The University of Maryland, College Park

robotics @ MARYLAND

Abstract

Tortuga IV is an autonomous underwater vehicle (AUV) designed and constructed by students from the University of Maryland, College Park as part of a student government sponsored club. It is designed primarily to compete at AUVSI and ONR's RoboSub competition, however, real world applications for AUVs such as sea floor mapping, mine detection, and underwater scientific research continuously motivate further development of similar vehicles. *Tortuga IV* is capable of serving as a development and testing platform for underwater actuation, underwater machine vision, sensor fusion algorithms, and six degrees of freedom control algorithms. *Tortuga IV* retains many of the previous design concepts of *Tortuga III* while improving significantly on the dynamics, sensing capabilities, and reliability. Advances in the software system provide expanded modularity, more sophisticated estimation algorithms, and invaluable testing and tuning software.

Authors: Chris Carlsen, Nicholas Limparis, Kit Sczudlo, Gary Sullivan, Donald Gregorich, Eliot Rudnick-Cohen, Stephen Christian, Vijay Baharani

Faculty advisor: Dr. Nuno Martins, Ph.D.

Introduction

Tortuga IV is an Autonomous Underwater Vehicle designed and built by Robotics @ Maryland, a student organization at the University of Maryland, College Park. The original *Tortuga* was created in 2006 to compete in the Tenth Annual International AUV Competition hosted by AUVSI and ONR. The club has progressed since its inception, and the fourth revision of our submersible AUV is ready to compete in the 2012 RoboSub competition. The purpose of the competition is to develop AUV technology by challenging students to construct submersibles capable of completing underwater tasks that simulate possible missions. The 2012 competition requires our robot to be capable of controlled underwater motion, manipulation of objects, navigation relative to active sonar pingers, and complex visual assessment in order to complete a range of objectives within a specified time limit.

Mechanical System

Our chassis is designed for simplicity and reliability. Featuring six separate pressure hulls mounted inside of an 80/20 frame, covered in foam, *Tortuga IV* looks more like a prototype platform than an AUV. Our objective in in designing *Tortuga IV* is functionality, not aesthetics.

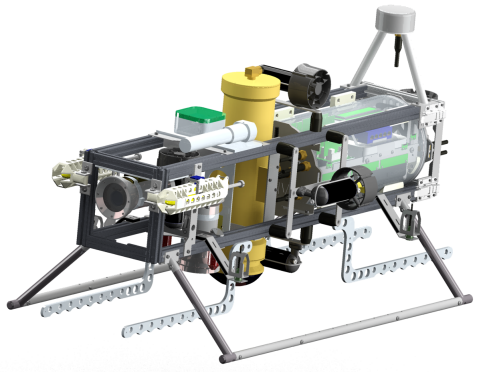

Figure 1: CAD rendering of *Tortuga IV*

Chassis

Modularly designed, *Tortuga IV*'s chassis does not maintain a constant form. During our prototyping phase, every component of *Tortuga* will be shifted around to obtain a desired center of mass. Rather than design the correct buoyancy into each individual hull, each hull is negatively buoyant to prevent the need for adding balancing mass. Currently, the grabber shown and the pneumatics housing do not represent the final positions or designs, due to some ongoing changes.

Master Hull

The master electronics hull is a remnant of *Tortuga I*, with piston seal end caps and a clear acrylic body for quick diagnostics. This design has remained unchanged for three years because its internal layout is preferred by our electronics team. Although the basic design has remained the same, *Tortuga IV*'s hull has far superior thermal properties, transmitting over 80 watts of thermal energy into the environment while keeping the electronics at a safe operating temperature. The rear end cap of *Tortuga IV* is designed with a corrugated shape and acts as a high-efficiency flat wall heat exchanger. The acrylic body of the master hull is the limiting structural factor for *Tortuga IV*, with a crush depth of fifteen meters.

Peripheral Hulls

The 2011 to 2012 year has had virtually no updates to the physical design of the main or peripheral hulls. The camera housings, Doppler velocity log (DVL) housing, and the inertial measurement unit (IMU) housing located at the rear of the main hull (also called the "magnetometer boom") continue to function and have little need to change. All peripheral hulls have a maximum operational depth of 600 meters.

Pneumatics

There are three basic types of motion that can be easily implemented underwater; pneumatic, hydraulic, and electric. We have previously used all three types with varied results. Electric motion, provided by servos and electromagnets, is prone to failure due to leaking at mechanical interfaces. Hydraulic motion, the most common among submersible vehicles, is effective but difficult to do at this vehicle scale. Hydraulic pumps are typically heavy and expensive, requiring a bellows system to compensate for external pressure changes and fluid loss. Pneumatic motion is ideal for this competition, as the operational pressure at depth is less than two atmospheres. We use compressed CO2, regulated to 100 psi absolute, which allows for the actuators to still function, even when a line or cylinder is leaking. A single sixteen gram CO2 cartridge is used to actuate up to twelve single acting, or six double acting, pneumatic pistons. These are used to release markers, fire torpedoes, and capture the wreath. This system is the result of years of submersible actuator design. We have found that a single centralized actuation system can be made far more reliable than a series of independent actuators.

