controlled spacecraft. In *AIAA Infotech@Aerospace Conference*, Arlington, VA, September 2005.

[131] Edward Wilson, David W. Sutter, and Robert W. Mah. Motion-based thruster fault detection and isolation. In *AIAA Infotech@Aerospace Conference*, Arlington, VA, September 2005.

[132] Steve Van Winkle. Advanced Video Guidance Sensor (AVGS) project summary. In Robert D. Habbit, Jr. and Peter Tchoryk, Jr., editors, *Spaceborne Sensors*, volume 5418, pages 10–20. SPIE, 2004.

[133] Trent Yang. Optimal thruster selection with robust estimation for formation flying. Master of science thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2003.

[134] Anatoly Zak. RussianSpaceWeb.com. Available at http://www.russianspaceweb.com/progress.html as of May 2007.

[135] Douglas Zimpfer, Peter Kachmar, and Seamus Tuohy. Autonomous rendezvous, capture and in-space assembly: Past, present and future. In *1st Space Explo-*

ration Conference: Continuing the Voyage of Discovery, number AIAA-2005-2523, Orlando, Florida, January 2005.