NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

LABORATORY EXPERIMENTATION OF AUTONOMOUS SPACECRAFT DOCKING USING COOPERATIVE VISION NAVIGATION

by

David A. Friedman

December 2005

Thesis Advisor:	Marcello Romano
Second Reader:	Vladimir Dobrokhodov

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2005	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Laboratory Experimentation of Autonomous Spacecraft Docking Using Cooperative Vision Navigation			**5. FUNDING NUMBERS**
6. AUTHOR(S) David A. Friedman			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory, Space Vehicles Directorate Kirtland AFB, NM			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

On-orbit, autonomous docking and spacecraft servicing are key areas of research in the defense and civil space communities. This thesis contributes to that effort by developing portions of a testbed and an experimental docking vehicle at the Spacecraft Robotics Laboratory of the Naval Postgraduate School.

The testbed was advanced by incorporating a large, flat epoxy surface and an indoor-GPS system into the laboratory framework. The epoxy floor allows a vehicle to emulate the space environment by floating on a near-frictionless surface representing motion in two dimensions. Pseudo-GPS was integrated into the testbed to allow for independent verification and validation of a vehicle's performance.

The docking simulator was developed by integrating computer hardware and attitude sensors into a newly-designed vehicle architecture to support its navigation and control needs. A position and attitude estimator was created to fuse the vehicle's sensor inputs. A control system was designed to allow for position control through eight thrusters and attitude control through the use of a reaction wheel.

Finally, experiments of proximity navigation were conducted. One experiment established the versatility of the vehicle's control system by performing a closed loop maneuver. A second experiment successfully demonstrated a complete docking scenario.

14. SUBJECT TERMS Autonomous Docking, Satellite Docking, Vision-based Navigation			**15. NUMBER OF PAGES** 99
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

LABORATORY EXPERIMENTATION OF AUTONOMOUS SPACECRAFT DOCKING USING COOPERATIVE VISION NAVIGATION

David A. Friedman
Captain, United States Air Force
B.S., Tulane University, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2005

Author: David Friedman

Approved by: Marcello Romano
Thesis Advisor

Vladimir Dobrokhodov
Second Reader/Co-Advisor

Anthony Healey
Chairman, Department of Mechanical and Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

On-orbit, autonomous docking and spacecraft servicing are key areas of research in the defense and civil space communities. This thesis contributes to that effort by developing portions of a testbed and an experimental docking vehicle at the Spacecraft Robotics Laboratory of the Naval Postgraduate School.

The testbed was advanced by incorporating a large, flat epoxy surface and an indoor-GPS system into the laboratory framework. The epoxy floor allows a vehicle to emulate the space environment by floating on a near-frictionless surface representing motion in two dimensions. Pseudo-GPS was integrated into the testbed to allow for independent verification and validation of a vehicle's performance.

The docking simulator was developed by integrating computer hardware and attitude sensors into a newly-designed vehicle architecture to support its navigation and control needs. A position and attitude estimator was created to fuse the vehicle's sensor inputs. A control system was designed to allow for position control through eight thrusters and attitude control through the use of a reaction wheel.

Finally, experiments of proximity navigation were conducted. One experiment established the versatility of the vehicle's control system by performing a closed loop maneuver. A second experiment successfully demonstrated a complete docking scenario.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author would like to acknowledge the financial support of the Air Force Research Laboratory, Space Vehicles Directorate.

The author would also like to thank the following people for their invaluable assistance in the completion of this thesis:

Dr. Marcello Romano and Dr. Vladimir Dobrokhodov - For their expertise, guidance, and patients throughout the entire thesis process.

Mr. Richard Howard – For allowing me unparalleled access into his laboratory and showing me the best of what NASA has to offer.

Mr. Mike Ramy – Who showed me everything I've ever wanted to know about epoxy floors…but was too afraid to ask.

LCDR Tracy Shay – For his friendship and partnership in completing the AUDASS II vehicle (and not dropping the ball).

And last, but not least…

My fiancé, Michelle, for listening to every story I have ever told her about those crazy, autonomously docking, battle robots, and for her love and support as we made it through our masters programs together.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Spacecraft docking and satellite servicing have been key areas of interest since the beginning of the space program. They combine aspects from Guidance, Navigation, and Control (GNC) research with robotics applications. These concepts were first experimented on-orbit in the 1960s during early manned space missions of the Gemini program. They were later expanded to such missions as the Hubble Space Telescope repair mission and the Space Station Remote Manipulator System, to name a few. However, in all of these missions, the execution required a man-in-the-loop.

Twenty-first century space programs have developed a need to further expand their docking and servicing capabilities for interplanetary travel and spacecraft servicing. With the cancellation of the Titan rocket program, the United States no longer has a heavy-lift capability that would allow a single vehicle to launch from Earth and travel to Mars (and beyond). Therefore, any interplanetary spacecraft would have to be broken down into separate components, launched on separate launch vehicle, and require on-orbit assembly through the use of autonomous docking technology. Furthermore, from a spacecraft servicing perspective, docking and robotic capabilities will be necessary for any on-orbit spacecraft requiring component replacement or propellant replenishment. The ability to perform these missions effectively will extend these spacecraft's mission life, saving millions of dollars in launching replacement spacecraft.

Adding autonomy to docking and servicing further enhances the usefulness of this technology. Man-in-the-loop systems cost many man-hours in training and execution time to perform tasks, not to mention exposing man to the dangerous environments of outer space. The Russian Soyuz capsule had/have the ability to autonomously dock with the Mir Space Station and International Space Stations. The U.S. Air Force, NASA, and Defense Advanced Research Projects Agency (DARPA) have all been investigating technologies that can conduct autonomous docking missions through the use of experimental small satellites. The Air Force Research Laboratory (AFRL) launched the XSS-11 in 2005 to experiment with proximity operations of a small satellite to an upper

stage of a Minatour I launch vehicle. NASA launched their Demonstration of Autonomous Rendezvous Technology (DART) in 2005 and had mixed results. DARPA is conducting the Orbital Express mission in 2006. DARPA is hoping to demonstrate a satellite being autonomously docked and refueled.

B. LITERATURE REVIEW

To further the technologies that will enable autonomous spacecraft docking to take place, several laboratories have developed on-the-ground experimental test-beds.

Stanford University's Aerospace Robotics Laboratory developed a testbed (see Figure 1) that is capable of capturing and manipulating free-floating objects in two-dimensions. This system is similar to the previous two systems in that it uses computer vision in the form of a CCD camera, 100 Hz control, and dual two-link manipulators. One unique capability of this testbed is that it takes advantage of a pseudo-GPS signal by using an offline "Global Vision System" that translates the vehicle's image into laboratory position. (Ref. [1])

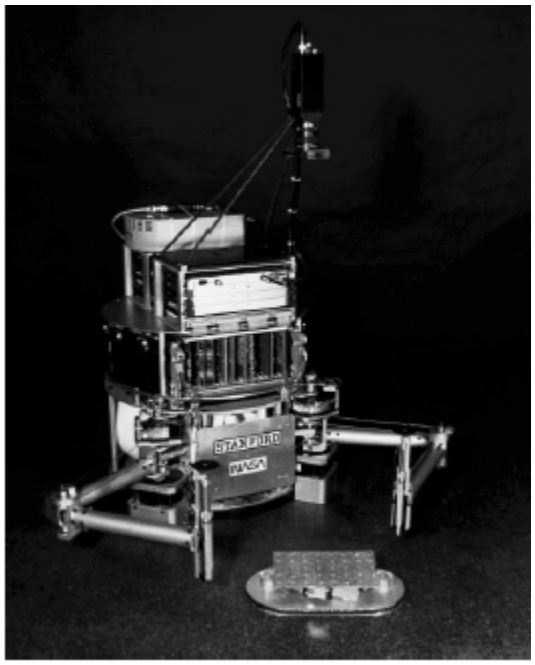

Figure 1. Stanford University's Free-Flying Space Robot (From Ref. [2])

Another testbed performing similar research is the Experimental Free-Flyer (EFF) (see Figure 2) robotics project developed by the University of Padova (Italy). It is an autonomous robot (although connected by an umbilical cord) that utilizes a large robotic arm used for grappling in support of a simulated space servicing mission. It floats on air-bearings to create a three degrees-of-freedom environment. It utilizes a video camera to measure relative position and attitude. (Ref. [3])

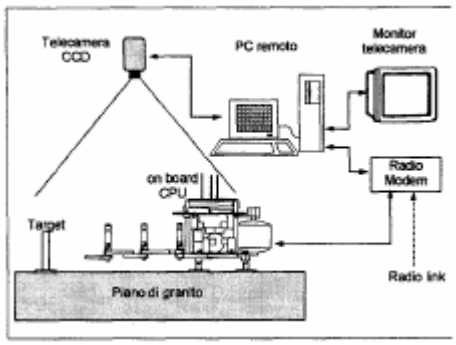

Figure 2. Experimental Free-Flyer Schematic (From Ref. [3])

The Astronaut Reference Flying Robot (ARFR), built by Japan's MITI Electrotechincal Laboratory (see Figure 3) is a laboratory version of flying telerobotics system. The research could potentially replace the work of an astronaut who performs extravehicular activities using a manned maneuvering unit. The particular task the researchers focus on is satellite capture. Like the EFF, the AFRF also operates on flat floor, floating on a cushion of air through the use of air bearings. It utilizes a capturing mechanism with two flexible manipulators complete with proximity sensors on the end of each arm. The vehicle uses thrusters for propulsion and attitude control. It also utilizes two cameras for stereo computer vision. (Ref. [4])

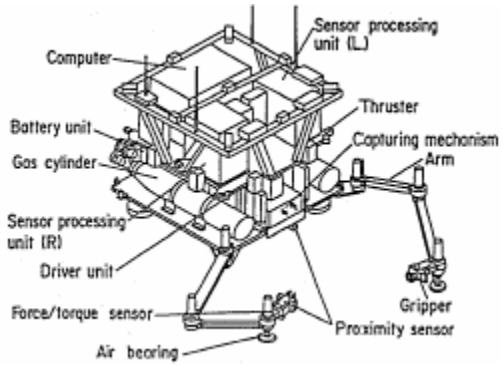

Figure 3. Astronaut Reference Flying Robot (From Ref. [4])

NASA Marshall Space Flight Center (MSFC) Flight Robotics Laboratory testbed has a large, 3800 square foot surface. With the floor is a matching 3000 pound airsled (see Figure 4) that floats on three large air-bearings; about 20 times larger than the previous examples. They use a laser-based vision system that is designed to receive a return signature from a cornercube device mounted on the target. Propulsion and attitude control is accomplished through the use of 16 thrusters. (Ref. [5])

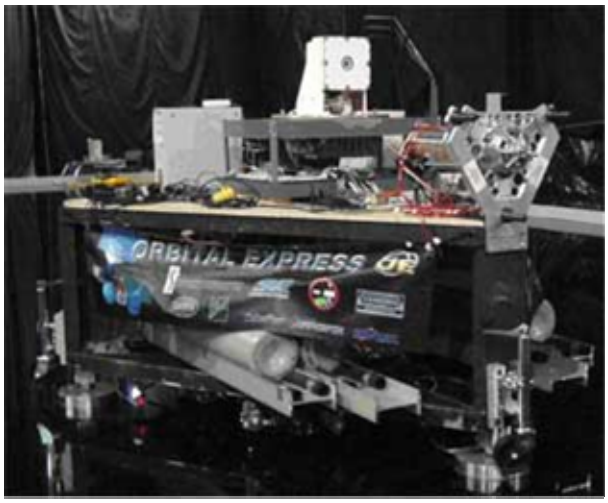

Figure 4. MSFC Flight Robotics Laboratory Airsled (From Ref. [6])

The Naval Postgraduate School's Planar Autonomous Docking Simulator (NPADS) (renamed Autonomous Docking and Servicing Spacecraft (AUDASS) in 2004 is the predecessor to the SRL's research effort (see Figure 5). It has eight thrusters for propulsion, four air bearings to float on an eight by six foot granite table, a reaction wheel for attitude control, and a camera, looking at features on the laboratory ceiling, to obtain position. Unlike the previous examples, this model did not integrate a docking mechanism into the unit which limits its usefulness in autonomous docking research. Further, the vehicle's single structure design makes it difficult to build in upgrades.

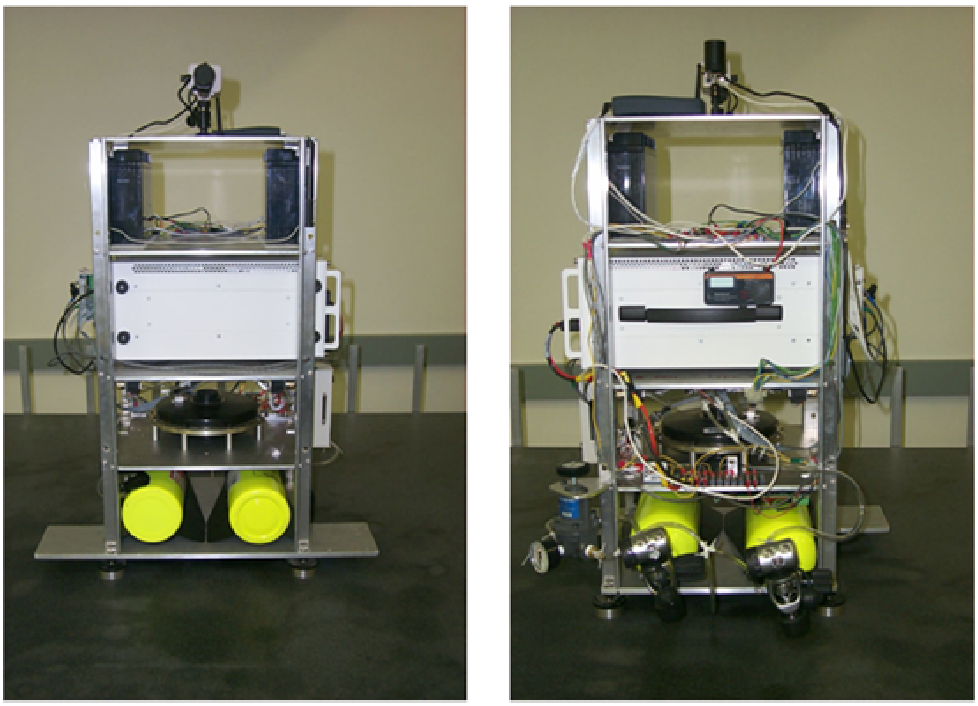

Figure 5. NPS Planar Autonomous Docking Simulator (From Ref. [7])

C. SCOPE OF THESIS

This thesis comprises the work involved in refurbishing the Spacecraft Robotics Laboratory, integrating computer and sensor components onto the AUDASS II chaser vehicle, and developing a Kalman filter, for fusing sensor data, as well as a control system to allow the chaser vehicle to perform a docking maneuver. It builds upon the work completed previously in developing the original AUDASS vehicle and its supporting control system. It is a concurrent work performed in parallel with a thesis devoted to the smart and modular construction of the AUDASS II. This work is an intermediate phase in the evolution of the Space Robotics Lab with an end vision of being a facilitator in testing new technologies and GNC systems that support autonomous docking and other spacecraft maneuvers.

The work of this thesis was completed throughout the 2005 calendar year. It started with a close examination of possible supporting architecture existing in the commercial environment that would enhance the laboratory's ability to simulate the space environment. From that, an epoxy surface was integrated into the laboratory and indoor GPS technologies were selected for future laboratory use. Those efforts are

examined in Chapter II. The next phase of the work involved researching, selecting, and ultimately integrating embedded computer technologies and inertial sensors into the framework of the AUDASS II. From that, the vision computer, the control computer, and an inertial measurement unit (IMU) are fully discussed in Chapter III. The third phase of the thesis involved developing a Kalman filter that would merge the IMU data with the computer vision data to accurately estimate position and orientation of the chaser vehicle with respect to the target vehicle. This work is shown in Chapter IV. The thesis culminates in integrating and modifying the pre-existing control from the original AUDASS into the new model to perform an actual docking maneuver. The results from those tests can be seen in Chapter V.

II. LABORATORY SETUP

A. INTRODUCTION

The Spacecraft Robotics Laboratory (SRL) is a research laboratory of the Naval Postgraduate School, founded in February 2004, and is still under development. Beginning in July 2004, the laboratory testbed consisted of an eight foot by six foot granite table, a single vehicle that operated on the table, and a remote computer with which to build and transmit software to the vehicle. Through efforts conducted primarily during the 2005 fiscal year, the laboratory was upgraded significantly. The granite table was replaced by a 16 foot by 14 foot epoxy floor leveled within 0.003 inches end-to-end. An "Indoor Global Positioning System (GPS)" system was added to provide absolute positioning up to one millimeter accuracy within the laboratory environment. This chapter will cover the details of these two new facilities.

B. EPOXY FLOOR

Beginning in July 2004, the laboratory testbed consisted of a granite table, eight feet by six feet long and the vehicles which maneuvered on it. The benefit of having this table was twofold: first, the table was highly smooth so that the air bearings, and thus the vehicle above them, would have a near friction-free surface on which to float. The second benefit was that the granite table was level and could be recalibrated via adjustable screws when needed. This equipment allowed the laboratory to perform some initial testing with the chaser vehicle.

Although the granite table's features initially enabled the laboratory to perform its work, its relatively small size eventually became the limiting factor in advancing the work. The table's dimensions were too small to perform a realistic docking simulation where all three degrees-of-freedom could be adequately exercised. In addition, the table's support structure had been partially damaged by a flood in the laboratory several years earlier. To counteract this, the table required supplement support in the form of a jack placed near the location of the damaged wheel. Since the jack was supporting a

cross-beam and not the corner, this made the support structure slightly unreliable as a flat surface due to some bending. This ultimately limited our ability to manually level the table.

To correct for the granite floor's deficiencies, research was conducted in finding a larger, more reliably flat surface. A solution presented itself in the form of epoxy surfaces. Epoxy surfaces begin in the form of a liquid and are mixed together with a resin hardener. These materials, when used properly, will form into a hard surface that lays level within 3 thousandths of an inch over the surface of the floor. The fluidity of the material has the benefit of being able to form to the contours of our laboratory, whereas granite tables are set in size and may not properly fit within the SRL's confines. Further and more importantly, the floor does not need to be brought into the lab's narrow 6.5 foot entryway in one piece. Given the constraints of the laboratory, a floor of 15 feet by 15 feet, or 225 square feet, was initially estimated as an appropriate size for the floor space available. The usable experimental area would increase by about 4.5 times of the granite table. This was deemed a suitable solution.

Several vendors were identified who sold and/or installed epoxy floors. Based on price of materials plus installation cost, Precision Epoxy Products out of Douglasville, GA. was selected. The company had ample experience installing similar level floors in NASCAR and Indy Car auto racing garages around the United States. In addition, they make all of their epoxy mixtures in-house so that the SRL only dealt with a single vendor, which in retrospect, limited the final cost of the floor. A sample of their work can be seen in Figure 6. (Ref. [8])

Figure 6. Typical Epoxy Floor (From Ref. [8])

Precision Epoxy Products required a one-foot perimeter on all sides of the floor to allow them room to maneuver during the construction and installation of the floor. This

limited the width the floor to 14 feet. To counter the shrinking area, the length of the floor was extended to 16 feet. The final dimensions of the floor were 16 x 14 feet, or 224 square feet; only 1 square foot smaller than initially planned. The SRL also decided that to allow for computer vision projects to determine vehicle positioning in the future (not covered in this thesis), a decal the size of the floor would be applied below the surface of the floor and shown through the clear epoxy coating. The decal would consist of a multicolor grid pattern in one centimeter blocks, blue lines spanning the length of the floor and red lines spanning the width.

This floor design employed several other features that are not typically performed on other flat floors. This floor is designed with a 16 inch tall barrier that surrounds the surface. The barrier also comes with a gate that can be opened up to slide new equipment onto the floor. At the base of the gate is a shallow-grade ramp connecting the tiled floor of the lab to the epoxy floor.

1. Epoxy Floor Installation Timeline

The vendor installed the floor over a period of ten days during July 2005. Appendix A shows a copy of their standard four-day timeline of tasks required to install a standard floor. Table 1 shows the timeline that was actually required to install the floor in the SRL.

The contractors spent most of Day 1 unloading the equipment from their truck to the basement of the building where the SRL lab is located. The several thousand pounds of material included a fork lift device used to heat and mix the epoxy, three 45-gallon drums filled with epoxy, seven 8 foot and 7 foot aluminum pieces used as the border, and about a dozen specialized machines used to prepare the surface and epoxy mixtures.

Once their equipment was unloaded, the contractors dug out the tiled floor of the lab where they would eventually pour the floor. To smooth out the floor, they used a converted floor-buffer with a sanding attachment. This eliminated the residual glue used to hold down the tile. Upon further investigation of the bare cement floor, the contractors discovered that there was a deep-rooted crack in the cement. Interestingly enough, the crack spans much of the entire basement of this building. The contractors speculated that

this was an expansion crack formed when the building was first constructed. Because it was not currently expanding, they decided to control it by laying bandage-like material along the crack and poured primer into the crack to seal it up. Although the crack's effect on the floor may not be realized for several years, chances are that these precautionary measures will have given the floor years of additional use.

Day 0	22 July	Orientation Meeting
Day 1	23 July	Unloaded Equipment, Clear out Tiled Area, Roll out Primer
Day 2	24 July	Construct Outer Edge
Day 3	25 July	Lay Down Base Coat
Day 4	26 July	Lay Copper Strips for Grounding, Pour First Color Coat
Day 5	27 July	Pour Second Color Coat
Day 6	28 July	Install Ramp
Day 7	29 July	Lay Down Decal
Day 8	30 July	Pour Clear Coat
Day 9	31 July	Off Day, Allow Floor to Harden
Day 10	1 August	Clean Floor, Fix Air Gaps, Final Checkout, Pack Up Equipment

Table 1 Timeline of Events for Epoxy Floor Installation

Day 1's activities were concluded by laying the same primer used for the crack down on the remainder of the floor with the use of paint rollers. Again the primer sealed the porous concrete off from any air pockets. Photos of Day 1 can be seen in Figure 7 to Figure 10.

Figure 7. Bare Floor Prior to Installation in the SRL Lab

Figure 8. Floor Tile Removal

Figure 9. Primer Application

Figure 10. Containment of the Crack in the Cement Floor

On Day 2, the contractors focused on constructing the border around the floor. The border will act as a boundary for the epoxy when it is poured, a barrier for laboratory staff and visitors to prevent from accidentally stepping onto the surface, and as wall to prevent the target and chaser vehicle from falling off the epoxy surface. To construct the border, the contractors used four eight-foot and three seven-foot lengths of precut

aluminum 13 inches high. A four-foot aluminum door was cut as well to allow easy access on to the surface. The prepared floor surrounded by its newly constructed boundary can be seen in Figure 11.

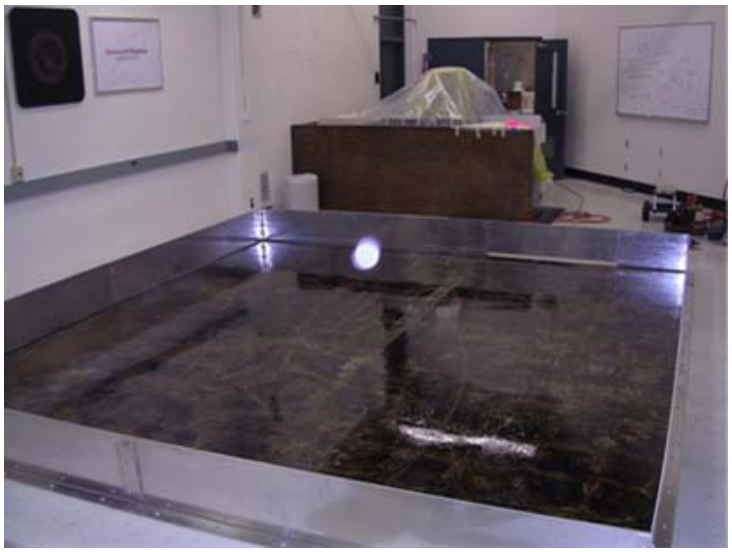

Figure 11. Floor with Border

The next step of the process was to lay down the base coat of epoxy. This happened on Day 3. In preparing any of the coats for pouring, the epoxy must be heated up prior to use. The contractors used heaters that strap on to the base of the barrel containing epoxy. It takes on average an hour to heat up a full barrel of epoxy. To distribute the heat evenly, the epoxy is mixed up with a mixing device.

Once the epoxy reaches the preset temperature, it is poured into small 5-gallon containers where it awaits the hardener. The number of containers used depends on how flat the floor is prior to the poor. Flatter surface will spread the epoxy more evenly and require fewer coats. Once the hardener is added to the epoxy, the contractors had four minutes to mix up the mixture and pour it onto the floor. After pouring, the mixture was let alone to dry for 24 hours. Photos of the contractors pouring the epoxy (from the second coat) can be seen in Figure 12 Figure 12. and the complete first coast can be seen in Figure 13.

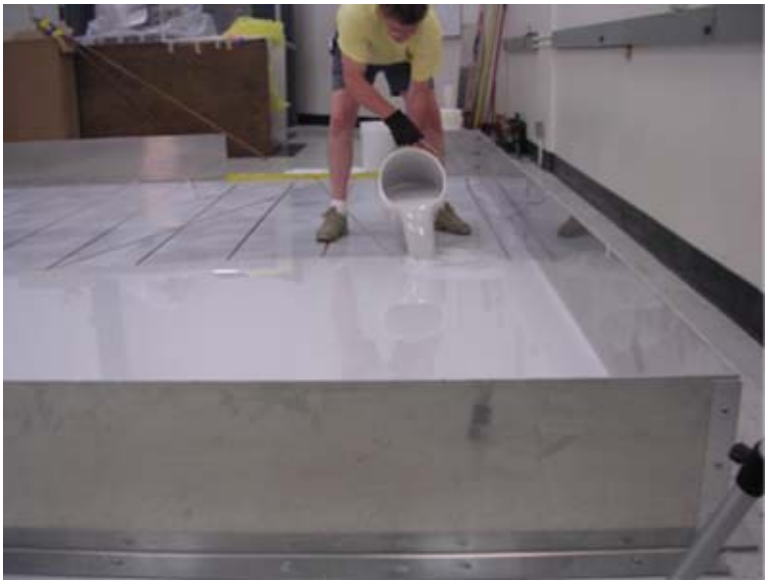

Figure 12. Liquid Epoxy Poured onto the Floor

Figure 13. Floor with First Coat of Epoxy

On Day 4, the results of the previous day's pour could already be seen. With the contractors precise leveler, capable of detecting deviation of five ten-thousandths of an inch, the floor already displayed areas that met specifications within the 3 thousandths of an inch from foot-to-foot.

However, not all areas of the floor met the final specifications. Additional steps were still required to complete the floor. The next step was to sand the floor so that the

second coat of epoxy would bond the base coat. Also flat copper strips were laid crosswise along the floor to prevent static electricity from building up.

The copper strips were grounded into the bare cement of the floor below were the ramp would eventually rest. This setup can be seen in Figure 14. In previous floors, there has been a noted problem with static electricity shocking mechanics using the lever floor in auto racing garages, especially within the floor's first month of use. With the SRL's computer-laden vehicles, static electricity could have the potential of causing major setbacks in research since the electronic could be vulnerable to short circuiting. This grounding technique has been proven to be effective on previous floors.

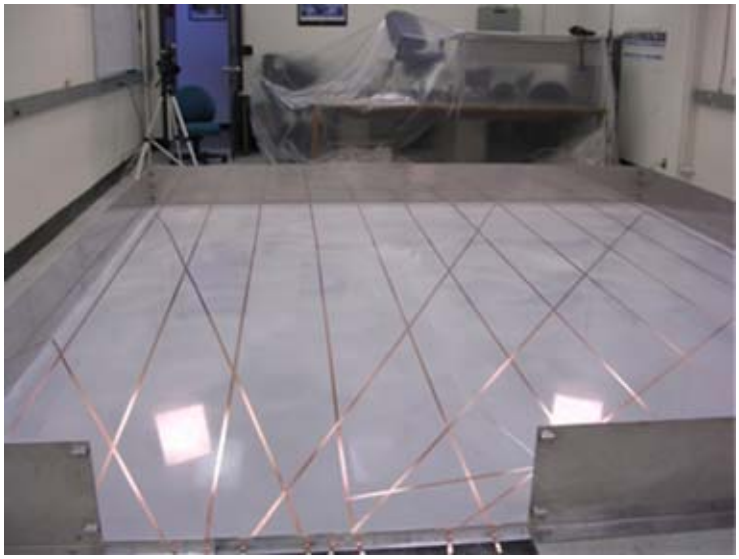

Figure 14. Copper Strips Applied to the Floor for Proper Grounding

The final step of the day was to pour the second coat, a color coat, onto the floor. The second coat was mixed and poured in the same manner as the base coat.

Day 5 was a setback day for the contractors. After waiting about 24 hours for the second coat to dry, they realized that the floor was not completely level with specifications. While much of the floor was level, there was one corner that exhibited areas out of specification. Since the decal and the final clear coat do no contribute the levelness of the floor, the color coat must meet the preset specification prior to proceeding. Because of this, a second color coat was mixed and poured and let to dry. This is the third coat of epoxy on the floor.

Day 6 was used to install the ramp. The ramp is about 44 inches wide and 5 inches deep. It was installed to allow the laboratory's staff to place the vehicles and other equipment onto the floor with ease. The ramp is also used as a stopper for when the final clear coat is poured. Its porous material easily absorbed overflow epoxy.

Day 7 was devoted to laying down the four, 16 foot by 4 foot decals on to the floor. The contractors have not previously applied such a large decal to any surface. Not only did all four sections need to be aligned straight both horizontally and vertically in accordance with the grid, but it also needed to be laid flat on the epoxy without the presence of any air-bubbles. The end result of the decal can be seen below in Figure 15.

Figure 15. Decal as Seen During Installation

The clear coat, the top surface of the epoxy floor, was laid on Day 8. The same procedure for preparing and applying this final coat was used as the previous three applications. The only notable difference, other than the color, was that the pour had to be 3/16 inches deep because of the floor design. Because the floor was measured after the second color coat to be flat within specifications, the contractors could measure and apply the proper amount of epoxy with confidence.

No work was accomplished on Day 9 so that floor could begin the curing process without any external interference.

On the final day of the installation, the finishing touches were applied. A vinyl safety trim was applied to the rim of the border. The floor was polished to a mirror shine.

Of interesting note, two air bubbles did pop to the surface. While measures are taken to prevent such occurrences, the bubbles are not too uncommon. To remedy the situation, a small hole is drilled into the floor at the location of the bubble. Then, a small amount of epoxy is mixed and injected into the hole to seal it up. The syringe used to perform the injection can be seen in Figure 16. The final product can be seen with original chaser vehicle in Figure.

Figure 16. Syringe Used to Fill in Air Bubbles in Floor

Figure 17. Final Product of Epoxy Floor with AUDASS

2. Epoxy Floor Performance

The most important thing to note is that despite the fact that the floor is within specifications, the vehicles exhibits some drifting towards the back corner of the floor. This is the same corner that caused the floor to require an extra coat of epoxy. Control algorithms used with autonomous docking should be able to account for this disturbance.

A second area that limits the floor's effectiveness is the cleanliness of the surface. Despite measure taken by the staff to reduce dust particles in the lab through the use sticky mats (which remove dirt from shoes) and through closing off all vents to the lab's exterior, layers of dust do build up on the floor and require frequent and thorough cleanings.

Thorough cleanings are to be accomplished through the use of a dry mop only. Initially, the contractors instructed the staff to use a silicon-based lubricant to help reduce the friction between the floor and the vehicles. Using this lubricant caused permanent damage to one set of air bearings by clogging their air-flow. The air-bearings perform much better without the use of lubricant.

C. INDOOR GPS

An important aspect of conducting research in autonomous docking is finding an independent means to validate the performance of the vehicles. In dealing with the guidance and control system, performance can measured through the use of an absolute positioning reference. When considering such a measuring system, Global Positioning System (GPS) satellites first come to mind. GPS receivers are readily available and relatively inexpensive. Unfortunately, its accuracy, on the order of meters, is not conducive to a laboratory environment where a positioning accuracy of five millimeters or better is essential for position measurements. Moreover the location of the SRL in the basement of a concrete building precludes it from receiving RF signals from orbiting satellites.

Research was conducted to find alternative methods to gain laboratory positioning data. The first area looked into was ultrasound based systems. In such a system, ultrasound devices placed around the room would emulate the GPS satellites. A receiver

placed on the vehicle would receive the ultrasound and based on basic triangulation equations, could calculate its position. This appeared to be a fairly inexpensive solution. However, during initial testing of the equipment, it was found that the thrusters onboard the vehicles provided far too much noise interference for the ultrasound signals to find a positioning solution. Since thrusters are intended to be used throughout the vehicle's operation, this proved to be an unacceptable solution.

A more costly laser-based system was then chosen as a primary means for laboratory positioning. The system was designed by and purchased from Arc Second, Inc. (http://www.arcsecond.com). The eye-safe Class 1 laser used by this system could provide a positioning solution without experiencing any degradation due to the normal operations of the vehicles. Further, since Class 1 lasers are used by the transmitters, safety for the laboratory staff was not compromised.

1. Laser GPS Theory of Operations

The idea behind Indoor GPS is fairly similar to real GPS. The underlying concept behind both has to do with reverse triangulation based on known references. The reference signals, in this case, come from transmitters that act as GPS pseudolites. Whereas in satellite GPS, 24 satellites are required for global coverage, the size and shape of the environment plus the user's accuracy requirements determine the number of transmitters required for a given laboratory setup. (Ref. [9])

For indoor GPS, each transmitters emits two beams vertically, spaced 90 degrees apart, with a fan of +/- 30 degrees from the horizontal plane. The beams are emitted from a rotating head. Each transmitter's head rotates at a unique speed so that each receiver can uniquely identify which transmitter is broadcasting which signal. A third beam is transmitted every other rotation to provide a timing signal to the receivers. (Ref. [9])

Each sensor and receiver pair lies within the volume illuminated by the transmitters. The sensor detects laser signals within its range and converts it to data which is then processed by the receiver. The receiver measures two things: the interval time between each strike of the laser and the interval time between laser strike and strobe. A combination of the interval times and a priori knowledge of the transmitters leads to

solving for the angle between the transmitter, receiver, and the horizontal plane. This leads to solving for the azimuth and elevation angle between the receiver and the transmitter. (Ref. [9])

While knowing the azimuth and elevation angles for multiple transmitters directly leads to triangulating the receiver's position, it is not the only requirement for determining a position solution. The spacing of the transmitters is of equal importance. Similar to satellite GPS's dilution-of-precision concept (where ideal spacing is three satellites 60 degrees apart at the horizon and one satellite directly overhead), a two transmitter configuration has an ideal convergence angle of 90 degrees and should be somewhere between 60 and 120 degrees for functionality. When the transmitters do not fall in this range the uncertainty of the triangulation will tend to increase and induce positioning errors. (Ref. [9])

A calibration is also required to obtain a positioning solution. In order for real time-measurements to produce a position, the software needs to determine where the transmitters are located. It accomplishes this by the user taking fixed measurements around the works space. Ideally, six measurements should be taken in a cubical environment, such as the SRL. Four measurements are taken in the corners, and two measurements are taken in the center with the aid of a scale bar. A scale bar is essentially a fixed yardstick with a tap on either end to screw in the receiver sensor. This allows for two measurements to be taken with a known constraint. The combination of these data allows for the software to calculate a bundle, essentially obtaining a fix on the transmitters. The mathematics by this is no different than an over-determined, least squares linear algebra problem. (Ref. [9])

The final concept that needs to be touched before obtaining a positioning solution is scale. While the receivers interpret the transmitter signals into angles, there is nothing for the receiver to judge the volume of the space in which it exists. During the calibration process described above, the constraint must be placed on at least two of the measurements taken so that they are a fixed and predetermined distance apart from each other. Essentially, this adds a scale to the volume that was already created. Without

scale, the receivers would know their relative angles to the transmitters, but would not know how far away they were. (Ref. [9])

Adding this scale to the calibration is the final step in the theory to gain three-dimensional position. A more detailed look at GPS can be seen the Indoor GPS Work Space User's Guide Version 6.0. (Ref. [9])

2. Laser GPS Equipment

To achieve positioning within the SRL, the two transmitters were purchased; one model was an ATX and the other was a TX. The TX model is pictured below in Figure 18. The two transmitters are identical with a couple exceptions. The ATX has self leveling device and additional controls on the operating panel to control its leveler. Each unit weighs approximately 10 pounds, plus the weight of its battery pack. Each transmitter has an effective range of 50 meters.

Figure 18. Indoor GPS TX Transmitter

Two receiver/sensor packages were also purchased. One of them is pictured in Figure 19. The receiver is the top box on the left of the picture. Below the receiver is the rechargeable battery pack. The small black and silver cylinder on the far right of the picture is the sensor. The base of the sensor has a metric M8 thread so to allow it to be

screwed into the sensor deck of a vehicle. The long cylinder pictured between the receiver and the sensor translates the raw sensor signal into data that the receiver can work with. Finally the receiver is connected to a PC through serial RS232 cable which extends from the receiver.

The receiver/sensor pair has a few nuances that must be complied with for a successful operation. The receivers must be at least ten feet away from a transmitter to detect the signal or the receiver will not recognize the laser signals emitted by the transmitter. The sensor must also have a line-of-sight to the transmitters. Each receiver operates with the "Black Sun" software package provided by Arc Second. The receivers have the ability to be reprogrammed by a computer loaded with Arc Second's Work Space software. This could prove useful if the manufacturer provides a software upgrade.

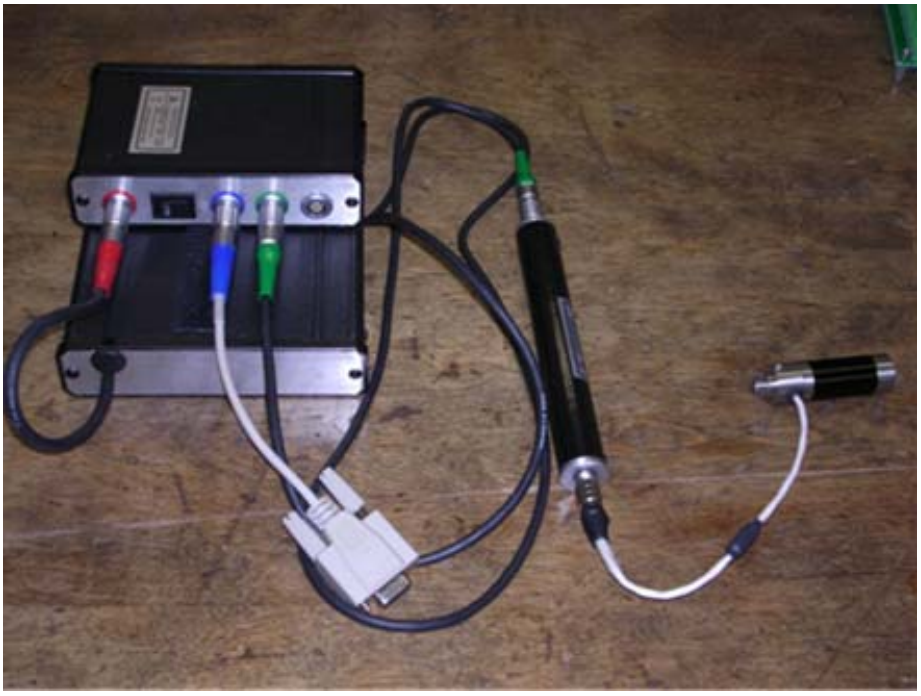

Figure 19. Indoor GPS Sensor and Receiver Equipment

The Work Space Version 6.0.12.1 software was provided by Arc Second and runs off a Windows-based operating system. It is the visual interface between the user and the receiver.

3. Laser GPS Laboratory Setup and Operations

The entire Laser GPS system is self-contained within the SRL. A schematic of the lab, seen in Figure 20, shows the interaction between the GPS equipment and the laboratory testbed. The transmitters are mounted on a wall, approximately eight feet above the floor, and aimed at the center of the epoxy floor workspace. One receiver is mounted on the electronics deck of the chaser vehicle. The sensor is connected to the receiver and mounted on the sensor deck on the roof of the chaser vehicle. The receiver is plugged directly into the serial port of the vision computer PC/104. The vision computer is remotely controlled through a VNC software connection on the off-line computer. The positioning data produced by the software is then sent via TCP/IP connection to the control PC/104 where it may be used either in control algorithms or as a validation measure.

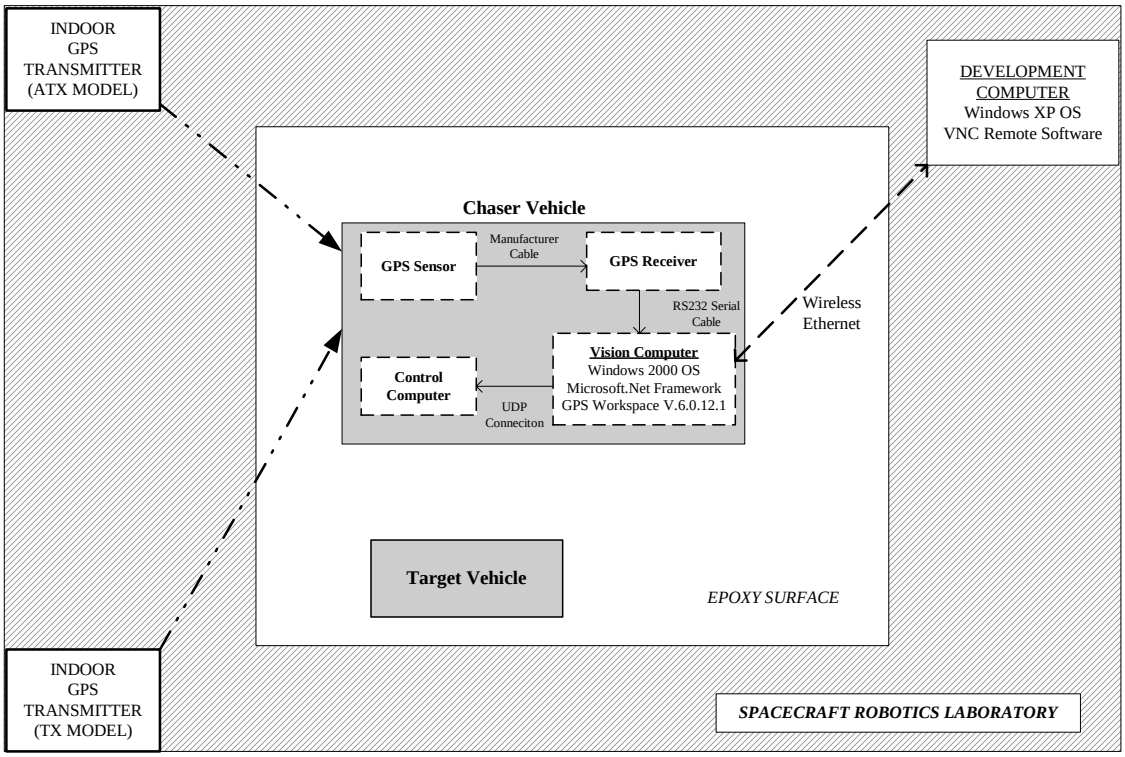

Figure 20. Schematic of the Indoor GPS in Relation to the SRL Testbed

The operational setup and calibration was accomplished with the aid of Arc Second's on-site training and the Indoor GPS Work Space User Guide Version 6.0. A

detailed procedure was written to describe the configuration process of the software and to describe the calibration process. This procedure is explained in Appendix B.

Several concepts described in the appendix worth mentioning here are receiver sensitivity, noise floor, and multipath. These three concepts all affect the sensor's ability to receive a quality signal, especially during the calibration procedure, where these factors can adversely affect the outcome of an attempted bundle. Receiver sensitivity and noise floor are quantitative values that can be adjusted on the control panel in the software. As they are adjusted, the standard deviation metric of the angular error will be directly affected. If standard deviation is greater that 50 microradians after at least 200 samples, the software will not and should not accept the measurement. There is no set formula for a proper adjustment; trial and error is the best means to gain a feel for how each parameter will effect a measurement. One thing to note is that fluorescent lighting will adversely affect measurements. Especially when performing a calibration, it is best to minimize the fluorescent lighting as much as possible.

The third concept that effects measurements is multipath. Multipath are reflections of the main signal that can be misconstrued as the intended signal. While it is difficult to get a feel for it in a quantitative sense, qualitatively it can bring about bogus results to a measurement. In the SRL, multipath has the potential for being a problem mostly due to its very flat, smooth epoxy floor. In fact, any flat, mirror like surface is a good conductor for multipath. In practice, the GPS sensors avoid multipath signals by being placed in center of the sensor deck of the vehicles and by being mounted on matted surface that will block multipath of the sensor deck.

III. TESTBED AND CHASER VEHICLE

A. OVERALL TESTBED

The SRL consists of a chaser vehicle, a target vehicle, a software development computer, a flat, epoxy floor, and two indoor GPS transmitters. To give the reader an overall sense of the setup, Figure 21 shows a photograph of the lab and Figure 22 displays the schematic of the laboratory's hardware.

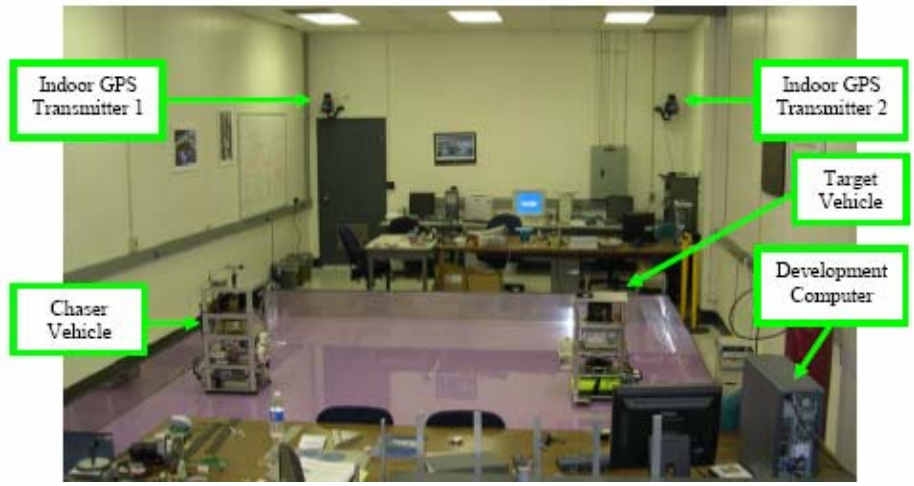

Figure 21. Spacecraft Robotics Laboratory

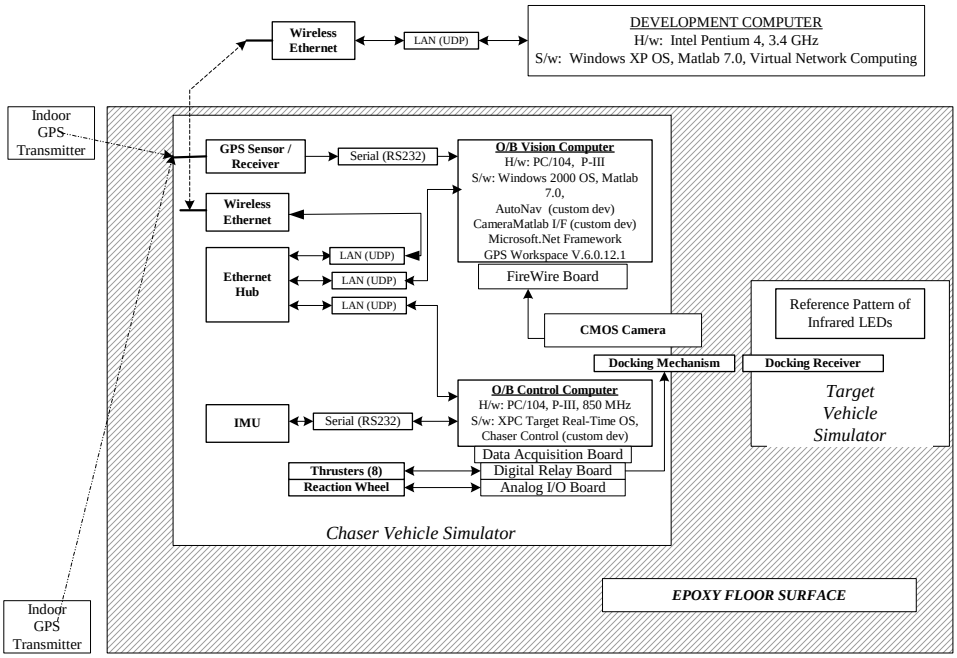

Figure 22. Testbed Schematic

B. CHASER VEHICLE OVERVIEW

The Autonomous Docking and Servicing Simulator II (AUDASS II) chaser vehicle was constructed in 2005 and replaced the legacy AUDASS as the SRL's primary vehicle for performing autonomous docking research. The older vehicle was converted into the target vehicle in the docking simulation. The primary functional difference between the two vehicles is that the AUDASS II has a single mechanism integrated into its structure that is capable of performing docking maneuvers and fluid transfer. While the original AUDASS used a camera looking at the laboratory ceiling to obtain position, the new version uses computer vision to obtain relative position. Beyond function, the new model is more elegantly designed, incorporating a modular approach, one deck stacked on top of the next. Each of these decks has a unique function to support the overall vehicle and can be removed and upgraded without affecting the structural design or the functionality of the vehicle as a whole. The older vehicle, while having multiple decks, did not employ a modular design. Upgrades to the older vehicle were made on a space-available basis. Throughout the design of the AUDASS II, each component was examined thoroughly and upgraded with superior technology. This chapter will touch on the overall design of the vehicle, while delving into the integration of specific hardware and computer components into the overall function of the vehicle. Another thesis (Ref. [10]) thoroughly covers the design and construction of the AUDASS II.

As stated before, the vehicle was constructed one deck at a time, with each deck having its own unique function. The base deck contains the thrusters, air bearings, compressed air, and its support structure. The docking deck contains the docking mechanism. The reaction wheel deck contains the reaction wheel and its support structure. The electronics deck contains a vast majority of the on-board electronics, including batteries, computers, digital and analog input/output (I/O) boards, Inertial Measurement Unit (IMU) and the Indoor GPS receiver/battery pack. On the roof of the vehicle is the sensor deck, where the GPS sensor, Complimentary Metal-Oxide Semiconductor (CMOS) camera, and wireless Ethernet hub all sit. Each deck was created out of aluminum frames for the support structure and sheet aluminum for the deck floor. The complete AUDASS II vehicle can be seen below in Figure 23. Table 2 details the main characteristics of the vehicle and its components.

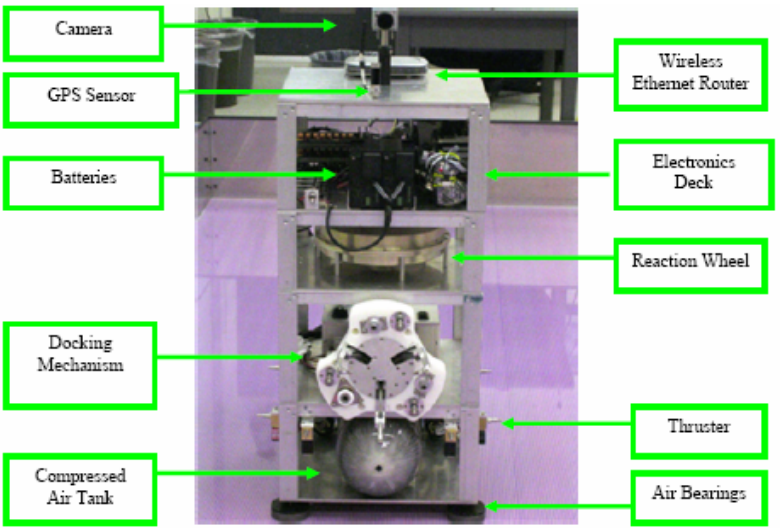

Figure 23. AUDASS II Chaser Vehicle

	Parameter	Value
Physical Size	Length [cm]	40
	Width [cm]	40
	Height [cm]	85
	Mass [Kg]	62.9
Propulsion	Thrusters Type	Cold-Gas
	Propellant	Air or Nitrogen
	Storage Capacity [L]	2460 @ 1 ATM
	Operating Pressure [Atm]	3.4 – 6.8
	Continuous Operation [min]	~ 20
	Thrust of each thrusters [N]	1
	RW Max Torque [Nm]	0.1624
	RW Max Ang. Mom. [Nms]	20.3
Electrical System	Battery Type	Lithium-Ion
	Storage Capacity	12 Ah @ 28Vdc
	Continuous Operation	~ 6 h
	Regulated Voltages	5, 18, 20, 24 Vdc
Vision and Control Computers System	Computers	PCI/PC104
	Processors	Pentium III
	Operating Systems	Win2000 XPC Target
	Input/Output Cards	Firewire, A/D
Sensors	IMU	
	Indoor GPS	
	Vision Sensor	

Table 2 Main Characteristics of the AUDASS II

1. Base Deck

The base deck contains all the components that allow the AUDASS II to move along the epoxy floor (see Figure 24). The compressed air tank is made of carbon fiber, instead of aluminum, to minimize the weight of the system. It can hold up to 65 liters at standard pressure. The tank has the ability to hold up to 4500 pounds per square inch (PSI) of compressed air.

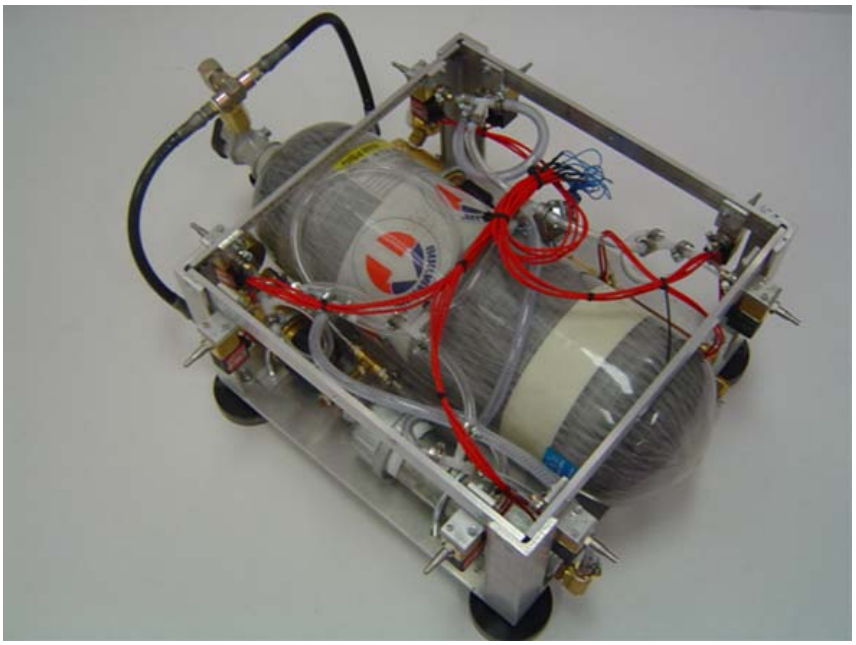

Figure 24. Base Deck

There are eight thrusters placed around the vehicle to propel the vehicle. Each has the ability to provide 1.1 Newtons of thrust. Each side of the vehicle has two thrusters identically spaced apart. The thruster's flow is controlled by a solenoid attached directly to the body of the thruster. The solenoids are regulated by the digital I/O board, which in turn is controlled by the control computer.

There are four air bearings placed at the base of the vehicle. The air bearings allow the vehicle to float over the epoxy table on a thin cushion of air. Each bearing is about 55 mm in diameter. The air bearings operate independently from the remainder of the vehicle.

The flow of air from the compressed air tank is split between the thrusters and air bearings. Each pathway is controlled by an adjustable regulator, capable of providing 0-400 PSI of airflow. The thrusters are nominally set at 90 PSI. The air bearings are nominally set at 30 PSI.

2. Docking Deck

The docking deck contains the docking mechanism together with its control box (see Figure 25). The docking mechanism and control box were developed by the Starsys Corporation and is a miniature prototype, 1/5 the size of the unit that will fly on Defense Advanced Research Projects Agency's (DARPA) Orbital Express mission scheduled for 2006. The docking mechanism's grappler is a three-armed manipulator that is driven by a worm gear. The grappler is designed to mate with the receiver, a passive body that is mounted to the target vehicle. The docking mechanism contains a nozzle that links to the receiver and allow fluid transfer to be possible. (Ref. [11])

The control box, while designed by Starsys, was modified in the SRL so that it could be integrated effectively with the AUDASS II. The control box receives its power from the lithium batteries, which in turn provides power to the grappler. The box has a variable voltage switch, which determines how fast the grappler will deploy and retract. The voltage is set manually and not controlled by the control computer's analog I/O board (although the potential exists of integrating that capability should the need arise). The control computer determines when to engage the grappler. To allow for this, the box was modified to take input from the digital I/O board.

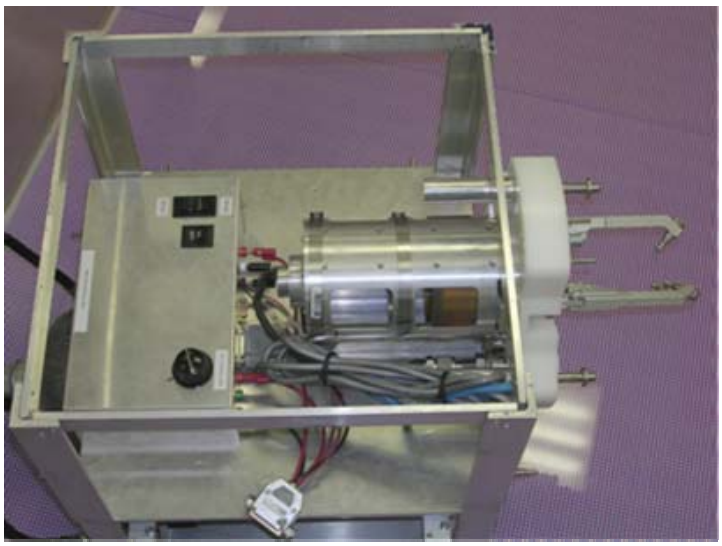

Figure 25. Docking Deck

3. Reaction Wheel Deck

The reaction wheel deck (see Figure 26) contains a reaction wheel and a voltage clamp. The reaction wheel is used for attitude control of the AUDASS II and is an upgrade from the legacy vehicle, which used thrusters for both propulsion and control. The reaction wheel, manufactured by Ball Aerospace, provides 20.3 Newton-meter-seconds of angular momentum storage and has a maximum torque of 0.2 Newton-meters. (Ref. [12]) The reaction wheel is connected, via the voltage clamp, to the analog I/O board where it will be send telemetry data and receive commands. The wheel is powered by the lithium batteries. The voltage clamp was built to prevent back-emf from flowing back into the control computer's circuitry and destroying the electronics.

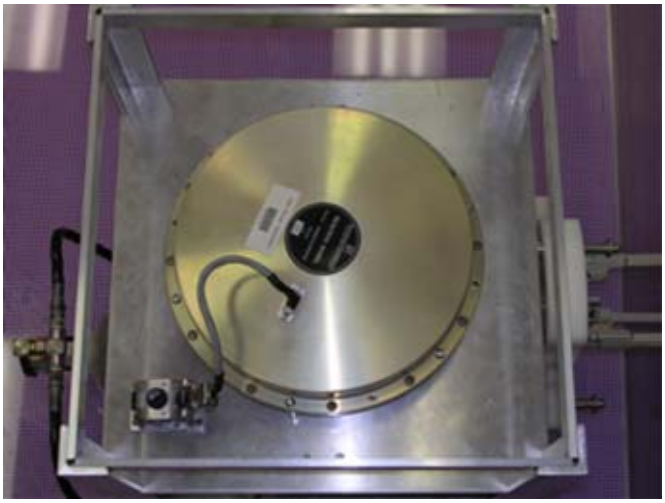

Figure 26. Reaction Wheel Deck

4. Electronics Deck

The electronics deck (see Figure 27 and Figure 28) is the most heavily loaded deck on the AUDASS II. This deck contains two lithium batteries, one DC/DC converter, two computers, one IMU, two digital and one analog I/O boards, and one indoor GPS receiver. One computer is used to determine attitude, relative position to the target vehicle, and laboratory position for the chaser vehicle. The other computer receives sensor inputs, executes a control algorithm, and transmits commands to the vehicle's actuators. The IMU senses both angular rates and accelerations, and provides a signal to the control computer. The two computers' and the IMU's integration into the vehicle will be described in greater detail later in this chapter.

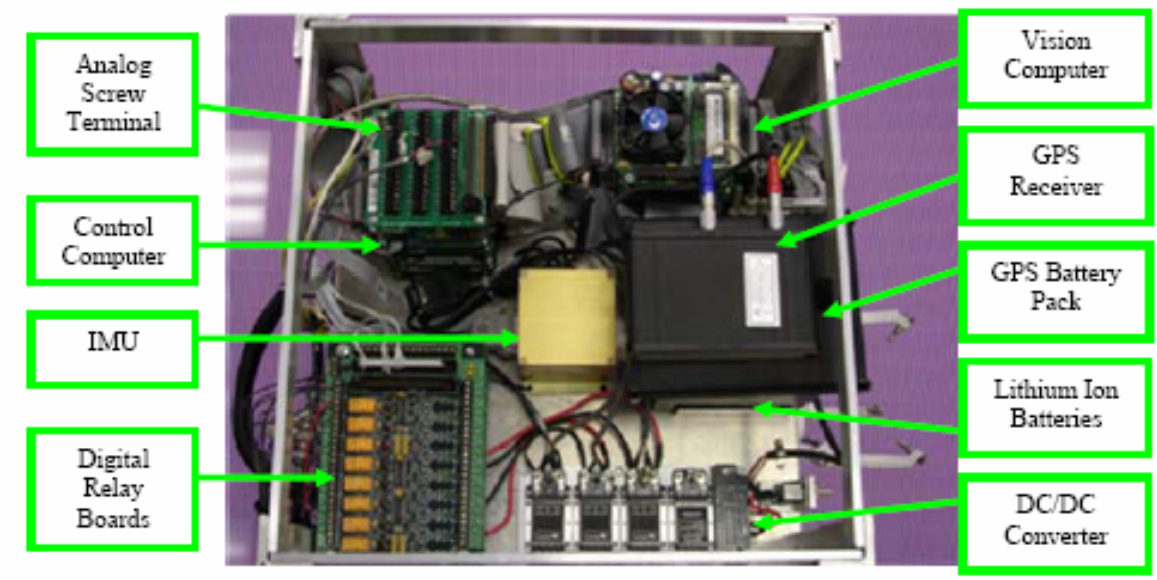

Figure 27. Electronics Deck

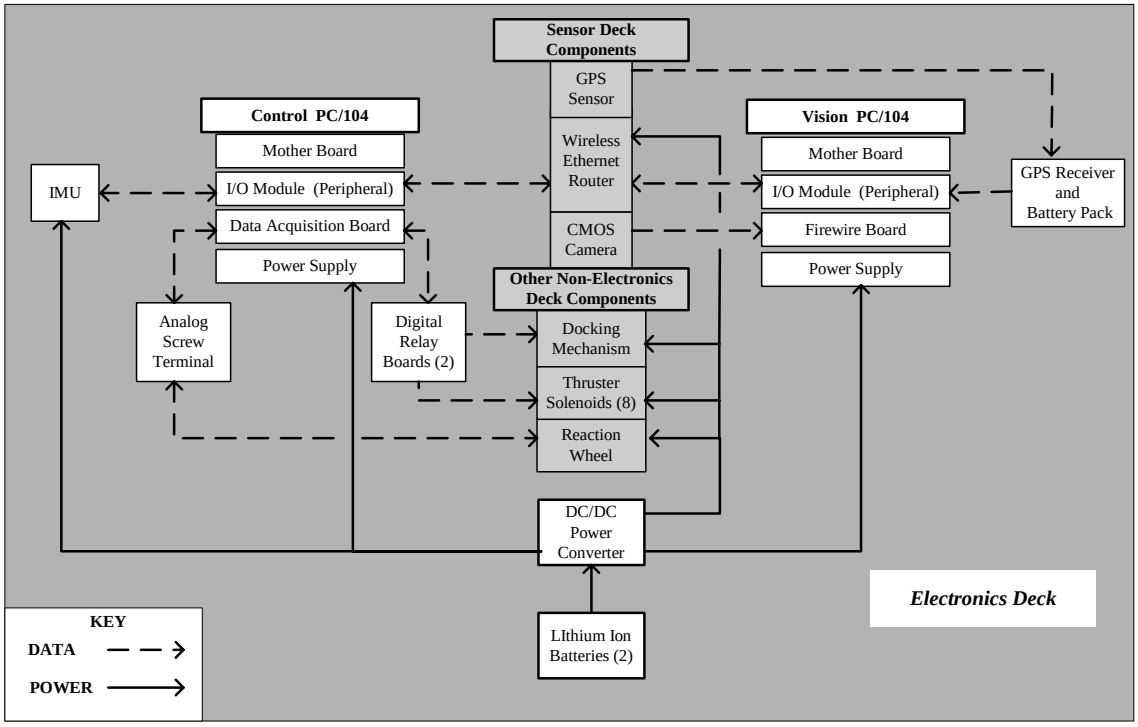

Figure 28. Data and Power Schematic of Electronics Deck Components

The batteries supply power to all the electronics on the AUDASS II with the exception of the GPS receiver. Two 28-volt lithium-ion batters carry six amp-hours of power. They are rechargeable and easy to remove from the vehicle.

The DC/DC converter conditions power from the two batteries and distributes the power load to the vehicle's components. The DC/DC converter has four outputs with which to distribute voltage. The nominal voltages from three of the outputs are 24 volts, while the fourth output is 5 volts, although all outputs have the ability to be trimmed. Trimming essentially modifies the voltages through the use of resistors to cater to individual component needs.

One analog screw terminal is the interface between the control computer's data acquisition board analog I/O to analog-based components on the vehicle. The analog interface contains 32 input ports, and is configurable between 32 single-ended inputs, 16 double-ended inputs, or a combination of 16 single-ended and 8 double-ended inputs. The interface also contains four 12-bit analog output channels. (Ref. [13]) The only component currently connected to the analog interface is the reaction wheel. The wheel's

Hall Effect sensors, which measure wheel speed, are connected as single ended inputs. The torque command is connected to the analog output. The torque feedback as connected as a double-ended input.

Two digital relay boards are the interface from the control computer's data acquisition board digital I/O and digital-based components on the vehicle. The digital I/O can communicate with up to 24 components (three ports with eight channels per port). (Ref. [13]) Since each digital relay board can take up to eight components, the vehicle could utilize three boards, although it currently only has two. (Ref. [14]) The first board sends signals directly to each of the eight thruster solenoids to allow for firing. The second board sends out three signals, while the other five channels are left idle for future component integration. Two of those signals go to the docking interface, one for the grappler's extension and the other for its retraction. The third signal is available to a future fluid storage system's solenoid valve and will allow for fluid transfer to take place.

The final component on the electronics deck is the indoor GPS receiver unit. The unit consists of the receiver, battery pack, and sensor converter. The battery pack contains a rechargeable 12 volt, nickel, metal-hydride battery and is positioned on the vehicle for easy removal. The receiver transmits data to the vision computer through a serial RS-232 connection.

5. Sensor Deck

The sensor deck (see Figure 29) contains all the components that interface with the external environment of the lab (with the obvious exception of the docking mechanism). This deck is unique in that will always rest on top of the vehicle.

The indoor GPS sensor provides the vehicle with laboratory position. Its mount is adjustable so that the sensor can always remain in line-of-sight of the two laser emitting transmitters, regardless of the simulation. Below the sensor is a matted surface used to absorb stray laser signals. The stray signals could reflect of the sensor deck's shiny aluminum surface and induce a multipath error on the GPS receiver.

The second sensor on the deck is the Pixelink CMOS camera. It serves as the primary sensor for the vehicle's computer vision navigation. The camera sits at the

center of the vehicle. It has a firewire interface and is connected to a specially designed board on the vision computer via a firewire cable.

A wireless Ethernet hub also sits on the sensor deck and serves two functions. It provides wireless connectivity from the off-line development computer to both control and vision computers. It also allows wired communication between the two online computers through Ethernet cable.

Figure 29. Sensor Deck

C. ONBOARD CONTROL COMPUTER

The control computer is charged with guiding the AUDASS II throughout the docking simulation experiments. It takes in sensor inputs, integrates them with control algorithms, and sends outputs to the reaction wheel and thruster actuators. To integrate a fully functioning computer on the vehicle, embedded computer technology was utilized.

To that end, PC/104 and PC/104 Plus modules were looked at to support the vehicle's computing needs. PC/104 embedded computers have several advantages over standard PCs including being lightweight, compact, and stackable. They have lower power requirements and have a higher resistance to shock and vibration than normal personal computers (PCs). They have standard interfaces between boards which allows for the addition of new boards and new functionality to the system. PC/104 Embedded Consortium is an organization dedicated to bringing the embedded computing community together to guarantee standards. For further capability, PC/104 Plus contains both an ISA and PCI bus which allows high speed processors, such as the Intel Pentium processor, to reach their full I/O bandwidth potential. (Ref. [15])

The VersaLogic EPM-CPU-10 was selected for the control computer (see Figure 30). The computer is compatible with PC/104 and PC/104 Plus interfaces. Mathworks' XPC Target Embedded Option is operated on it to execute Simulink based code built from the development computer. The computer also has a Diamond MM-33-AT data acquisition board for communication to external sensors. It uses a TRI-M HESC104 power supply to transfer power from the DC/DC converter to the computer. (Ref. [16])

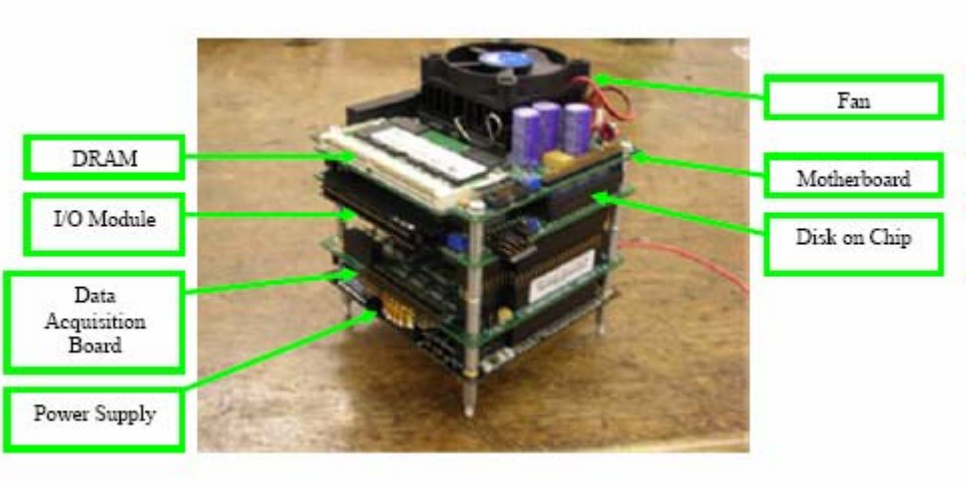

Figure 30. Control Computer

1. Computer Module and Peripheries

Starting from the top, the central processing unit (CPU) module controls the basic functions of the computer. It has an 850 megahertz Pentium III processor. It has a slot for dynamic random access memory (DRAM) chip. The chip holds 256 megabytes. The CPU also has a video interface, although this was only utilized during initial setup of the computer. To cool the module, a fan is directly attached to the CPU and operates whenever the computer is powered on. (Ref. [16])

Below the CPU and attached through a standard PC/104 plus 4x30 pin interface is the I/O module. The I/O module has three connections that are used routinely throughout its preparation and operation. The first is the Disk-on-Chip (DOC) interface. Rather than using a hard drive, which has more memory than the computer will ever use, the CPU relies on a DOC 2000 which can hold up to 256 megabytes. The DOC is much smaller

than a standard hard drive, measuring in at about 0.5 square inches of surface area. One caution that is emphasized in the computer manual bears repeating here. The DOC is the only device on the computer that could be installed backwards. If this is done, it will cause serious damage to the DOC making it inoperable. Pin 1 on the board must be aligned correctly with the printed dot on the DOC. The second connection on the board has connectors to a wide array of computer peripheries. These include one parallel port, two serial ports, keyboard and mouse ports, and one Ethernet port. The Ethernet port is connected to the wireless hub and is used during communication with the vision computer and the offline computer. The serial port is connected to the IMU. The keyboard interface was necessary during the computer's initial setup. The board's third interface goes to the 3.5 inch floppy drive. It is used to load the operating system and the XPC Target driver. (Ref. [16])

The next module down is the data acquisition board. It is attached to the above computer through a PC/104 connection. It has two connections to external digital and analog I/O boards. The configuration was discussed in the previous section.

The final module on the computer is the power supply. It is also connected to the above module through a PC/104 connection. One external connection allows for power to transfer from the DC/DC converter directly into the computer. (Ref. [17])

2. Details on BIOS Configuration and Software

As the PC/104 is a versatile machine, it comes completely unformatted with only the basic configurations preset in the Basic Input Output System (BIOS). Several steps were taken to configure the computer from a blank shell to the control computer required. The BIOS required configuration. DOS needed to be loaded on the computer to serve as a supporting operating system. Finally, XPC Target needed to be loaded as the main operation system.

a. Step 1

To configure the BIOS, the "delete" key was pressed at start up. Two items must be set before continuing. The DOC must be enabled as a drive. The Advanced Configuration screen was selected and the DOC setting was enabled to the

base address, D000:0h. The DOC becomes the C drive because of the absence of a hard drive. Under Basic CMOS Configuration Boot Order, the C Drive was added as the second boot device. (Ref. [16])

To load the operating system, a DOS boot disk, formatted as a boot floppy, was created and inserted into the floppy drive before turning on the computer. Once the computer was turned on and booted up, typing "format c: /s" at the A: prompt formatted the DOC, making it a bootable disk in the process by transferring the system files to the C drive. This allowed the computer to boot-up on its own without the use of a system disk. Note that in order to execute the format command, the format.exe file was included on the DOS boot disk.

b. Step 2

Once the computer was bootable, the XPC Target boot disk was created and installed. XPC Target is a real-time operating system from Mathworks that allows control algorithms to be developed using Simulink and sent as an executable to a target computer as the PC/104. XPC Target works with a host computer, in this case, the offline computer mentioned in a previous section. The host computer must have installed within Matlab the XPC Target and XPC Target Embedded Option toolboxes. Also, the XPC Target boot disk must be created from the host computer to work properly with the target PC, in this case the PC/104. (Ref. [18])

To create the boot disk, in the Matlab command prompt, type XPCEXPLR. Under the "Target PC" branch, the communications path sets up the TCP/IP protocol settings that the control PC will use. While the IP address can be set arbitrarily, the other settings must match that of the bus. (See Appendix C for a complete list of the settings.) The settings and appearance paths can be left at the defaults. Finally, under the configuration path, select the "Create Bootdisk" button to build the boot disk. Matlab will copy four files to the boot floppy: checksum.dat, xpctto16.rtb, xpcboot.com, and autoexec.bat. The autoexec.bat file contains the command xpctto16.rtb which will start XPC Target using a TCP/IP connection, with the target scope disabled, and accepting a maximum application build of 16 megabytes. Although not required, these

files were copied to a new directory with the C drive. To account for this, a second autoexec.bat file was created in the main directory directing the system to execute the autoexec.bat file in the new directory. The last step is to restart the computer and XPC Target will boot automatically. (Ref. [18])

If modifications or upgrades need to be made to XPC Target, the floppy drive will need to be reattached and the DOS boot disk will need to be placed in the floppy disk drive at start up. Then, the contents of the new XPC Target boot disk should be copied into the location of the old boot disk.

After completing these steps, the control computer was operational.

D. ONBOARD VISION COMPUTER

The purpose of the vision computer (see Figure 31) is to support applications required for the AUDASS II that cannot be supported by XPC Target on the control computer. Two applications required support from this second PC. To support the computer vision function, Matlab (and its Image Acquisition Toolbox) is needed to convert camera data into relative position data (chaser vehicle with respect to the target vehicle). To support the indoor GPS functionality, Workspace (Arc Second's software) is needed to calculate sensor inputs. (For Workspace to run properly, Microsoft.net framework must be installed first. Failure to do so will cause Workspace not to function.) Windows 2000 was chosen as an operating system in particular due to its program size. Again, the VersaLogic EPM-CPU-10 was selected to support the aforementioned programs. It is both PC/104 and PC/104 Plus compatible. (Ref. [19])

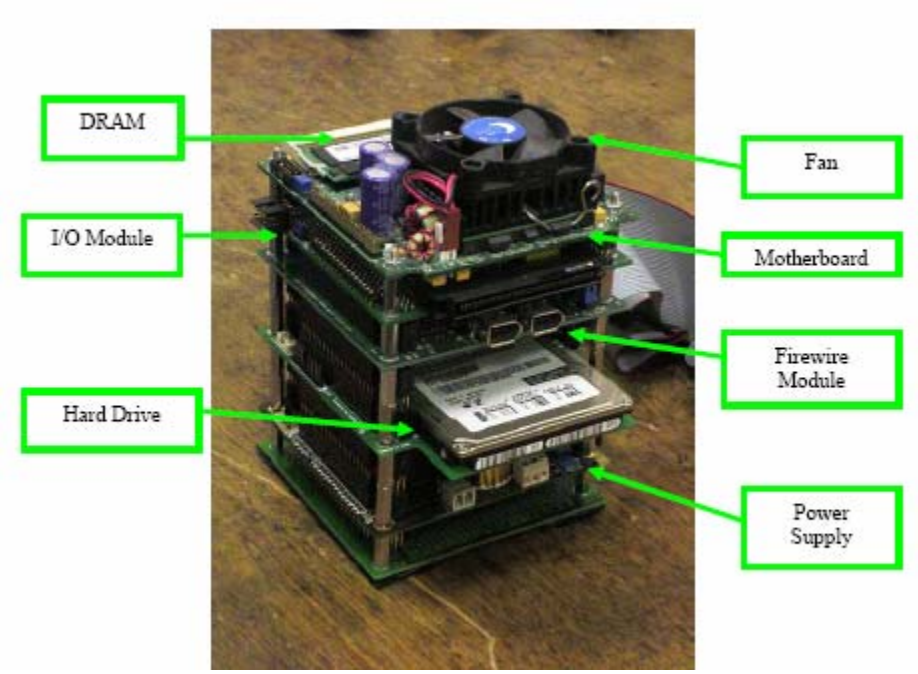

Figure 31. Vision Computer

1. Computer Module and Peripheries

The vision computer has a slightly different design from the control computer. The motherboard and I/O functionality are the same. It has connectors to support Ethernet, keyboard, mouse, two serial ports, a monitor, and an external power supply. One serial port supports the indoor GPS hardware. The Integrated Drive Electronics (IDE) interface supports two devices.

Below the I/O module is an Embedded Designs Plus firewire card. The firewire card is the interface between the CMOS camera and the computer. It is compatible with both PC/104 and PC/104 Plus modules and has two inputs for firewire compatible devices.

Unlike the control computer that used DOC flash memory, the vision computer, with its higher storage requirements uses a 40 gigabyte hard drive. Although it is connected through the IDE cable, the drive rests on its own module below the firewire card.

Connected below the hard drive is a Tri-M HESC104 power supply module. One external connection allows for power to transfer from the DC/DC converter directly into the computer.

E. IMU INTEGRATION

An IMU is used on the AUDASS II to collect angular rate and acceleration data of the vehicle and provide that data to the control computer for use in the Kalman Filter (see Chapter IV) and the vehicle's control algorithms (see Chapter V). The Crossbow's IMU400C model (see Figure 32) was selected for its compact, 3.0 x 3.75 x 3.72 inch frame, its low bias terms (+/- 1 degree/second for angular rate and +/- 12 milli-G's for acceleration), its ability to measure six degrees-of-freedom, and its serial format output. These factors allowed for the device to be integrated into the AUDASS II vehicle very easily.

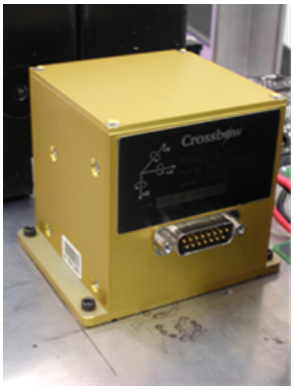

Figure 32. Crossbow IMU

The data provided from the unit comes in serial RS-232 format. Each packet of data has a standard format, seen below in Table 3. The header of each packet contains "255". The checksum value is the remainder when the same of the payload bytes, in decimal form, is divided by 256. Each element of data is made up of two, 8-bit unsigned integers to form one, 16-bit signed integer. The IMU has the capability to transmit data in a scaled sensor mode or a voltage mode. According to the Crossbow Manual, in the scaled sensor mode, "the analog sensors are sampled, converted to digital data, temperature compensated, and scaled to engineering units," using the full 16-bit range. In the voltage mode, "the analog sensors are sampled and converted to digital data with 1 [milivolt] resolution," but will only use 12-bits. To maximize the capability of the sensor, the scaled sensor mode was used and commanded on with an ASCII C command. (Ref. [20])

Byte	Scaled Sensor Mode	Voltage Mode
0	Header (255)	Header (255)
1	Roll Rate (MSB)	Gyro Voltage X (MSB)
2	Roll Rate (LSB)	Gyro Voltage X (LSB)
3	Pitch Rate (MSB)	Gyro Voltage Y (MSB)
4	Pitch Rate (LSB)	Gyro Voltage Y (LSB)
5	Yaw Rate (MSB)	Gyro Voltage Z (MSB)
6	Yaw Rate (LSB)	Gyro Voltage Z (LSB)
7	Acceleration X (MSB)	Accel Voltage X (MSB)
8	Acceleration X (LSB)	Accel Voltage X (LSB)
9	Acceleration Y (MSB)	Accel Voltage Y (MSB)
10	Acceleration Y (LSB)	Accel Voltage Y (LSB)
11	Acceleration Z (MSB)	Accel Voltage Z (MSB)
12	Acceleration Z (LSB)	Accel Voltage Z (LSB)
13	Temp Voltage (MSB)	Temp Voltage (MSB)
14	Temp Voltage (LSB)	Temp Voltage (LSB)
15	Time (MSB)	Time (MSB)
16	Time (LSB)	Time (LSB)
17	Checksum	Checksum

Table 3 IMU Data Packet Format (From Ref. [20])

While the IMU could be taken “out of the box” and plugged directly into the control computer’s serial port, its data would only be of use if it were properly parsed. Unfortunately, Simulink does not support any modules to interface with a serial device, check for a valid packet, and parse the signal. Simulink’s XPC Target and RT toolboxes only provide a portal to receive the data; parsing is left up to the user. (Ref. [18], [21]) Dobrokhodov, et al, provides a detailed “how-to” manual on parsing serial data by building a Level-2 S-Function in Simulink. (Ref. [22])

Although the code described how to parse a Microstrain 3DM IMU, only a few modifications to the source code were necessary to correctly parse the Crossbow IMU. The source code can be seen in Appendix D. The messages length is “18” (bytes) as can be inferred from Table 3. The header remained at “255” (FF in hex). The input width is

set to “1” signal. The output width is set to “16.” Once modified, the code was converted to a dynamic link library (executable file that Simulink references in an S-Function block) using the “mex” command in Matlab.

To verify the parsing code was functioning correctly, three tests were conducted in Simulink.

1. Check for proper output of a known data sample
2. Verify near real-time functionality of the code
3. Verify the code worked correctly with XPC target.

Mockups of the three Simulink tests can be seen in Figure 33.

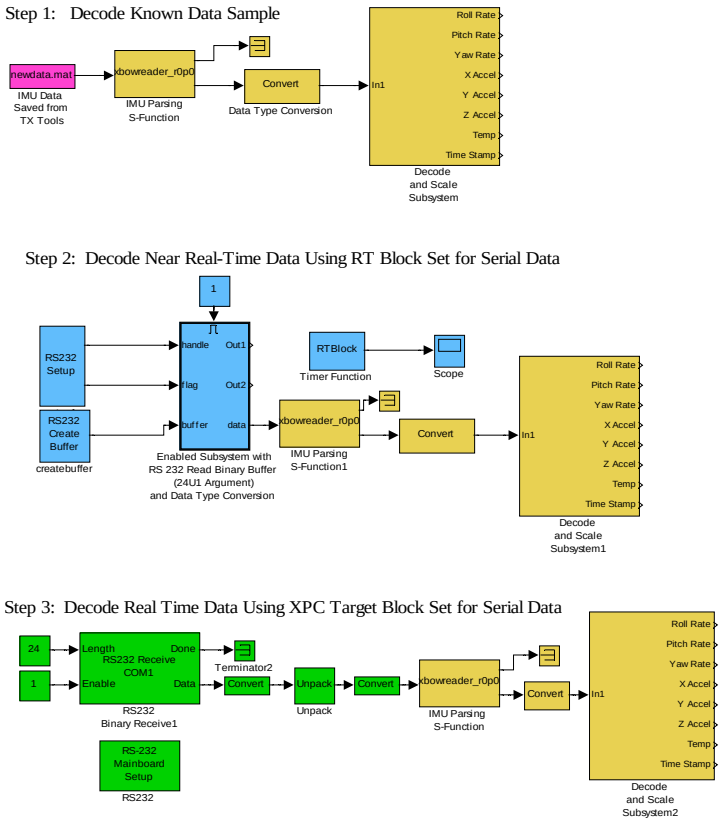

Figure 33. Simulink Block Diagrams of the Three Stages to IMU Integration

To execute the first test, a sample of serial data was taken from the IMU while performing three known eigenaxis (roll, pitch, and yaw) maneuvers using the freeware program, TXTools. The data sample was converted from ASCII data (TXTools output)

to eight-bit, unsigned integers and placed in a matrix to be read by the S-Function. Once the test completed, it was easy to verify the header and checksum data were removed.

Before executing the second test, the outputted data bytes were combined and scaled to verify the output was in fact recognizable angular rate and acceleration data. To combine two, 8-bit unsigned integers into one, 16-bit signed integer, the following formulas were used, as seen in Equations 1 and 2 (converted into Simulink block language).

Equation 1, 2

If MSB < 128, then Answer = MSB*256 + LSB
If MSB > 128, then Answer = (MSB-256)*256 + LSB

To properly scale each of the eight data points the following conversions were applied as seen in Equation 3, 4, and 5.

Equation 3, 4, 5

$$\text{Angular Rates } (^\circ / s) = \text{data} * AR * 1.5/2^{15}$$
$$\text{Acceleration (G)} = \text{data} * GR * 1.5/2^{15}$$
$$\text{Temperature } (^\circ C) = [\text{data} * 5/4096 - 1.375] * 44.44$$

Here, AR stands for Angular rate Range and GR stands for G Range. Both ranges are provided by the manufacturer and are 100 degrees/second and 4 Gs respectively. Temperature refers to that of the IMU. The time number has no conversion. It is a time tag internal to the IMU and counts down from 65,535 to in 0.79 microsecond increments.

Once the data was properly read and scaled, the second test for reading near real-time data was executed. To set up the test, the RT RS-232 block sets were utilized for serial interface functionality. (Both block sets are freeware downloads from the Mathworks user-community website. Note that this RS-232 block set is not the same block set used by XPC Target.) Windows typically has a problem with real-time applications. A user can increase his priority in Windows through the RT block set gaining near real-time results. One downside to this block is it can potentially crash a computer if running a complex enough program. Luckily, it did not occur when this program was run.

Once the second test proved successful, the code was implemented using XPC Target. While XPC Target reads the data in real-time, it does not output this data to the

user until after a simulation is run. To verify proper functionality, three eigenaxis maneuvers were performed. Note that when using the XPC Target RS-232 functionality, whether sending or receiving data, an RS-232 Mainboard Setup block must also be included in the simulation. Additionally, the Pack and Unpack blocks must be used for proper data communication. They come before an RS-232 Send block and after an RS-232 Receive block.

While receiving data was a major integration challenge, initializing the IMU into "continuous" mode, was also essential for proper testing. The ASCII "C" command (an 8-bit, unsigned integer equivalent of "67") is sent once after the component is powered on.

A final point to discuss about IMU integration is the performance of the parsing function. Performance has to do with output rate in which a valid data packs are identified. Reference [22] discusses this in some detail. It suggests that over-sampling the incoming data stream by 10-15 % of the packet length will lead to optimal performance. With that in mind, the incoming width of the parsing function (set in the Simulink S-Function block) was set to 20 bytes.

F. REACTION WHEEL INTEGRATION

The Ball Aerospace reaction wheel is integrated into the chaser vehicle to perform attitude control. Analog voltage input commands are sent to the reaction wheel. Those commands will cause the wheel to either slow down or speed up, thus imparting a torque on the vehicle. This torque will then adjust the orientation of the vehicle.

A software-based magnitude-only tachometer (Appendix E) was developed in the control software to measure the reaction wheel's speed and protect it from user-induced damage. (The reaction wheel can be operated up to 3500 revolutions per minute (RPM), although the manufacturer does not recommend the operation beyond 2500 RPM.) The tachometer uses the square-wave signal from one of the three Hall Effect sensors embedded within the reaction wheel structure. Since every revolution produces four rising triggers, determining the wheel's speed becomes a simple exercise. (Ref. [12])

IV. STATE ESTIMATION THROUGH KALMAN FILTER PROCESSING

A. INTRODUCTION

The primary function of the AUDASS II is to perform an autonomous docking maneuver. In support of this, control algorithms must be developed to articulate how the thrusters and reaction wheel should be used to adjust the vehicle's position and attitude orientation. However, this control system will be ineffective unless it has an input reference signal which will tell it how far off it deviates from its final objective in terms of position and attitude.

The CMOS PixeLINK camera was installed on the AUDASS II to provide that reference, by taking pictures of three LEDs mounted on the target vehicle. Those pictures, once interpreted by the vision computer's Matlab developed software, calculate an instantaneous reference for the control. A reference vector can only be provided five times every second (5 Hz). The control functions at a much higher frequency of 100 Hz.

A Crossbow IMU was placed on board the AUDASS II to provide angular rate and accelerometer information. This device can operate up to 133 HZ. The downside of such a device is that it is inherently noisy and has a drift rate term, which if left uncorrected, would hinder the control.

A Kalman filter was developed to take in the CMOS camera data and IMU inputs and produce an accurate estimation of the current attitude and position, and the bias signals of the gyro and accelerometers. This chapter will describe the mathematical foundation, implementation, and performance of the Discrete Kalman Filter (DKF) used for the AUDASS II.

B. THEORY

A Discrete-Time Linear Kalman filter was implemented (Ref. [23]) considering the following discrete dynamics model, based on the kinematics of the rotational and linear motion:

Equation 7 $$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \Gamma_k \mathbf{u}_k + \Upsilon_k \mathbf{w}_k .$$

The state vector at sample time, k, is given by

Equation 8 $$\mathbf{x}_k = \begin{bmatrix} x & \dot{x} & \alpha & z & \dot{z} & \beta & \theta & \gamma \end{bmatrix}^T,$$

where x and z are components of the position vector of the chaser spacecraft with respect to the target spacecraft along the target frame, θ is the relative attitude-angle of the chaser vehicle with respect to the target, and α, β, γ are the biases of the two acceleration measurements along x and z, and of the rate gyroscope measurement along y, respectively.

The state transition matrix of Equation 7 is given by

Equation 9 $$\Phi_k = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -\Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -\Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

where Δt is the sample time.

The input vector, at sample time k, in Equation 7 is given by

Equation 10 $$\mathbf{u}_k = \begin{bmatrix} \tilde{\ddot{x}} & \tilde{\ddot{z}} & \tilde{\dot{\theta}} \end{bmatrix}^T,$$

where $\tilde{\ddot{x}}$, and $\tilde{\ddot{z}}$, are the measurement signals of the accelerometers along x and z, respectively, projected along the target reference frame. And $\tilde{\dot{\theta}}$ is the measurement signal of the rate gyroscope.

The remaining terms in Equation 7 are defined as follows:

Equation 11, 12

$$\Gamma_k = \begin{bmatrix} 0 & 0 & 0 \\ \Delta t & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta t \\ 0 & 0 & 0 \end{bmatrix}, \Upsilon_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

Equation 13

$$\mathbf{w}_k = \begin{bmatrix} w_{acc\ x} \\ w_{\alpha} \\ w_{acc\ z} \\ w_{\beta} \\ w_{gyr} \\ w_{\gamma} \end{bmatrix},$$

where the elements of the $\mathbf{w}_k$ are the assumed zero-mean Gaussian white-noise processes related respectively to the output of the accelerometer along z, the bias of the accelerometer along x, the output of the accelerometer along z, the bias of the accelerometer along z , and the output of the gyroscope and the bias of the gyroscope.

The measurement equation used for the Discrete-Time Linear Kalman filter implementation is

Equations 14, 15, 16

$$\begin{aligned} \tilde{y}_{k-delay(k)} &= H_{k-delay(k)}\mathbf{x}_{k-delay(k)} + \mathbf{v}_{k-delay(k)} \\ H_{k-delay(k)} &= [1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0], \\ \mathbf{v}_{k-delay(k)} &= \begin{bmatrix} v_{cam\ x} & v_{cam\ z} & v_{cam\ \theta} \end{bmatrix} \end{aligned}$$

where the subscript $(k - delay(k))$ indicates that the camera outputs the angle measurement with a delay Δ, which is in general different for different sample times. Moreover, $v_{cam\ x}$, $v_{cam\ z}$, $v_{cam\ \theta}$ are the assumed zero-mean Gaussian white-noise processes related to the vision sensor measurements.

In order to compensate for the measurement delay, a propagation of the state from the $(k - delay(k))^{th}$ sample-time to the k^{th} sample-time is conducted through the Kalman

Filter propagation equations, whenever a new camera measurement is obtained. This propagation is performed within one sample time (between $(k-1)$ and k).

The process noise covariance and measurement noise covariance, used in the Kalman filter, follow:

Equation 17

$$Q = \Delta t \begin{bmatrix} \sigma_{accx}^2 + \frac{1}{3}\sigma_\alpha^2 \Delta t^2 & -\frac{1}{2}\sigma_\alpha^2 \Delta t & 0 & 0 & 0 & 0 \\ -\frac{1}{2}\sigma_\alpha^2 \Delta t & \sigma_\alpha^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{accz}^2 + \frac{1}{3}\sigma_\beta^2 \Delta t^2 & -\frac{1}{2}\sigma_\beta^2 \Delta t & 0 & 0 \\ 0 & 0 & -\frac{1}{2}\sigma_\beta^2 \Delta t & \sigma_\beta^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{gyr}^2 + \frac{1}{3}\sigma_\gamma^2 \Delta t^2 & -\frac{1}{2}\sigma_\gamma^2 \Delta t \\ 0 & 0 & 0 & 0 & -\frac{1}{2}\sigma_\gamma^2 \Delta t & \sigma_\gamma^2 \end{bmatrix}$$

Equation 18

$$R = \begin{bmatrix} \sigma_{camx}^2 & 0 & 0 \\ 0 & \sigma_{camz}^2 & 0 \\ 0 & 0 & \sigma_{cam\theta}^2 \end{bmatrix}.$$

The values of the noise parameters used for the experimental tests reported in this thesis, are given in Table 4.

Δt	0.01 s
σ_{accx} , σ_{accz}	5.6 x 10^{-3}
σ_α , σ_β	10^{-4}
σ_{gyr}	4.9 x 10^{-3}
σ_γ	10^{-3}
σ_{camx}, σ_{camz}, $\sigma_{cam\theta}$	3 x 10^{-4}
P_0	diag[10^{-4} 10^{-8} 10^{-3} 10^{-4} 10^{-8} 10^{-3} 10^{-3} 10^{-2}]
Kalman Filter Sample Time	100 Hz
Vision Sensor Nominal Update Frequency	5 Hz
IMU Bandwidth	133 Hz

Table 4 Values of Kalman Filter Parameters Used During Experimentation

C. IMPLEMENTATION

The Kalman filter algorithm described in the previous section has been implemented in Simulink and run in real time during experimental tests. The external

view of the orientation DKF can be seen below in Figure 34. The filter block is an enabled subsystem that operates on a 100 Hz clock. The inputs to the system are the rate gyro measurement, the orientation measurement for the camera, and a flag to signal if a new camera measurement is available. The principle outputs of the filter are the state variable, the estimated orientation, and the estimated gyro drift rate. The filter also internally loops the covariance matrix to transfer it from posteriori to a priori status.

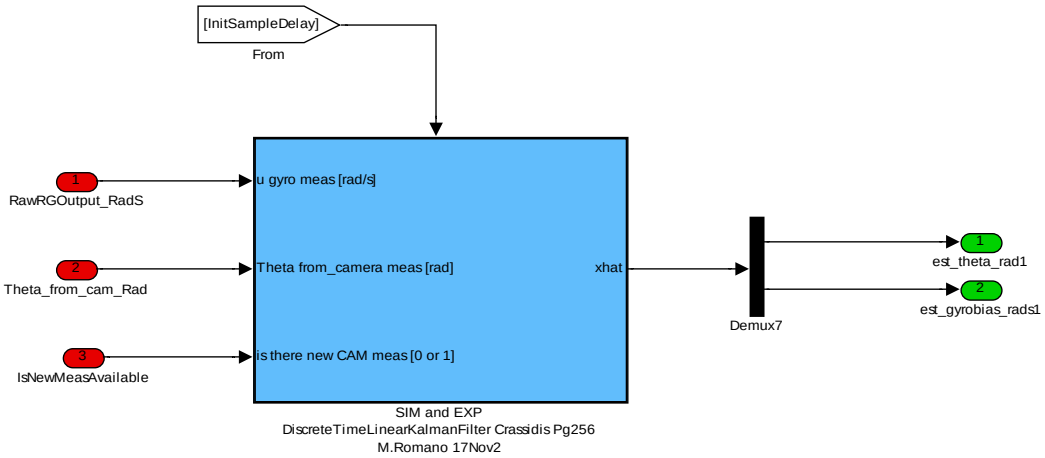

Figure 34. External Simulink View of the Orientation Kalman Filter

The internal view of the filter is a little more convoluted, as seen in Figure 35. However, it does directly reference the equations in the previous section. The major block on the left contains the propagation equations. Once propagated, the results are numerically integrated and fed into the major block on the right. Within that block are the Kalman gain and the state vector update equations.

The internal structure for the X and Z position DKFs are identical to the orientation DKF. The only differences are in its specific inputs which are described in the previous section.

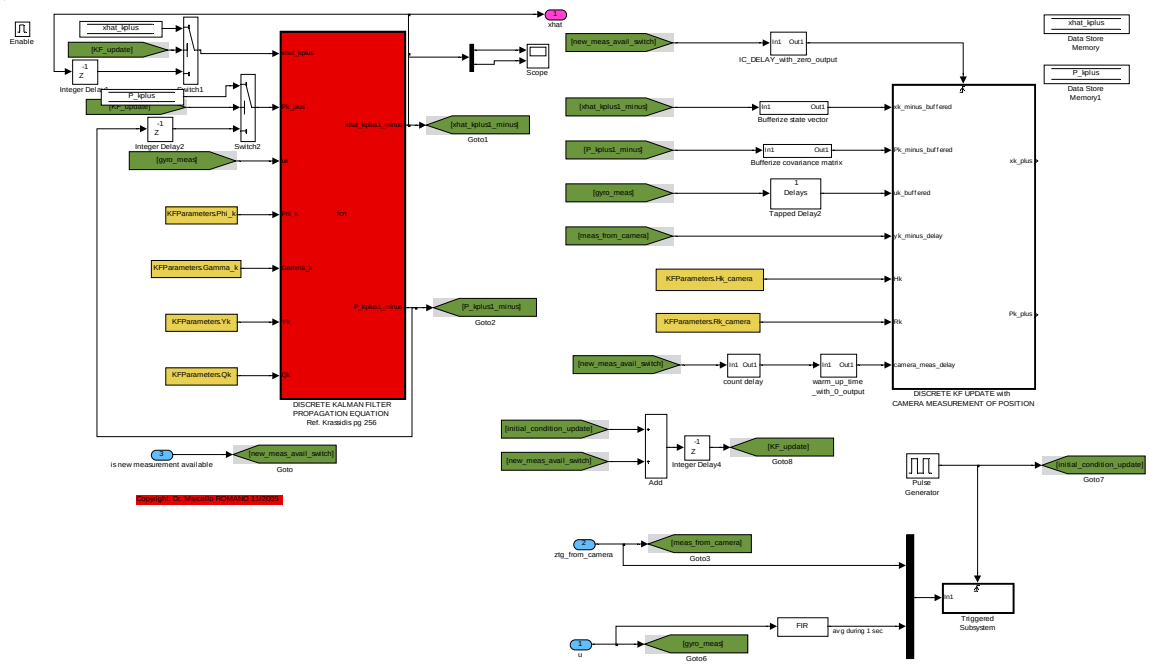

Figure 35. Internal Simulink View of the Orientation Kalman Filter

D. PERFORMANCE

After designing and implementing the Kalman filter on the AUDASS II, it was tested to verify that the relative orientation, position, and sensor biases were being estimated correctly. The data used in those test were taken from the two experiments described and plotted in Chapter 5. This Kalman filter, to function properly, must be given enough time to converge on a solution without any disturbances. Ten seconds was found to be an acceptable duration.

After each simulation was run, a good first indication that the filter was operating correctly was that the estimated biases from the rate gyro and accelerometers displayed linear characteristics. Figures 47, 48, 49, 57, 58, and 59 report examples of how the Kalman filter correctly estimated that the biases changed at a very slow rate.

With the bias assumed correct, estimated navigation states can safely be compared to the position and orientation measurements produced by the camera. Figures 42, 43, 44, 52, 53, and 54 show that the Kalman filter position estimates with the camera measurements to within 1 centimeter. The 10 second discrepancy at the beginning of each experiment is a preprogrammed hold, to allow the filter to converge on a solution.

V. EXPERIMENTAL RESULTS OF AUTONOMOUS CONTROL

A. DESIGN

The final goal of this thesis was to perform an autonomous docking maneuver using the AUDASS II as the chaser vehicle and the original AUDASS, mounted with the docking receiver, as the target. To that end, the controls and supporting code were developed to bridge the gap between the Kalman Filter output (and derived error signal) and the actuators performing the docking maneuver. The reaction wheel was controlled through a Proportional-Derivative (PD) control. The thrusters were controlled by way of a Schmitt Trigger using Pulse-Width Modulation (PWM).

1. Reaction Wheel Control

The reaction wheel can be modulated through an analog, +/- 2 Volt signal sent from the control computer. Because it behaves as a simple, continuous system, a PD controller was implemented (see Figure 36). The controller used the orientation estimated through the Kalman Filter as the proportional signal. The derivative signal was taken from the IMU's rate gyro (after the estimated bias was removed).

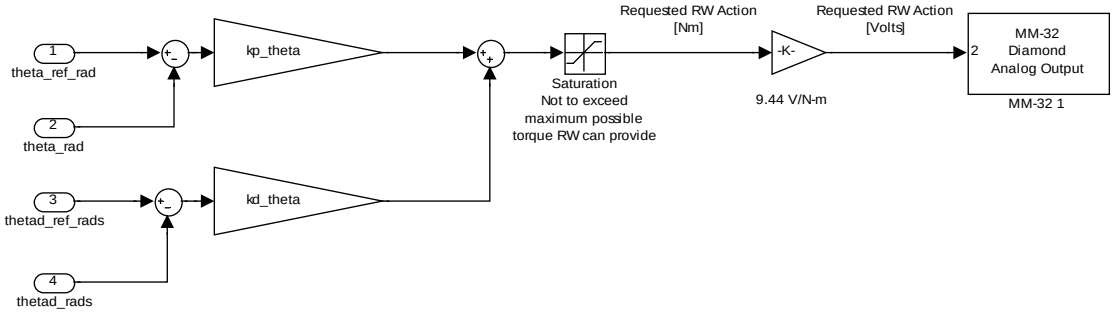

Figure 36. Simulink Model of the Reaction Wheel's PD Control

2. Thruster Control

The thruster control was implemented in two phases. The first phase uses a Schmitt Trigger algorithm. A Schmitt Trigger is essentially a "relay with a deadband/ hysteresis." A graphical representation is seen in Figure 37. The net thruster output will either be 0, 1, or -1, with each individual thruster's output being completely open or

closed. The benefit of using this method of control, as opposed to the PD control used for the reaction wheel, is that it can treat the thrusters as a discrete system, thus conserving propellant throughout the simulation. The tradeoff in using such a system is that target attitude error and attitude rate error will never reach absolute zero; rather a low-frequency limit cycle will be the result. Having this deadband is an acceptable tradeoff to the docking mechanism/receiver since the two pieces can mate even with a horizontal offset of approximately 5 centimeters. (Ref. [24])

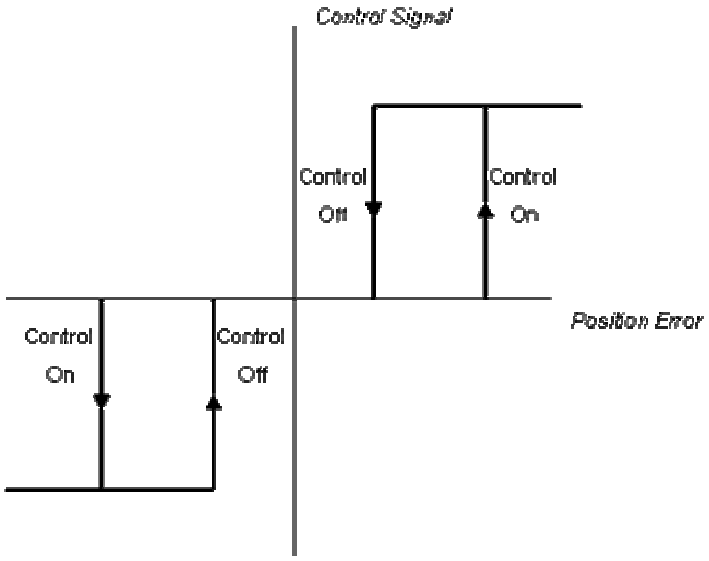

Figure 37. Schmitt Trigger (After Ref. [24])

The second phase of the thruster control comes through the use of PWM. PWM controls the pulse width of the thruster, as the name would imply, to meet the thrust demands generated by the Schmitt Trigger. The width can range from a minimum physical pulse width equal to the thrusters' solenoid minimum opening time (represented by T_1 in Figure 38) to a maximum pulse width equal to the PWM logic's sampling period (represented by T_2 in Figure 38). The PWM logic can be seen in Figure 38. Its implementation into the AUDASS II can be seen through the Simulink model in Figure 39. (Ref. [24])

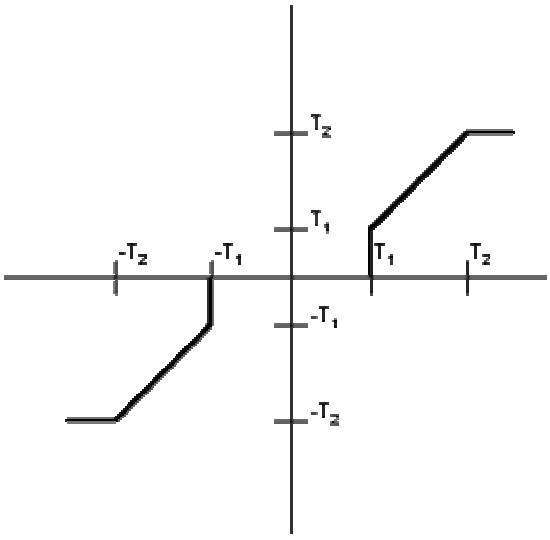

Figure 38. PWM Logic (After Ref. [24])

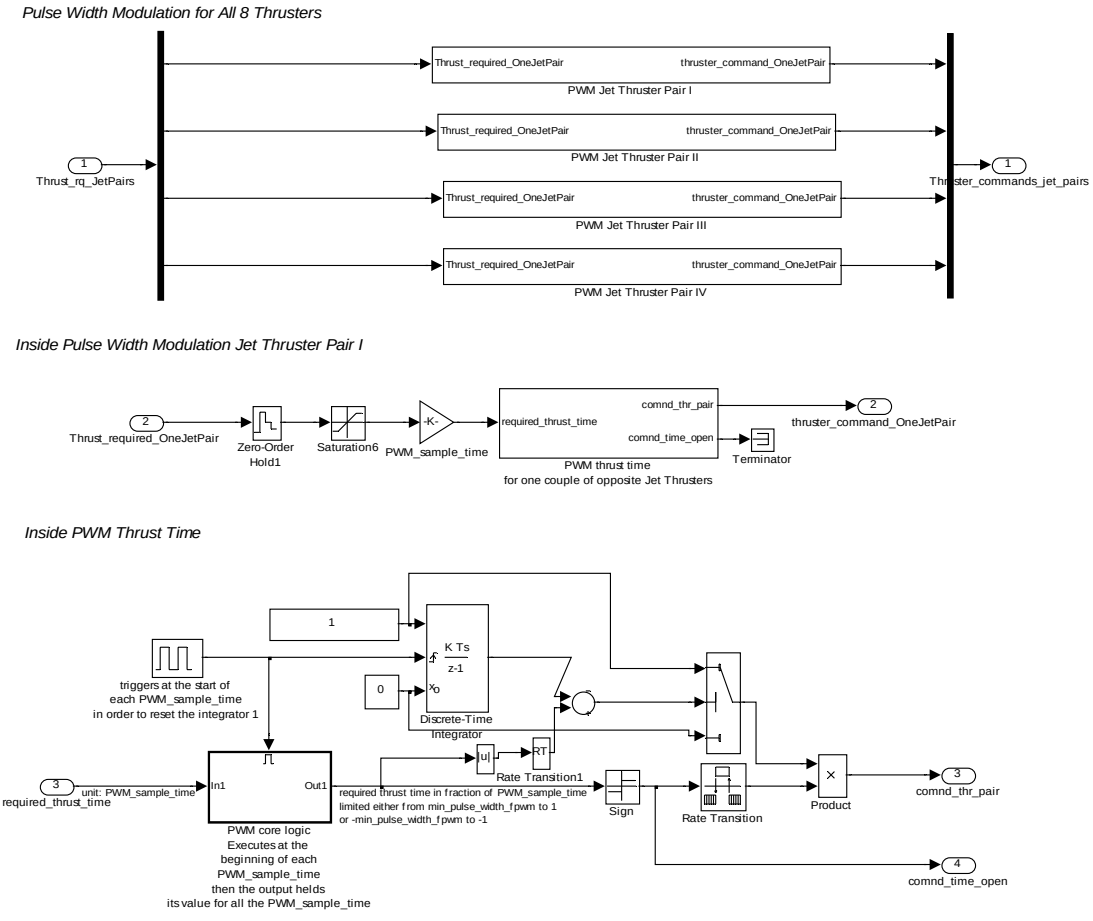

Figure 39. Simulink Model of Pulse Width Modulation as Used by AUDASS II

B. EXPERIMENTS

In order to gauge how well the control actually performed on the AUDASS II, two experiments were conducted. Each experiment comprised a series of maneuvers. The first test had the chaser vehicle travel in a closed loop path in front of the target vehicle. The second test had the vehicle perform an L-shaped maneuver where the chaser vehicle would approach slowly and finally dock with the target vehicle.

1. Dodecagon Tracking Maneuver

The importance of completing a dodecagon maneuver is that it can show how agile the vehicle is at it moves in the plane of the flat floor. The reference maneuver is diagramed in Figure 40. In the experimental run, the vehicle was commanded to move to each of the 12 spots in the polygon pattern. The vehicle was given ten seconds to reach each spot and hold its position before being commanded to the next spot. At the same time, the vehicle was required to hold its orientation by looking ahead in the direction of the target vehicle. The experiment lasted 180 seconds.

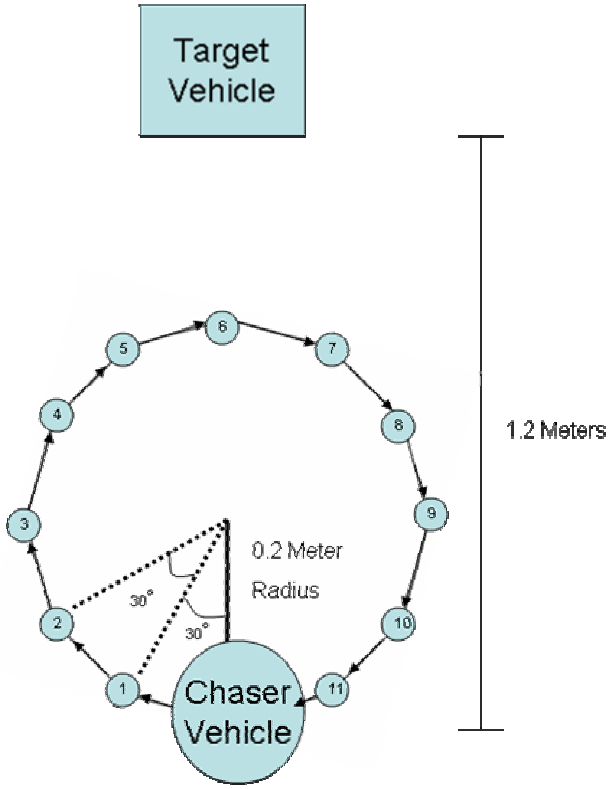

Figure 40. Dodecagon Tracking Maneuver Diagram

The results of the dodecagon test were judged to be rather successful. Data plots can be seen from Figure 41 to Figure 49. In Figure 41, the spatial position from the simulation is plotted. The filtered data for each individual axis is seen in Figure 42 and Figure 43. Of note, the vehicle spends the first 15 seconds of the simulation (post Kalman filter convergence) acquiring the starting point. This is seen at the bottom-center of the plot. Following initial acquisition, it appears that there was some transient motion in the system as it attempts to move to the next several set-points, which is shown in the loops that were mapped out. At any given point, the overshoot does not exceed 5 centimeters.

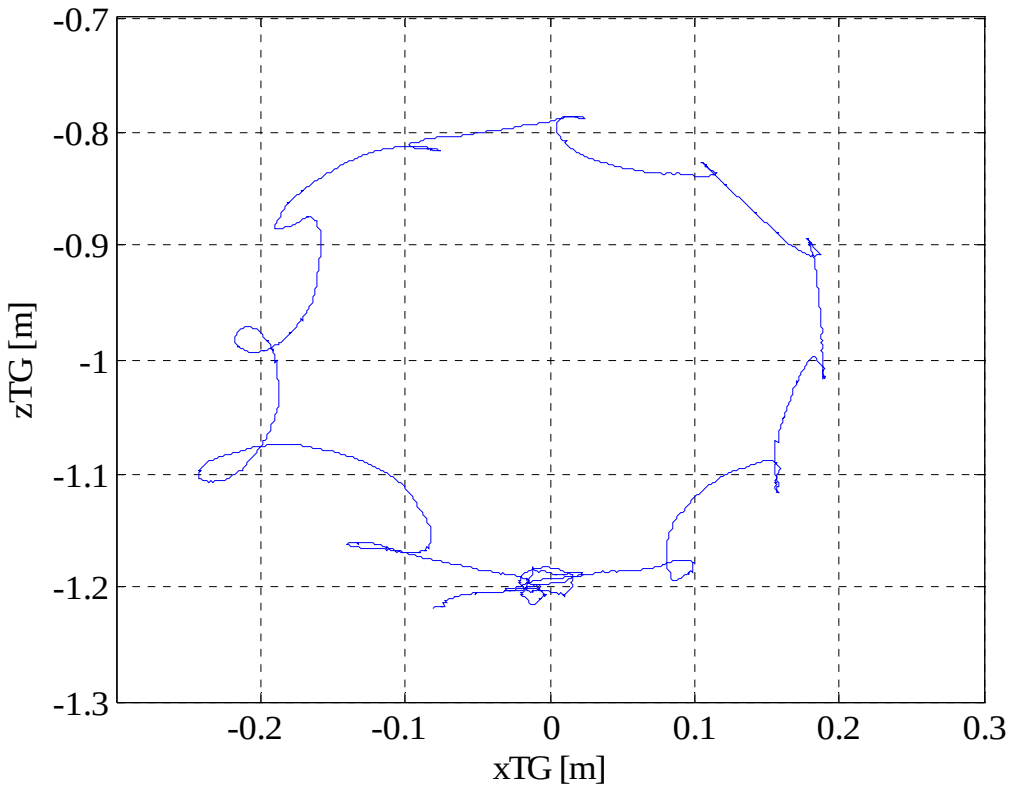

Figure 41. Experimental Results: Position of the Chaser Vehicle During the Dodecagon Tracking Maneuver

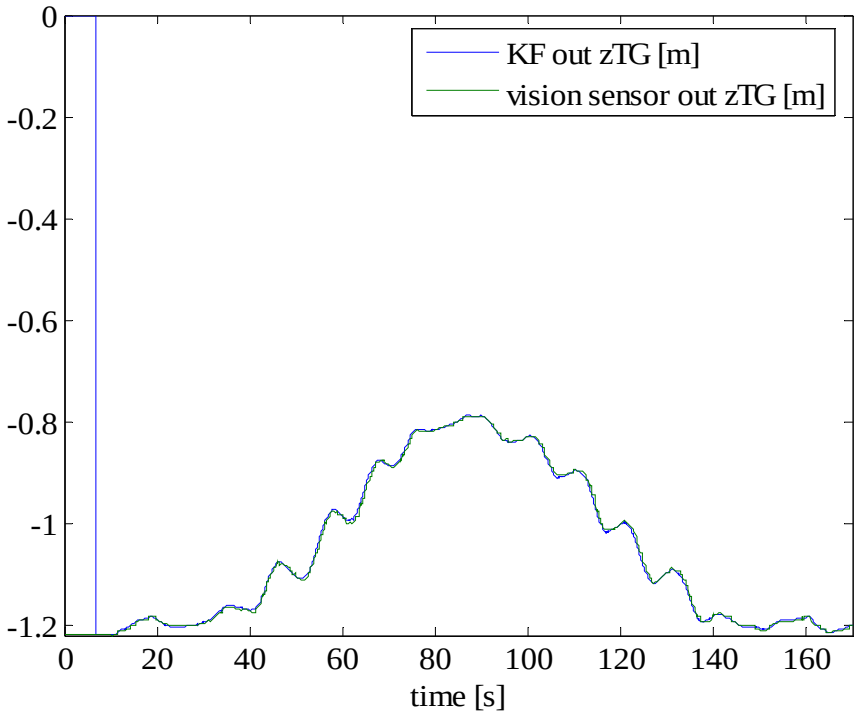

Figure 42. Experimental Result: Ranges from Chaser to Target Using Vision Sensor and Estimation

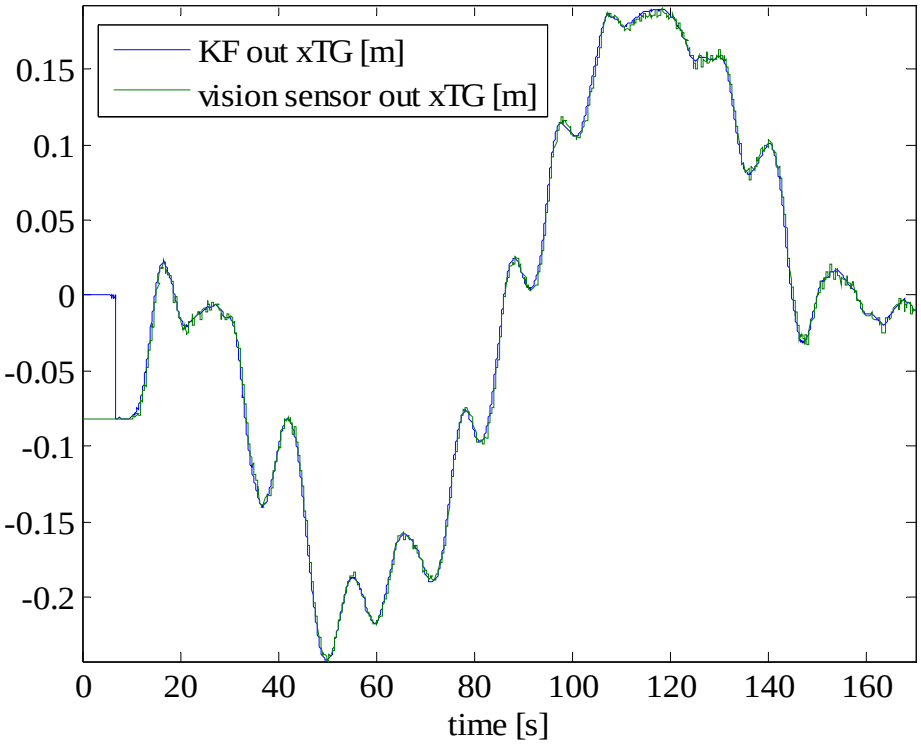

Figure 43. Experimental Result: Alignment Offset of Chaser to Target Using Vision Sensor and Estimation

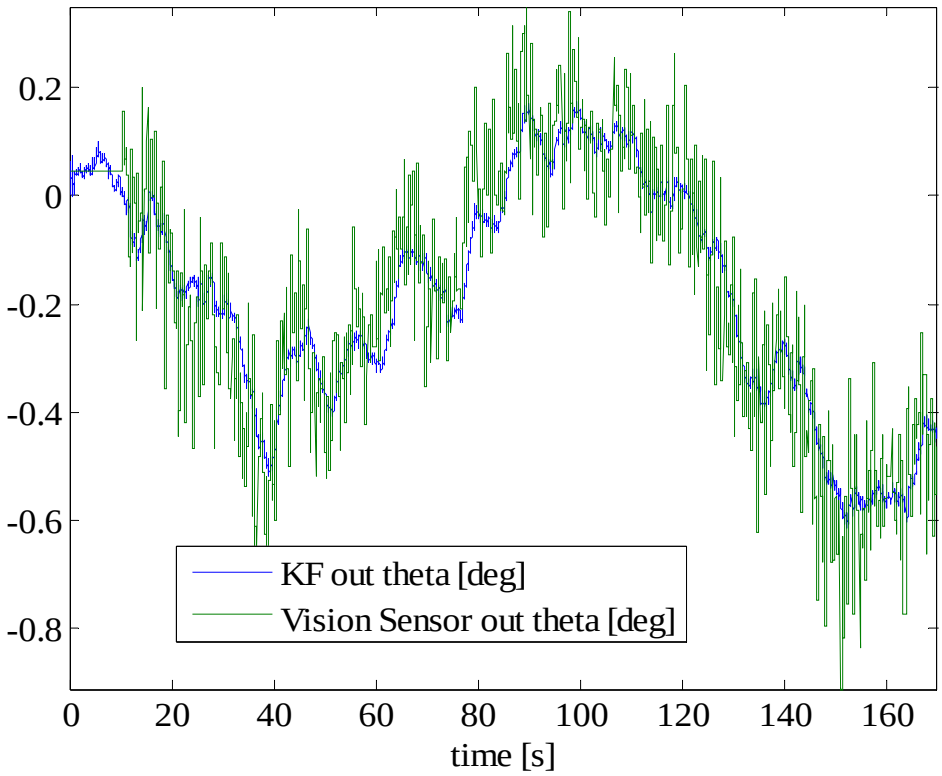

Figure 44. Experimental Results: Relative Orientation of Chaser with Respect to Target Using Vision Sensor and Estimation

The actuator commanding is shown in Figure 45 for the thruster pairs and in Figure 46 for the reaction wheel. Thruster Pairs I and II refer to the forward and aft thrusters while Pairs III and IV refer to the port and starboard thrusters. For each set-point, one grouping brings the vehicle to the point and one grouping slows it down. For the reaction wheel, since its control is at a much higher bandwidth, the resultant high-frequency plot comes as expected.

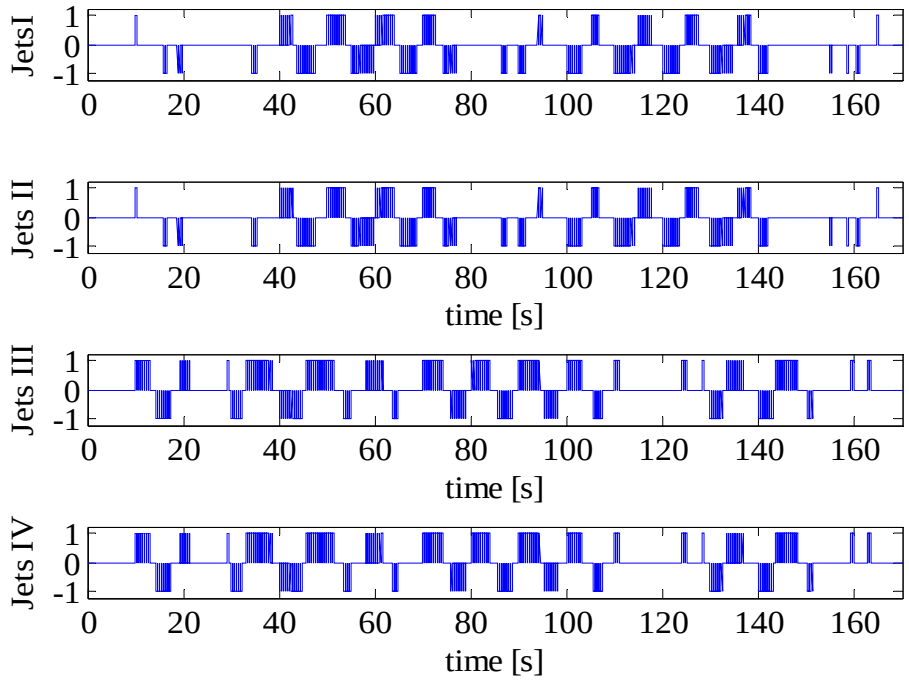

Figure 45. Experimental Results: Commands to Each Thruster Pair

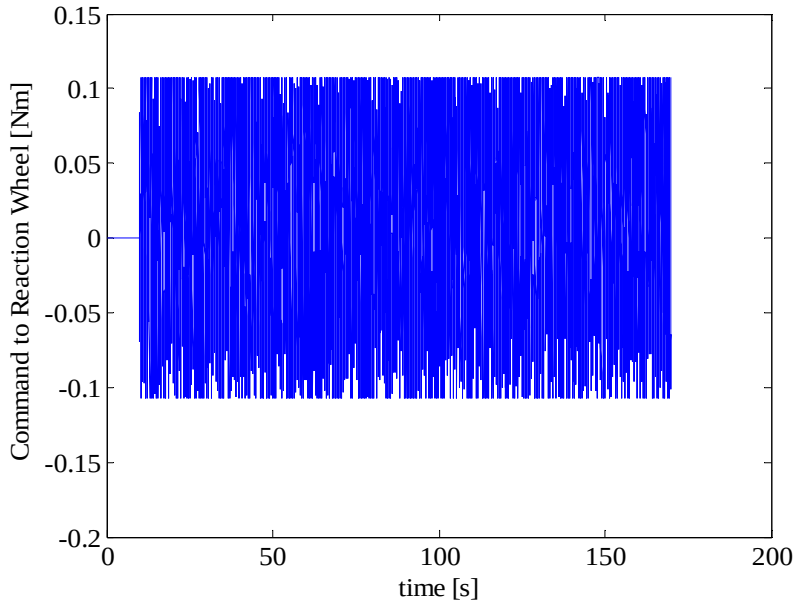

Figure 46. Experimental Results: Commands to the Reaction Wheel

The rate gyro and accelerometer data is shown before and after the bias is filtered out, below in Figures 47, 48, and 49. As was discussed in the previous chapter, the Kalman solution looks to be a major contributing factor to the success of this simulation.

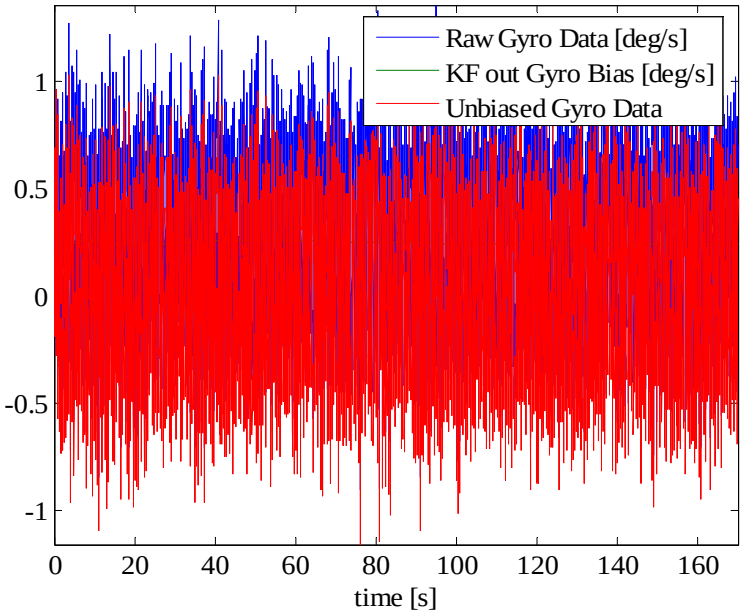

Figure 47. Experimental Results: Filtered and Unfiltered Rate Gyro Data

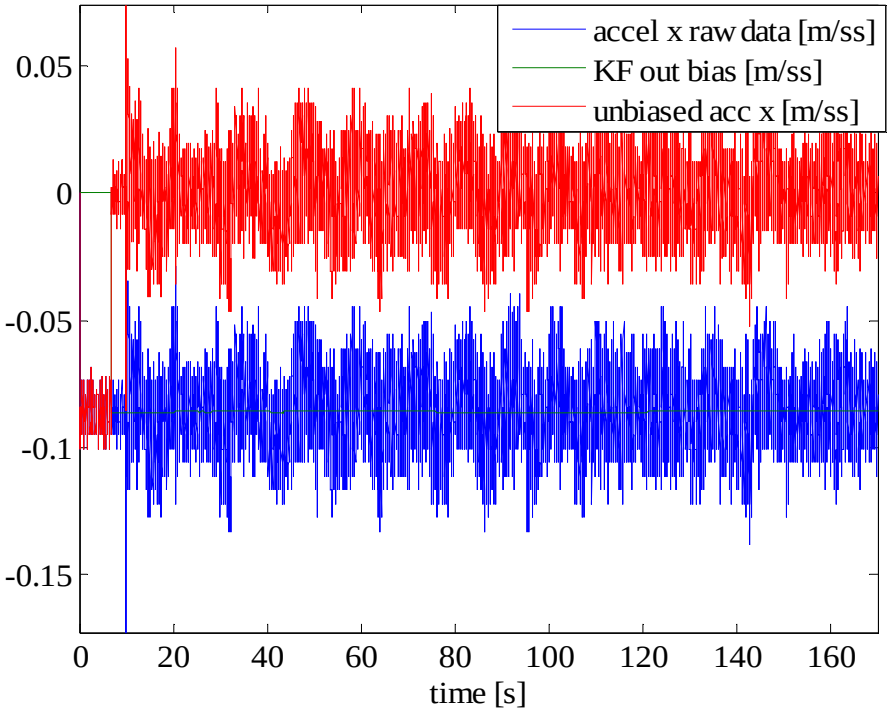

Figure 48. Experimental Results: Filtered and Unfiltered Accelerometer Data (Along X Axis of Target Frame)

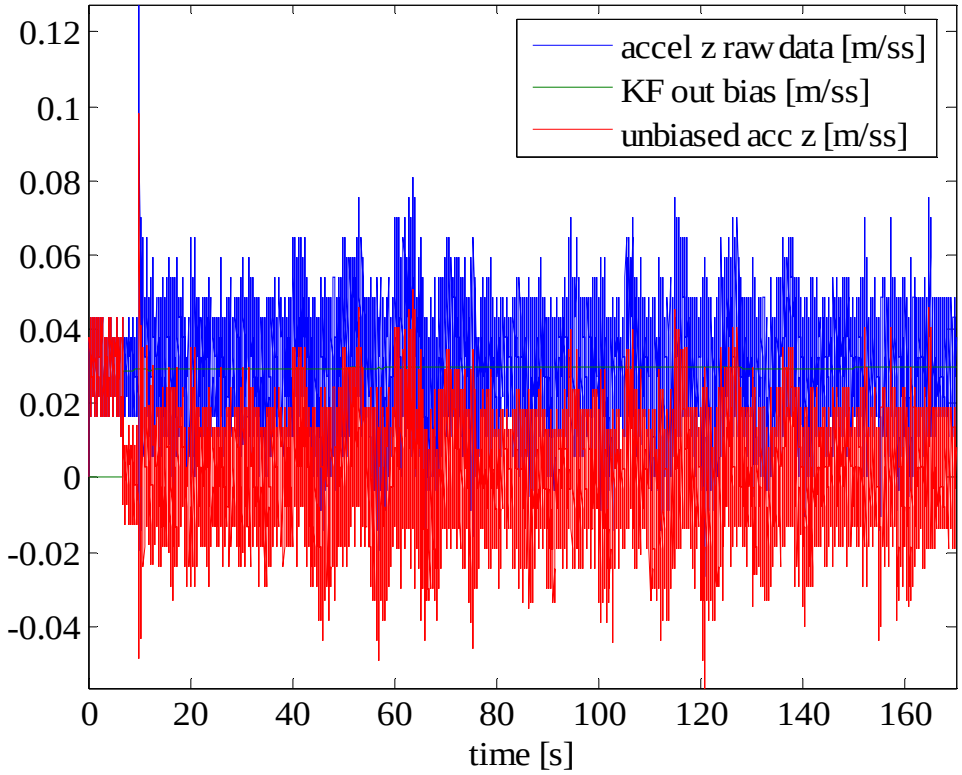

Figure 49. Experimental Results: Filtered and Unfiltered Accelerometer Data (Along Z Axis of Target Frame)

2. L-Shaped Docking Maneuver

This test is the most important test performed. To emulate an actual spacecraft docking, the test was designed to have the vehicle approach the target at a very slow rate, as to avoid any possibility of a collision. The test occurred in several stages, as seen in Figure 50. Stage 0 is a warm-up phase where the vehicles remain stationary and the Kalman filter is allowed to converge on a navigation solution. This is done for 10 seconds. In Stage 1, the vehicle moved parallel to the target vehicle until it was aligned with the target at a distance of 2 meters away. It was required to accomplish this within 15 seconds. In Stage 2, the vehicle approached the vehicle at a rate of 5 centimeters every 5 seconds until it was 0.2 meters away from the target. To achieve this, the vehicle was required to achieve scheduled waypoints. In Stage 3 the vehicle approached the target at a slower rate, creeping 5 centimeters every 10 seconds, until the vehicle was 5 centimeters from the target. In Stage 4, the vehicle made its final approach, taking 10 seconds per point to reach the 2.5, 1.0, and 0.0 centimeter waypoints. In Stage 5, the

docking mechanism was autonomously commanded to engage the target. At this point, the thrusters and reaction wheel are disengaged. This occurs 260 seconds into the simulation.

The results of the docking were definitively successful based on the physical achievement of the chaser vehicle docking with the target vehicle. The simulation results are shown from Figure 51 to 59. Figure 51, 52, 53, and 54 show the spatial position and orientation of the chaser vehicle throughout the scenario. One note from the plots is that there is higher oscillatory frequency in lateral position during the initial approach than at latter stages. This appears to be error carried over from the alignment stage. Another feature found in the plots is the effect that the docking mechanism engagement had on the chaser vehicle. Since control is discontinued during that stage, the chaser is literally left to its own devices, as the docking mechanism imparts a torque on the chaser as it grapples the receiver located on the target vehicle. The jolt can be seen in various forms throughout the plots over the last ten seconds of the experiment.

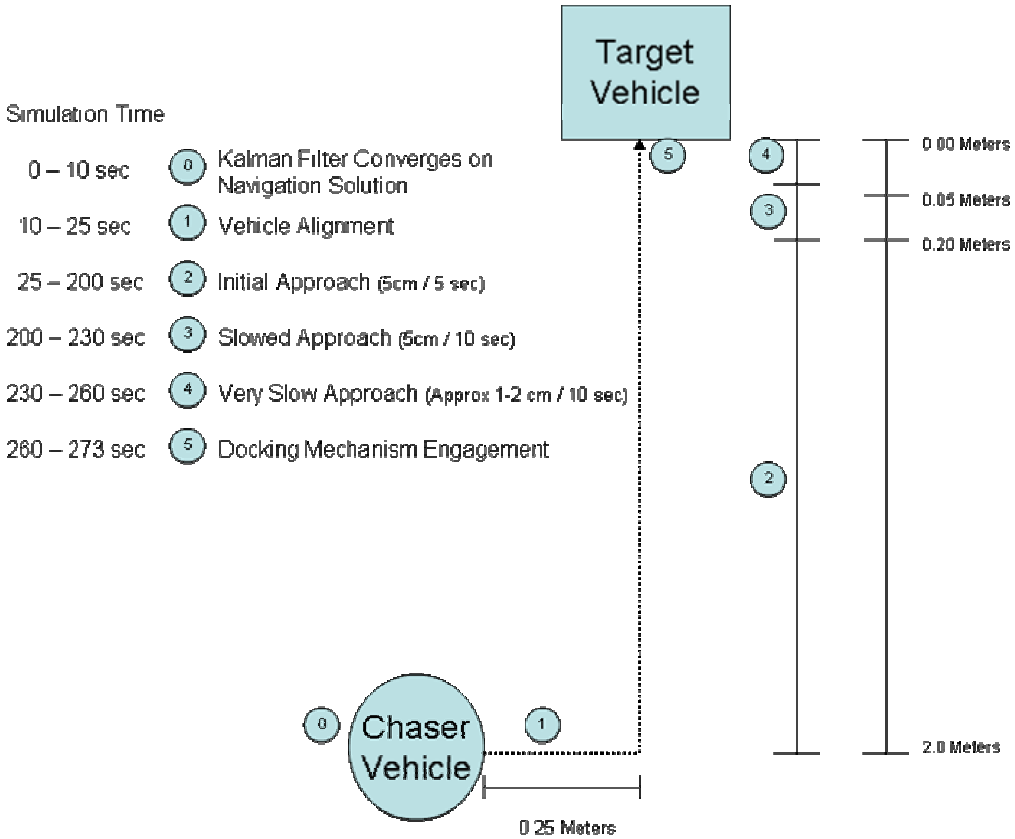

Figure 50. L-Shaped Reference Docking Maneuver Diagram

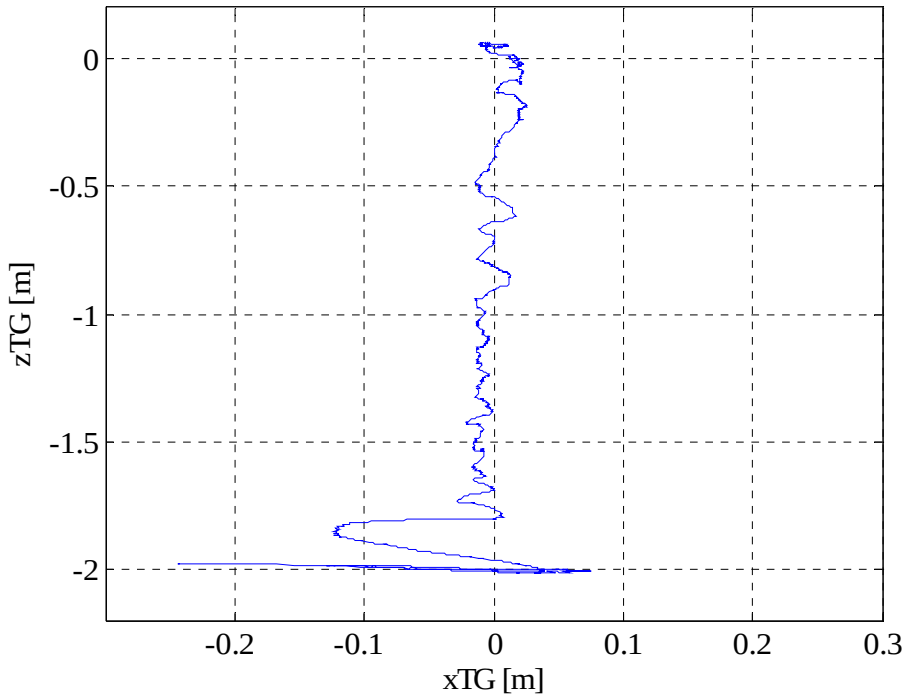

Figure 51. Experimental Results: Position of the Chaser Vehicle During the L-Shaped Docking Maneuver

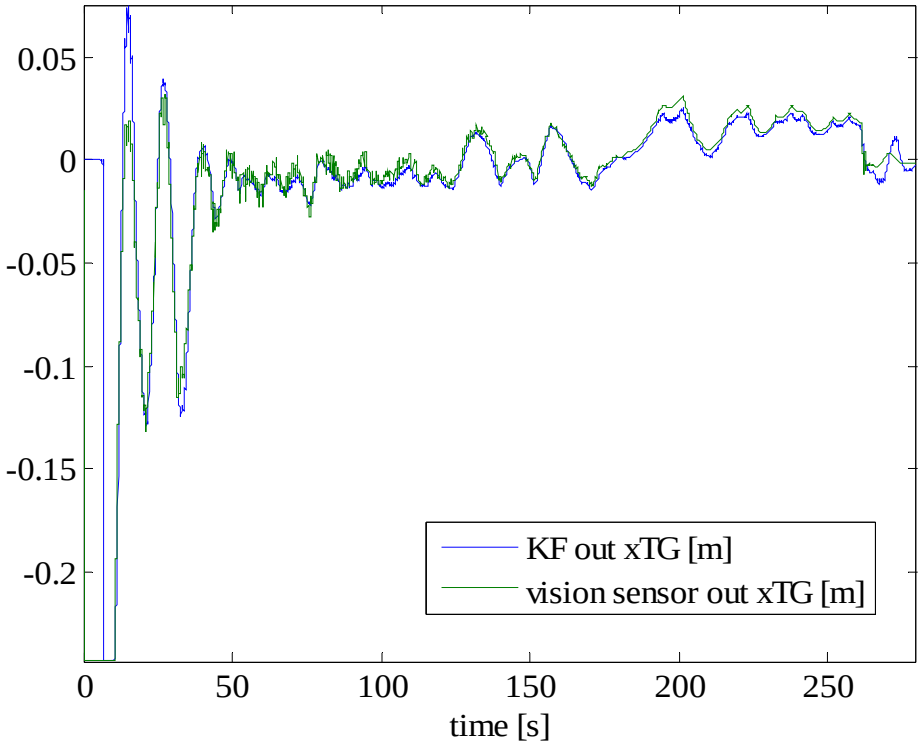

Figure 52. Experimental Results: Chaser Orientation with Respect to the Target Vehicle Using Vision Sensor and Estimation (in Degrees)

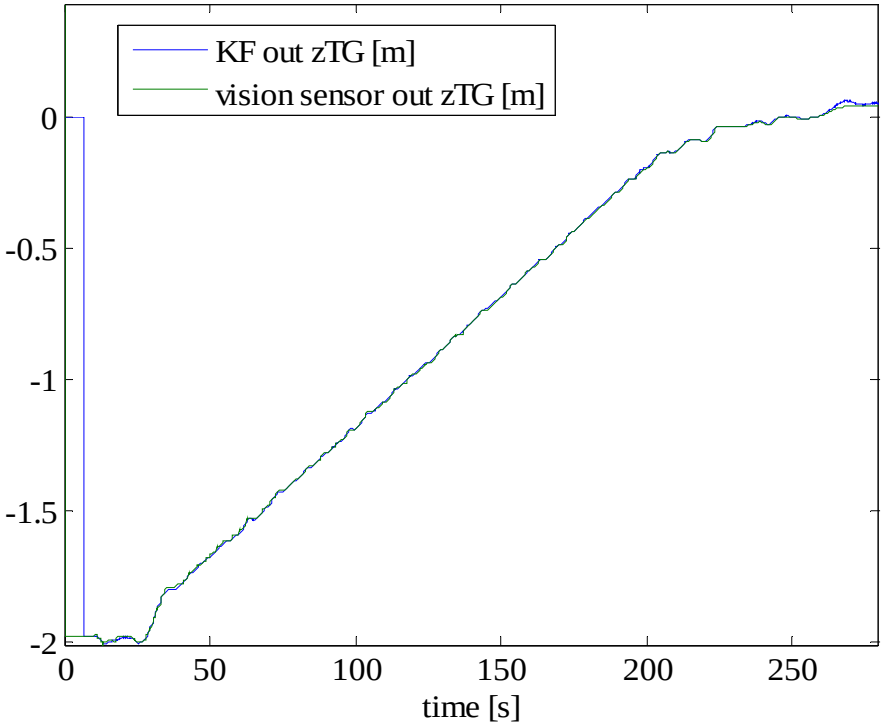

Figure 53. Experimental Results: Ranges from Chaser to Target Using Vision Sensor and Estimation

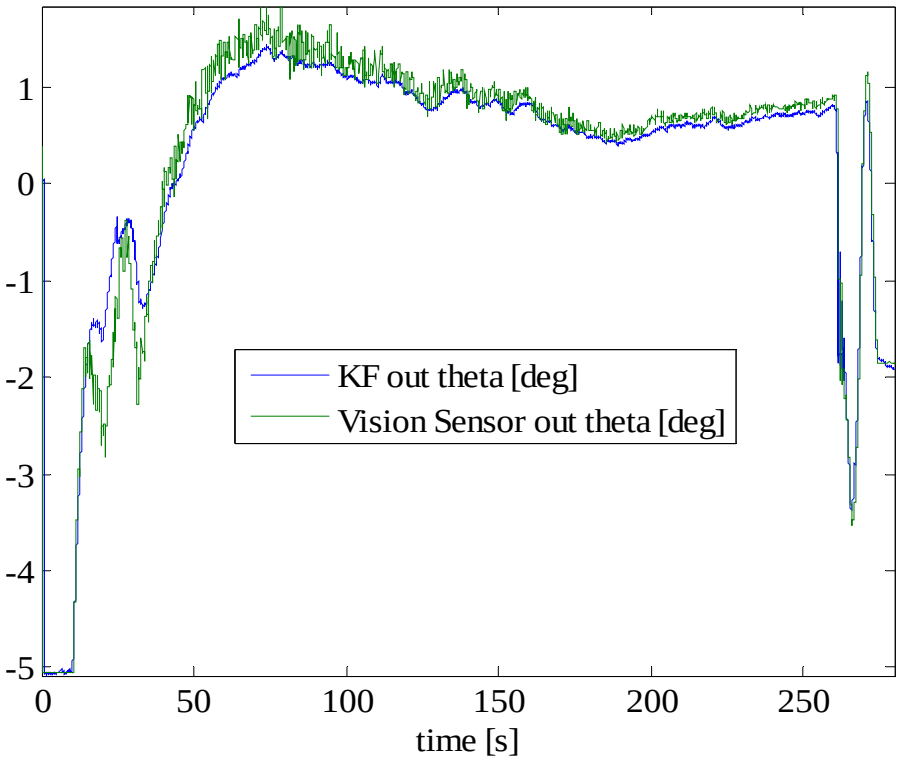

Figure 54. Experimental Results: Relative Orientation of Chaser with Respect to Target Using Vision Sensor and Estimation

Again, the actuator commanding is shown in Figure 55 for the thruster pairs and in Figure 56 for the reaction wheel. Thruster Pairs I and II refer to the forward and aft thrusters while Pairs III and IV refer to the port and starboard thrusters. The distribution of forward and aft thruster firing show that the vehicle is working to maintain a slow controlled approach through meeting each set-point along the path. However, the relatively extra work performed by Pairs I and II have little impact on Pairs III and IV and do not cause them to work at the same rate. The sparser spread of those thrusters shows that the vehicle is staying within its deadband. Possible reasons for the vehicle leaving the deadband are imperfections in the floor, causing a minor lateral drift of the vehicle, and misalignments of thruster Pair I and II, causing inadvertent lateral force. Either way, the errors are minor enough not to adversely affect the performance of the vehicle. Again for the reaction wheel, since its control is at a much higher bandwidth, the mass of color comes as expected.

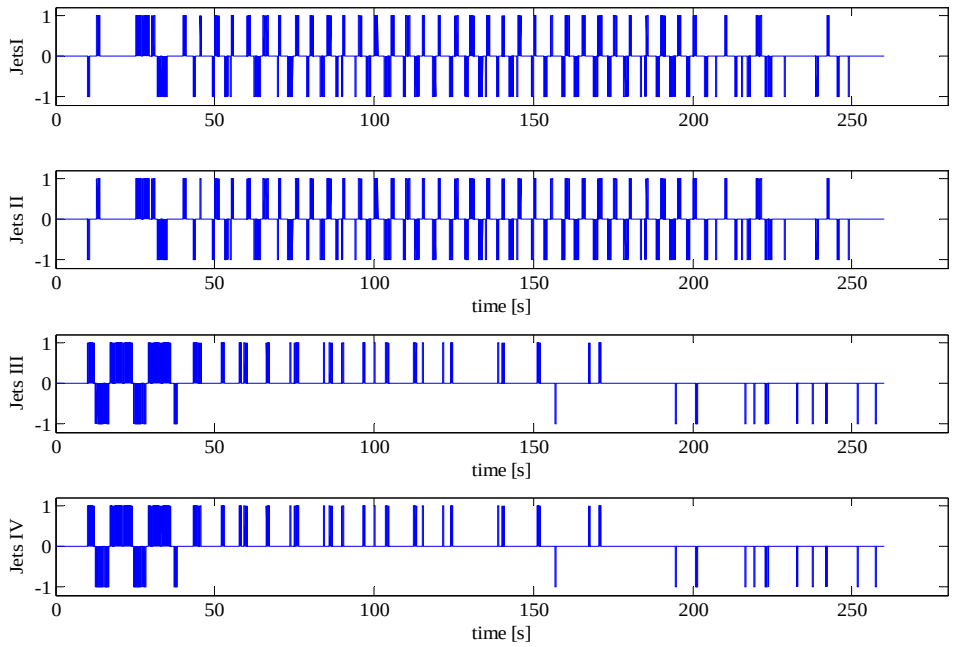

Figure 55. Experimental Results: Commands to Each Thruster Pair

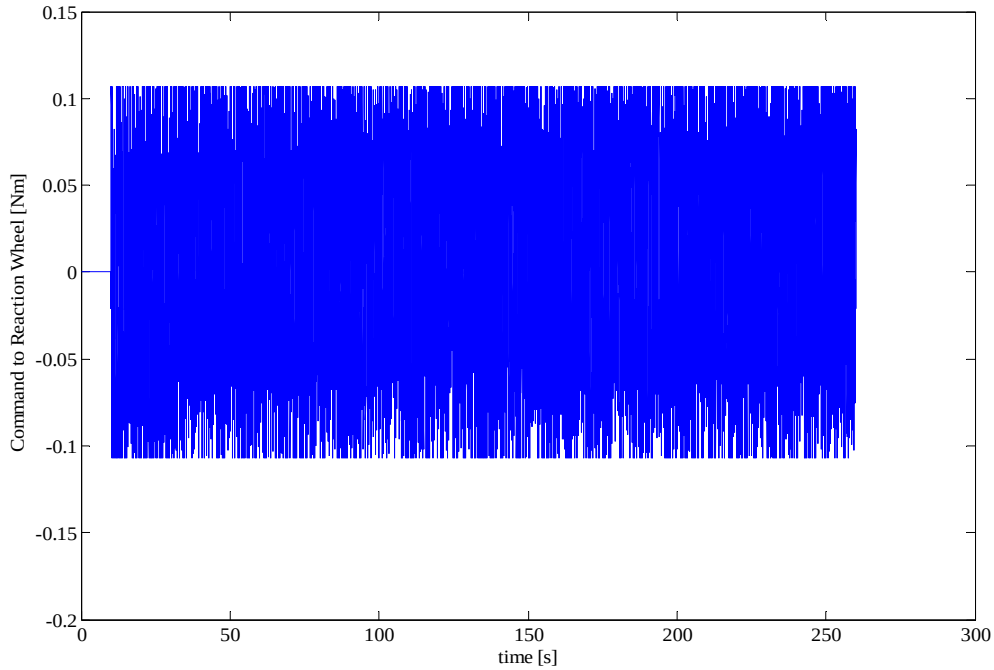

Figure 56. Experimental Results: Commands to the Reaction Wheel

The rate gyro and accelerometer data is shown before and after the bias is filtered out, below in Figure 57, 58, and 59. As was discussed in the previous chapter, the Kalman filter looks to be a major contributing factor to the success of this simulation.

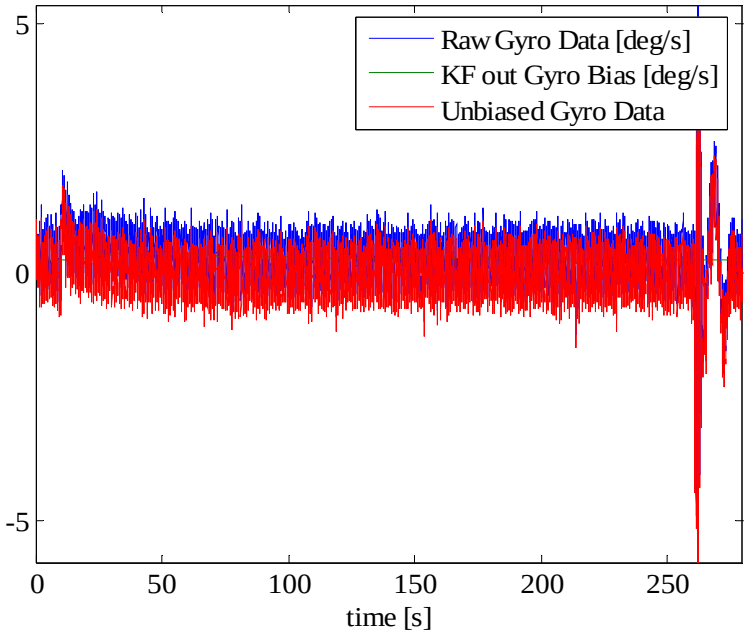

Figure 57. Experimental Results: Filtered and Unfiltered Rate Gyro Data

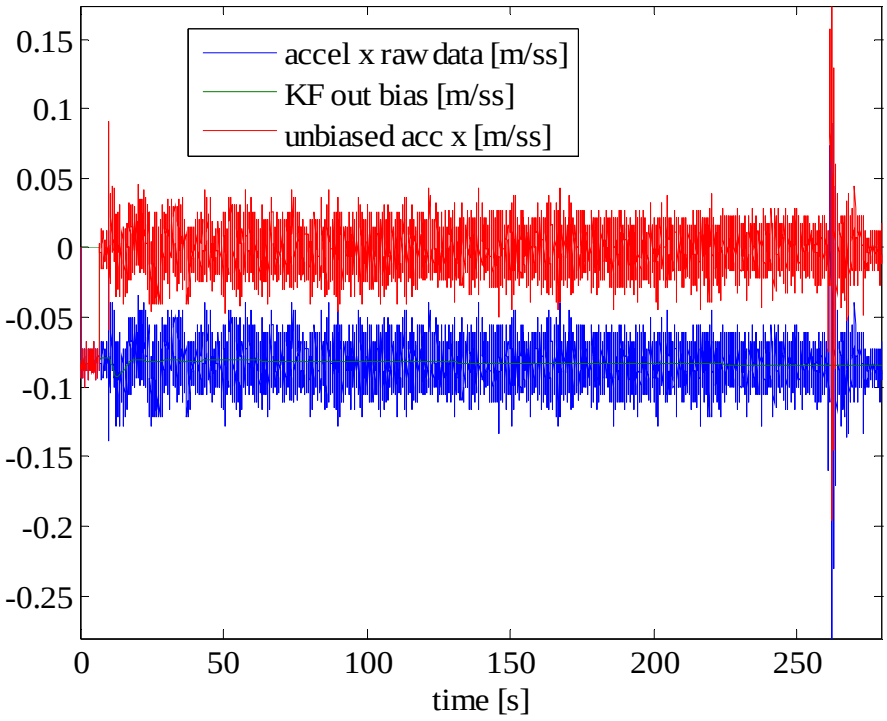

Figure 58. Experimental Results: Filtered and Unfiltered Accelerometer Data (Along X Axis of Target Frame)

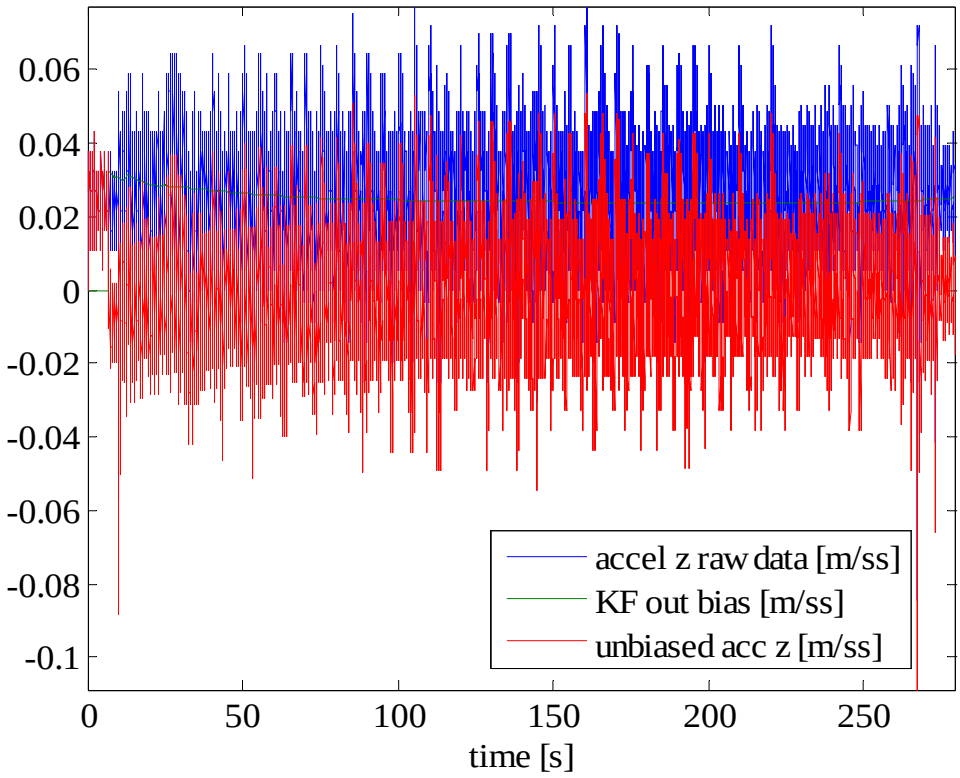

Figure 59. Experimental Results: Filtered and Unfiltered Accelerometer Data (Along Z Axis of Target Frame)

VI. CONCLUSION

A. SUMMARY OF WORK

The objective of this thesis was to advance the study of autonomous docking and spacecraft servicing techniques through the development of the AUDASS II testbed and the Spacecraft Robotics Laboratory. The testbed's supporting architecture, in terms of the flat floor and the indoor-GPS, has been developed to the point where future endeavors can focus far more attention to the advancement of the control strategies and the utilization of cutting-edge sensor technology. The powerful, yet compact computers built for this vehicle utilized the emerging PC/104 technology. Their flexible and stackable design will allow future researches the ability to easily modify the existing setup for increased computing capability. The navigation sensors and sensor data fused together with the Discrete Kalman Filter were proven to provide navigation solutions up to 1 centimeter accuracy. Finally, autonomous docking experimental tests were successfully accomplished.

B. FOLLOW-ON WORK

Although basic control of the chaser vehicle was established in a manner that led to a successful docking maneuver, further development is planned to create a more realistic scenario.

1. Fluid Storage Deck

A fluid storage deck may be integrated into the vehicle architecture. The fluid storage deck will contain a tank used in transferring fluid to the target vehicle via the docking mechanism. This function will be used to simulate the act of transferring propellant in a satellite refueling mission. The fluid could be gravity fed into the docking mechanism for simplicity. The flow would be triggered by a solenoid, operated by the control computer through the digital I/O board.

2. New Vision Computer

When the new vehicle was initially constructed, the vision computer was envisioned to operate with a Pentium 4 processor to allow for the camera software to function at a higher bandwidth and to support simultaneously functioning software programs. Only one computer with PC/104 or PC/104 Plus technology was found to utilize the Pentium 4 chip, an Advantech PCM-3380. After some extensive testing with the device, it was discovered that the mother board required ATX power. However, ATX power supplies not only were not-stackable in the PC/104 format, but the smallest commercially available ATX power supply was twice the size of the vision computer stack. From that, it was concluded that embedded PC/104 technology has not matured to the point where it can utilize a Pentium 4 chip. The high-end commercial PC/104 market should be continually monitored to determine when a PC/104 has advanced to meet the AUDASS II's processing needs.

3. Advanced Control Goals

As the title of this work alludes to, only cooperative vision navigation was addressed in this thesis. However, autonomous docking scenarios in space might not have this luxury. Experiments in non-cooperative vision navigation will need to be addressed, such as a docking maneuver with a free-floating target vehicle and/or with a spinning target vehicle. Experiments in control will also need to be conducted during a spacecraft servicing mission where the mass and moments of inertia for both of the free-floating target and chaser vehicles are constantly chasing.

APPENDIX A

Below, Table 5 describes a typical epoxy floor installation timeline as provided by Precision Epoxy Systems. The reader will notice the abbreviated schedule compared to that which was described in Chapter II. Under the proper conditions, an epoxy floor can be installed in minimal time. (Ref. [8])

1st Day
Select location in shop and clear out to expose floor allowing extra space for adjoining epoxy mixing shop. This will basically be two car stalls or similar size area.
Measure and mark dimensions on floor at corners and chalk lines to establish surface plate perimeter allowing for an extra inch of width and length.
With a 3/16 inch masonry bit and a hammer drill, mount 1 to 2 inch 'L' angle aluminum using 6-8 plastic anchors and special zinc screws to form the retaining walls of the surface plate.
Take a rotary saw with masonry blade and cut a 1/4 inch deep groove 4 to 6 inches outside the aluminum perimeter on all sides of the surface plate that needs to be ramped to floor.
The floor area inside the plate needs to be prepped. This may involve additional consulting in some cases, but basically would be tack-ragged with denatured alcohol to clean any contaminates such as oil, grease, etc., then sanded with a rotary floor sander. The area is then vacuumed for dust and debris. Additional treatment of expansion joints and/or cracks will be needed.
Using standard 2 inch duct tape, tape a waterproofing seal around the inside perimeter of the aluminum and floor. Area must be water tight to contain the fluid self leveling epoxy.
Now you're ready for the 1st pour of **Floor Plate Epoxy FP-85**. This 2-component, self leveling, pigmented 100% solids epoxy system is mixed and poured to a depth of 1/4 inch at a rate of 6.4 square feet per gallon to create the structural body coat of the surface plate. This coat will establish the level plane of the surface plate in relation to your concrete floor.
Spread epoxy with squeegee or trowel to aid leveling if needed.
While the epoxy is leveling and starting its curing cycle, you must assist with air release because of surface tension. Air bubbles come from several sources and need to be minimized as much as possible; when measuring and pouring epoxy components into containers, when stirring components and when pouring mixed epoxy into surface plate area. During the 1st pour, air is also released from the concrete as the epoxy penetrates in. The amount of this air source is determined by the concrete's degree of porosity. This factor alone makes one pour application impractical as air is still releasing as the epoxy cures. To release air, take a propane torch and wave the blue flame over the surface like a wand. This is called torching and you will be able to see the air release. Mounting of lights around the area will enhance reflection making bubbles more easily seen. The epoxy is not flammable; however, should you dip the flame into the epoxy, it could leave a charred or burnt looking spot which should be dipped out before continuing, especially on the next two coats. This is a base coat with no cosmetic concerns.
2nd Day
The 1st pour of **FP-85** epoxy has been allowed to cure and you can now see the relation of the floor to the surface plate. Any high spots of the concrete floor will dome up like islands in a lake. These areas should be ground down flush with the epoxy plane. Any low areas of the concrete floor will show at the aluminum perimeter. It must be determined if the floor area will level with the 2nd coat or should an additional base coat be made.
At this point, all foot traffic from now to completion onto surface plate area should be with clean, oil and grease free shoes wiped off on a towel or mat at edge of work area.
Sand entire area to profile plate. Vacuum thoroughly all dust and debris. Then tack rag with denatured alcohol.

Special attention needs to be given to holes in 1st pour left by air bubbles. Mix a micro batch of epoxy and hand pour to fill; should epoxy continue to flow through hole, it will need to be plugged.
Now you are ready for the 2nd pour of **Floor Plate Epoxy**. The **FP-85** epoxy is mixed and poured to a depth of 3/16 inch at a rate of 8.5 square feet per gallon to create the cosmetic color coat of the surface plate.
Spread epoxy with squeegee or trowel to aid leveling if needed.
This coat must then be rolled with a special "Spiked Roller" to assure a uniform color between batches.
Area is then torched to release air. Air bubbles will be at a minimum with the first pour sealing off the concrete.
3rd Day
The 2nd pour of **FP-85** epoxy has been allowed to cure. Now using the 'L' angle aluminum as a guide, mount the special 3/16 x 1/2 inch 'L' angle zinc strips to the surface plate. The zinc strips are again mounted using a 3/16 " masonry bit, hammer drill, plastic anchors and screws. A vacuum is also necessary to clean drill dust as you go to assure proper seating of the zinc. This strip will reduce the plate size by 1 inch at length and width.
After zinc has been mounted, remove "L' angle aluminum and duct tape, then detail perimeter as needed.
Grind or sand to clean concrete floor areas between surface plate outer wall and saw cut made the 1st day in floor.
Vacuum all dust and debris around work area. Then tack rag floor perimeter with denatured alcohol.
Tape outside perimeter of saw cut with 2" masking tape. Tape inside perimeter using paper to protect plate.
Apply **IG-100 Epoxy** penetrating primer system to concrete perimeter (Consult Technical Bulletin on IG-100 Epoxy).
Trowel apply **Epoxy Santex System** to create access ramp from surface plate top to flush with concrete floor in matching or contrasting color. The saw groove made in the concrete allows the **Santex** to be troweled flush with the concrete while maintaining proper structural thickness. The saw groove also creates a detailed "cut-off" point. **Santex** is troweled off even with the 3/16" zinc strip lip which will be the surface plate top.
After rough troweling into place, pull masking tape off floor only and clean up excess **Santex**. Detail trowel to finish access ramp.
4th Day
Remove tape and paper from inside perimeter and detail cured **Santex** as needed with sander, grinder or scrape.
Apply team, sponsor, corporate or personal logos or graphics as required. Reference points or measurements can also be installed if desired. Anything that you wish laminated underneath the clear top coat would be applied at this time.
Tack rag entire surface plate for final cleaning. Normal scratches in color coat will disappear with clear coat pour; however, discolorations such as **Santex** color, shoe or vacuum scuff marks, bugs, etc. will show if not detailed properly.
Now you are ready for the 3rd pour of **Floor Plate Epoxy**. The **FP-80** 2-component, self leveling, clear 100% solids epoxy system is mixed and poured to a depth of 3/16 inch at a rate of 8.5 square feet per gallon to create the perfectly level, perfectly flat chemically resistant cosmetic top coat of the surface plate.
Spread epoxy with squeegee or trowel to aid leveling as needed.
This coat must then be rolled with a special "Spiked Roller" to assure a uniform blend between batches.
Area is then torched to release air. All air must be released in order to achieve the maximum cosmetic advantages the **Pro Surface Plate** has to offer.
FP-80 epoxy is then brush applied to seal **Santex** access ramp. Keep airborne dust, debris, fumes, bugs, etc., to a minimum during this pour. Surface plate should be allowed to cure for 7 to 10 days depending on temperature before using to its full capacity.

Table 5 Manufacture's Directions for Constructing an Epoxy Floor (After Ref. [8])

APPENDIX B

The following is the laboratory procedure (see Table 6) for configuring the Indoor GPS Setup configuring a software template on a new computer. It also includes a procedure for collecting data once the software template has been built.

Note that the following direction assumes the use of an RS232 serial port connection from the receiver to the host computer and that the software has been loaded on the host computer. Note that the transmitter files provided by Arc Second must be loaded on the computer as well. For the software to load properly, Microsoft.net software (available free from the Microsoft website) should be installed prior to installing the WorkSpace software For initial operation, the transmitters must warm up for approximately 30 minutes prior to use. Note that Sensor refers to the equipment viewing the transmitter, and Receiver refers to the equipment which transmits the signal to the processor.

1. Open WorkSpace Wired.
2. Create a new 3DI Configuration.
3. Starting with the 3DI Visualization Tab, right click on "Setup Plans" and enter the wizard.
4. Click next, name the model, and click next.
5. Load the transmitter files by selecting "Load from ASD file..." Load the ATX1639 transmitter first, followed by the TX 2140 transmitter. Click on next.
6. In the Transmitter Layout panel, enter the approximate distance the transmitters will be away from each other. Note that units of measurement will probably be in inches, although that can be changed later. Leave all other parameters alone. Click next.
7. Uncheck the two boxes on the page and click on Finish.
8. You should now see a tab under Setup Plans with Practice, Initial Transmitter Locations, with the two transmitters selected. Each transmitter will display its relative coordinates. Since ATX was selected first, its coordinates will be 0,0,0 and the other transmitter will be the x distance identified in step 6, 0,0. Remember, these coordinates are best guesses. However, the computed values change based on the setup measurements later in these instructions.
9. Select the 3DI tab at the bottom left of the screen.
10. Open the Conductor option
11. Under Configuration, add a configuration and rename it "SRL Template" (or as desired).
12. Open SRL Template and notice the options Transmitters and Zones. Highlight SRL Template.
13. Click Import to add a transmitter. Select the ATX 1639.asd file. Repeat the step for the TX 2140.asd file. In the Conductor pane, you will see ATX 1639 and TX 2140 appear. Within those paths, you can view all of its associated data such as Definition, Calibration, and Placement.
14. Below Transmitters will be the Zone option. Ignore for now.
15. Below is the Sensors option. Name each sensor that will be managed by this software with an appropriate name (i.e., Chaser 1, Target 2, etc.) by selecting Add.
16. Under Type, select Sides32 (default). Leave the remaining boxes unchecked and click OK.
17. Below Sensor is the Receivers option. Add a receiver. Rename the receiver to match with the sensor (only for consistency). As an option, match the software name with the receiver. Select the appropriate

hardware running on the PCE (receiver hardware). The newer version provided by Arc Second is Black Sun. The older version of software is called "409 PCE Receiver." Both versions may come into play. See section below about reprogramming PCE with appropriate version of software. Click OK. Repeat for each receiver.
18. Under each receiver are a set of tabs that need adjustment. They are Setup, Parameter, and Advanced.
19. Under Setup, Sensors, add the sensor directly related to the receiver. <None> will appear. Right click on <None> and select the appropriate sensor from the scroll down menu. To the left is the Port option. Select the Port that the Receiver is plugged into. For the serial port connection, it will probably be COM1, but double check. Finally to connect the computer to the receiver, check the Connected box. If the connection is good, a check should appear almost immediately. If not, you will need to troubleshoot the connection. Although every troubleshooting session will have its unique events, some good pointers are: cycle the power on the receiver, ensure the right software is supporting the receiver (compare to the previously selected software), good connections a must, call Arc Second for additional assistance.
20. Look under the parameters tab. The manual says that you should not adjust these without the help of Arc Second. This is somewhat true, but a good rule of thumb is to note the previous configuration and adjust the noise floor, followed by the sensitivity settings in increments of 50. Note that the numbers are somewhat arbitrary. To bring the receiver within tolerance, consider matting down the base of the sensor with paper or cardboard. Also consider dimming the fluorescent lights as their oscillatory nature can interfere with the signal. The default setting for sensitivity is 1000. The AUDASS II was adjusted it to 773. The default setting for Noise Floor is 4000. The AUDASS II was adjusted to 2700. Note that these values are not necessarily optimized, and can still be improved via trial and error. No adjustment is required to the Narrow and Wide fields.
21. The Advanced tab is left alone.
22. Returning to the Setup tab, press the button, Set Configuration, to set the configuration.
23. In the left pane below the Receivers path is the Servers path. This allows the user to select the type of data needed to be collected for each sensor. To gain position data, click Add and select Single Point Server. Rename it to "Chaser 1 XYZ," for example. Optional: To gain angular data, click Add and select Azimuth and Elevation Server and rename as required. For each Server, select the sensor related to that server. The Max Standard Deviation metric should be 0.0005 which translates to 50 Microradians as described in the manual.
24. For the Point Server path, there are some important points to discuss. There is a box for Precision. For streaming data, click on Precision. Then check Streaming. See Help, Manual or Arc Second for more details on customizing the data stream.
25. Below the Server Path is the Connections Path which allows a remote computer to observe a computer running the software. It is not required for SRL applications.
26. Below the Connections Path is the Display Path. Here you can set the base units of the system. The SRL is a metric only lab. Meters and Radians are its standards.
27. Within the Display Path is the Transforms path. The Transforms options is useful to translate or rotate the coordinate reference frame to a more useful frame.
28. In the left pane next to the 3Di tab is the Message tab. This tab can be very useful when troubleshooting initial setup.
29. To perform a Setup on the system, i.e., getting a fix on the transmitters, you must take at least six measurements in various places around the floor. Two of the measurements need to be performed with a scale bar so that there is a constraint on the measurements taken. To take a measurement, from the pull-down menu, choose Setup, followed by Perform Setup. Go through the wizard to select the six points. Perform Step 1 and take the six points. Click New Sensor Observation. Press the Begin button. Collect at least 200 samples from each transmitter. Press stop. If the Std Dev1 is below 50 microradians, press next to collect the next data point. If the value StdDev1 is above the 50 microradian threshold, press Reset. This will clear this measurement. Before proceeding with the collection, try to determine the reason for the bad data (e.g., noise thresholds out of tune, multipath issues, being within ten feet of the transmitters). Once all six data points are collected continue with the wizard. Enter the constrained data points with the constraint measurement (the SRL's scale bar is 0.9 meters). The next step in the wizard will compute the bundle. If the bundle calculation is unsuccessful, retake the measurements and/or troubleshoot.
30. Data will now stream from the server. Prior to shutting down, uncheck the box from Step 19.

Table 6 Checklist for Initial Laser-GPS Setup

APPENDIX C

The following details the necessary settings to create an XPC Target boot disk and how to load the boot disk on to the Control PC. This could be especially useful if a newer version of Matlab is available. The settings (see Table 7) apply to the AUDASS II control computer only. The XPC Target and XPC Target Embedded Option toolboxes must be included with Matlab to run the XPCEXPLR function from the Matlab command prompt. The boot disk should be created on the Host computer, in this case, the SRL development computer. Once all the settings are entered, click the Create Bootdisk button located on the Configuration path.

Target PC1	
Configuration	
Target Boot Mode	DOS Loader
Communication	
Host Target Comm	TCP/IP
Target IP Address	192.168.0.3
TCP/IP Target Port	22222
LAN Subnet Mask Address	255.255.255.0
TCP/IP Gateway Address	255.255.255.255
TCP/IP Target Driver	I82559
TCP/IP Target Bus	PCI
Settings	
Target RAM Size (MB)	Auto
Maximum Model Size	16MB
Appearance	
Enable Target Scope	Leave Unchecked
Target Mouse	None
Files Created	autoexec.bat
	xpcboot.com
	xpctto16.rtb
	checksum.dat

Table 7 XPC Explorer Setting to Build XPC Target Boot Disk for Chaser Vehicle

To install the files on the Control PC, the vehicle should be powered down. With an ATX power supply, connect power supply cables to both the Control PC I/O board and to a 3.5” disk drive. Connect the IDE ribbon cable from the I/O board to the 3.5 inch disk drive. (Because there are not any Pin 1 markings on the disk drive and the IDE

cable can be installed in either direction, a trial and error approach is necessary to ensure the IDE cable is connected properly. An easy check to ensure proper installation is to verify the drive light is not constantly on. If it is, then flip the cable upside down and reconnect.) A keyboard and a monitor will also need to be connected to the Control PC and a DOS boot disk should be inserted into the 3.5" disk drive. After powering up the PC with the ATX power supply, an "A:" prompt will be seen on the monitor. Now place the XPC Target boot disk into the 3.5" drive. Copy the contents of the disk onto the Disk-on-Chip by typing "copy a:*.* c:\work" and answering yes to any questions about overwriting the same files.

APPENDIX D

The following script decodes IMU data by performing a checksum validation of the message and stripping it of its header and checksum. The code is written in C and is the basis for a Simulink Level 2 S-Function block. A dynamic link library file, used by Simulink, is created from Matlab's "mex" command.

```
/* $Revision: 1.0 $ $Date: 2004/05/14 13:20:41 $ */
/* rs232rec.c-XPC Target, non-inlined S-function driver for RS-232 receive (asynchronous)  */
/* Serial Driver that reads Crossbow output*/
/* Written by Dr. Vladimir Dobrokhodov*/
/* Version 2.0
/* Driver Modified by David Friedman on 1 Oct 05*/
////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME xbowreader_r0p0

#include <stddef.h>
#include <stdlib.h>
#include "tmwtypes.h"
#include "simstruc.h"
#ifdef MATLAB_MEX_FILE
#include "mex.h"
#else
#include <windows.h>
#include <string.h>
#include "rs232_xpcimport.h"
#include "time_xpcimport.h"
#endif
        /* Input Arguments */
#define NUMBER_OF_ARGS      (1)             // Width
#define INPUT_WIDTH mxGetPr(ssGetSFcnParam(S,0))[0]
#define NO_I_WORKS          (4)             // Current pos pointer in buffer, rec length, bufCount
#define NO_D_WORKS          (2)             // For buffer array and remains data
        /* Declare constants */
#define  MESSAGE_FOUND      77
#define  MESSAGE_LENGTH     18              // The longest possible message
#define  HEADER             255             // Crossbow Header in Hex (FF)
        /* Declare Global Variables */
static char_T msg[256];
////////////////////////////////////////////////////////////////////////////////////////////////////////////////

static void mdlInitializeSizes(SimStruct *S)
{
  ssSetNumSFcnParams(S, NUMBER_OF_ARGS);
         /* Set-up size information */
  ssSetNumContStates(S, 0);
  ssSetNumDiscStates(S, 0);
  ssSetNumOutputPorts(S, 2);                                    // Function call, data
```

```
   ssSetNumInputPorts(S, 1);                                   // Raw data
   ssSetOutputPortWidth(S, 0, 1);                              // Function call
   ssSetOutputPortWidth(S, 1, (MESSAGE_LENGTH-2));             // Data
   ssSetOutputPortDataType(S, 1, SS_UINT8);                    // Send only bytes to be decoded later
   ssSetInputPortDirectFeedThrough(S, 0, 1);
   ssSetInputPortWidth(S, 0, INPUT_WIDTH);
   ssSetNumSampleTimes(S,1);
   ssSetNumIWork(S, NO_I_WORKS);
   ssSetNumDWork(S, NO_D_WORKS);
   ssSetDWorkDataType(S, 0, SS_UINT8);
   ssSetDWorkWidth(S, 0, 128);                                 // incoming buffer for temporary storage
   ssSetDWorkDataType(S, 1, SS_UINT8);
   ssSetDWorkWidth(S, 1, 128);                                 //Remains data buffer
   ssSetNumModes(S, 0);
   ssSetNumNonsampledZCs( S, 0);
   ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE | SS_OPTION_PLACE_ASAP);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////

        /* Function to initialize sample times */
static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);     // Inherit the sample time from the model
   ssSetOffsetTime(S, 0, 0.0);
   ssSetCallSystemOutput(S, 0);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////

        /*Function:  mdlStart
static void mdlStart(SimStruct *S)
{
   #ifndef MATLAB_MEX_FILE
      //Initialize array of global variables
      ssGetIWork(S)[0] = 0;          /* set current buf pointer = 0 */
      ssGetIWork(S)[1] = 0;          /* set recLength = 0 */
      ssGetIWork(S)[2] = 0;          /* set bufCount = 0 */
      ssGetIWork(S)[3] = 0;          /* set remainsSize = 0 */
   #endif
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////

        /* Find checksum of the Payload Data */
unsigned char GetCheckSum(unsigned char *tmp)
{
   int sumbytes = 0,j=0,checksum=0;
   for (j=1;j<=(MESSAGE_LENGTH – 2);j++)
   {
      sumbytes = sumbytes +(*(tmp+j));
   }
   checksum = sumbytes % 256;
   return checksum;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
static void mdlOutputs(SimStruct *S, int_T tid)         // Function to compute outputs
{
  unsigned char checksum=0;                             // array of char data = result of parsing procedure
  int i=0,j=0,notenoughbytes=0;                         // pure counter and Boolean value of EnoughBytes
  int numBytesAvail=0, flag=0,message_status=0;
  int serbufCount;                                      //number of bytes available in the port
  int_T temp[1024];                                     //make an alias for the uPtrs

  // Assign the values of global variables
  unsigned char *buf = (unsigned char *)ssGetDWork(S, 0);          // char allocates bytes from serial port
  unsigned char *remainsdata = (unsigned char *)ssGetDWork(S, 1); // char to allocate remain bytes
  int *current   = ssGetIWork(S);                       // current = adds of current pointer position in buffer
  int *recLength = ssGetIWork(S) + 1;                   // recLength = adds of received data length
  int *bufCount = ssGetIWork(S)+ 2;                     // count number of useful bytes in buffer.
  int *remainsSize = ssGetIWork(S)+ 3;                   // count size of the inner loop remains

  InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

  // Get the number of bytes available in the port
  serbufCount = ( int_T)(ssGetInputPortWidth(S,0));
  //Put all available bytes to the temp
  for (i=0; i<serbufCount; i++)
  {temp[i] = ( int_T)(*uPtrs[i]);}

  *bufCount = serbufCount + *current;
  // place new chunk of data at the end of previously unprocessed data
  // 'buf' also keeps the remains of previous step ending at the *(buf+current),
  // where 'current' is the position of last byte
  for (i=*current;i<*bufCount;i++)
  {
    *(buf+i)=(unsigned char)temp[i-*current];
  }
  i=0;//move local pointer i to the beginning of buf

  // Start Main Loop
  {
    // Iteration to find 1 byte header {FF}={255}
    while (  ( *(buf+i)!= HEADER) && ( i< (*bufCount-1)) )
    {
      // by shifting along the *buf array to find a header
      i = i + 1; //counter i holds the position of first header byte
    }
    // check availability of at least one complete message with correct checksum
    if (( *(buf+i)== HEADER) && (i+MESSAGE_LENGTH)<(*bufCount))
      {checksum=GetCheckSum(buf+i);}               // get checksum
    else
      {notenoughbytes=1;}
    if(*(buf+i)== HEADER && *(buf+i+MESSAGE_LENGTH-1)==checksum && notenoughbytes!=1)
      {message_status=MESSAGE_FOUND;}
    else
      {i+=1;}
    if(message_status==MESSAGE_FOUND && notenoughbytes!=1)
    // If a message is complete and identified then send it out.
    {                                   // PORT 1: Data is of j-bytes length starting from 1st => see data;
      memcpy(ssGetOutputPortSignal(S,1),buf+i+1,(MESSAGE_LENGTH-2));
      //Port 0: Function call
```

```
      ssCallSystemWithTid(S, 0, 0); //issue done pulse to outport 0 then shift the index inside the buf
      i=i+MESSAGE_LENGTH;     //include the length of payload, HEADER and CHECKSUM
      message_status=0;               //bytes at the end reset message status
      flag=0;
    }
        /* Check if there are enough bytes for a minimal message */
    if((i+1+j+1)>=(*bufCount)){notenoughbytes=1;}
        /* Check if we do not have enough bytes. */
        /* If there are some bytes remaining, then save them and leave the procedure. */
        /* Substitute remaining bytes from this step at the beginning of "buf." */
        /* Save remaining bytes into the "buf" and shift current to the end of a new buf */
    if ( notenoughbytes==1 || (i<(*bufCount) && i>=(*bufCount-MESSAGE_LENGTH)))
    {
      *bufCount=*bufCount-i;      //number of remain bytes
      notenoughbytes=1;
      for (j=0;j<*bufCount; j++)
      {
        *(buf+j)=*(buf+i);
        i++;
      }
      *current=*bufCount;
    }
    message_status=0;
  }
}                                       // End of mdlOutputs
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/* Function to perform housekeeping at execution termination */
static void mdlTerminate(SimStruct *S)
{}
#ifdef MATLAB_MEX_FILE        // Is this file being compiled as a MEX-file?
#include "simulink.c"                 // MEX-file interface mechanism
#else
#include "cg_sfun.h"                  // Code generation registration function
#endif
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

APPENDIX E

The following Simulink code in Figure 60 was developed to measure the AUDASS II's reaction wheel speed. A Hall Effect sensor, embedded within the reaction wheel, provides the square wave necessary to determine the reaction wheel speed. The code searches for rising triggers from the sensor and counts them over a 1-second period. The tachometer then coverts those triggers per second into RPMs using the conversion rate of "4 Rising Triggers = 1 RPS = 60 RPM." Knowledge of the wheel speed will prevent the control from inadvertently damaging the wheel mechanism. (Ref. [12])

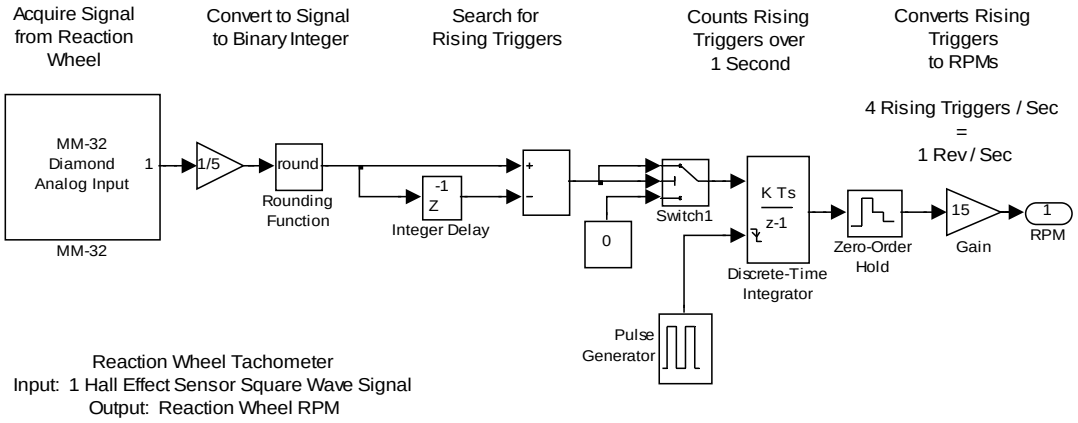

Figure 60. Simulink Diagram of a Reaction Wheel Tachometer

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Ulman, M, *Experiments in Autonomous Navigation and Control of Multi-Manipulator, Free-Flying Space Robots*, Ph.D. Dissertation, Stanford University, March 1993.

2. Wilson, E., Rock, S., *Neural Network Control of a Free-Flying Space Robot*, Simulation, May 1995.

3. Marchesi, M., Angrilli, F., Venezia, R., *Coordinated Control for Free-flyer Space Robots*, 2000 IEEE Conference on Systems, Man, and Cybernetics, 8 October 2000.

4. Machida, K., Toda, Y., Iwata, T., *Maneuvering and Manipulation of Flying Space Telerobotics System*, 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, 7 July 1992.

5. Howard, R., Bryan, T., Book, M., *An Advanced Sensor for Automated Docking*, Digital Avionics Systems, October 2001.

6. *MSC Software*, Retrieved 10 October 2005 from http://www.mscsoftware.com/success.

7. Porter, R., *Development and Control of the Naval Postgraduate School Planar Autonomous Docking Simulator (NPADS)*, Master's Thesis, Naval Postgraduate School, September 2002.

8. *Precision Epoxy Products*, Retrieved 1 October 2005, from http:/www.precisionepoxy.com

9. Arc Second, *Indoor GPS Workspace, Advanced Metrology Edition, User's Guide Version 6.0*, 2005.

10. Shay, T.J., *Design and Fabrication of Planar Autonomous Spacecraft Simulator with Docking and Fluid Transfer Capability*, Master's Thesis, Naval Postgraduate School, December 2005.

11. Starsys Research, *Mechanical Docking System Manual*, November 2001.

12. Ball Aerospace, *20.3 N-m-s Reaction Wheel Interface Control Drawing*, March 2000.

13. Diamond Systems Corporation, *Diamon-MM-32-AT User Manual V2.64*, 2003.

14. Real Time Devices USA, Inc., *DMR24 Mechanical Relay Output Board User's Manual*, 1997.

15. *PC/104 Embedded* Consortium, Retrieved 15 October 2005, from http://www.pc104.org.

16. VersaLogic Corporation, *EPM-CPU-10 Reference Manual*, 2004.

17. Tri-M Engineering, *PC/104 Vehicle Power Supply Technical Manual*, June 2005.

18. Mathworks, *Matlab Manual*, 2005.

19. Romano, M., *On-the-Ground Experiments of Autonomous Spacecraft Proximity-Navigation Using Computer Vision and Jet Actuators*, Advanced Intelligent Mechtronics, Proceedings on 2005 IEEE/ASME International Conference on, 2005.

20. Crossbow, *IMU User's Manual*, 2005.

21. Houska, J., *RTDemo*, Retrieved 5 October 2005, from http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=262

22. Dobrakhodov, V., Lizarraga, M., *Developing Serial Communications Interfaces for Rapid Prototyping of Navigation and Control Task*, AIAA Modeling and Simulation Technologies Conference and Exhibit, August 2005.

23. Crassidis, J.L., Junkins, J.L., *Optimal Estimation of Dynamic Systems*, CRC Press, 2001.

24. Wie, B., *Space Vehicle Dynamics and Control*, American Institute of Aeronautics and Astronautics, 1998.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. Marcello Romano
   Naval Postgraduate School
   Monterey, California

4. Dr. Vladimir Dobrokhodov
   Naval Postgraduate School
   Monterey, California

5. Khan Pham
   Air Force Research Laboratory Space Vehicle Directorate
   Kirtland AFB, New Mexico

6. Gordon Roesler
   Defense Advanced Research Projects Agency Tactical Technology Office
   Arlington, Virginia

7. Richard Howard
   NASA - Marshall Space Flight Center
   Huntsville, Alabama

8. Mike Ramy
   Precision Epoxy Products
   Douglasville, Georgia