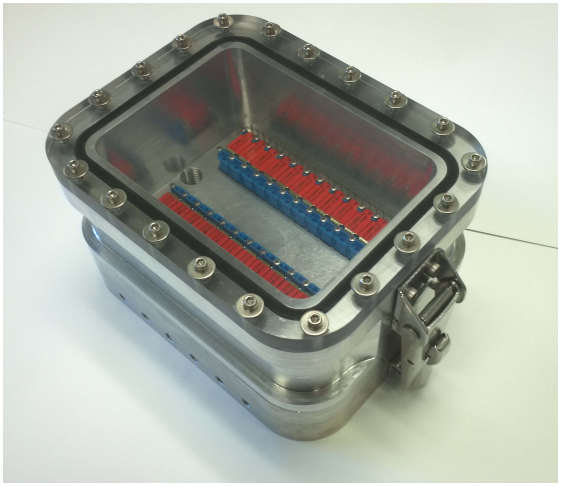

Figure 2: Sealed pneumatics housing

In 2010, we captured the sonar object using a passive grabber and released it actively. This worked well because the object was a closed shape made from small-diameter PVC. Because the wreath is a return to a partially closed shape, we returned to the passive-grab active-release model. This system uses two pneumatic cylinders to bow out the passive capture system from the craft, which causes the mechanism to disengage and drop the wreath.

Electronics

Introduction

The electrical system uses the same modular design found in *Tortuga IV* predecessors. In this design, a central board links plug-in cards, and each card has a specific set of functions. The modular design was chosen for its logical separation of functions, expandability, and ease of servicing. Our electronics stack consists of a backplane board which holds six other modules: the power regulator, battery load balancing, power distribution, actuator control, sensor, and sonar processing boards.

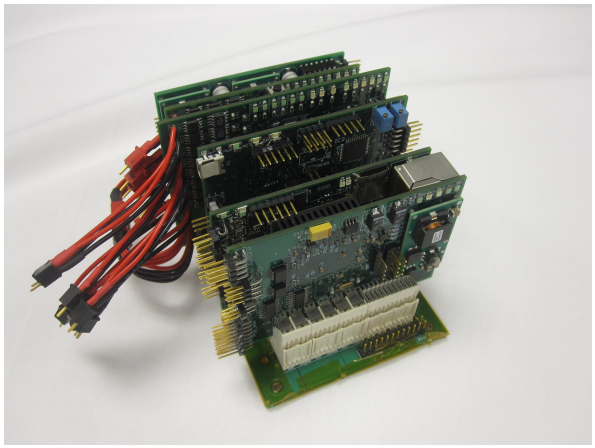

Figure 3: The backplane removed from *Tortuga IV*

Power distribution

The power system in *Tortuga IV* is separated into three major operations over five separate boards. The operations are battery balancing, distribution, and regulation. The first step,

battery balancing, is handled by the battery balancing board which takes in 28 volts from the lithium-ion batteries and keeps the draw as even as possible from each pack. The even draw mechanism disconnects batteries which have a lower nominal voltage, preventing the batteries from wasting power, ensuring a long operational period.

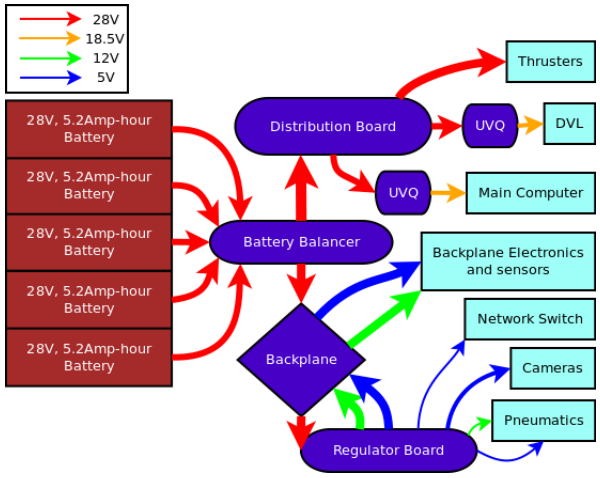

Figure 4: A block diagram of *Tortuga IV*'s power system

The next step is power distribution. The distribution board gives power to devices which require high currents, such as the motors, the Doppler velocity log (DVL), and the computer. The rest of the distribution is handled by the backplane itself, which shares the power between all of the plug-in cards. The distribution board gives *Tortuga* control over which devices are currently receiving power. This manner of power distribution does not require all devices to be active, which allows for quick diagnosis of problems and power conservation when the robot is not in the water. There is also a safety mechanism which can terminate power to any device if the software detects unsafe conditions.

The final step in the power subsystem is to regulate the voltage from the input batteries to useful, common voltages. This is handled by three separate boards. The first is the regulator board, which handles power to the backplane, hydrophones, inertial measurement units, and the router. The voltages produced by this board are 20 volts, 12 volts, 9 volts, and 5 volts. The remaining regulation is performed by the two UVQ boards, so named for their Murata UVQ isolating DC-DC converters. The UVQ boards provide the power needed for the DVL and the Mac Mini.

Currently, power into and out of the UVQ boards requires filtering to prevent excessive electrical and magnetic interference (called EMI) from interferring with other systems. This is accomplished using an inductor and capacitor pair on the input and output wires.

Software, Firmware, and Sensor Communication

Communication between the Mac Mini and the various actuators and sensors happens through a serial communications bus to a single integrated circuit (IC). This Microchip dsPIC (the master IC), distributes commands to all of the other microcontrollers, and allows any microcontroller with enough pins to interface with the current electrical system. Any microcontroller capable of a few simple protocols (I^2C, rs-232, and SPI) could easily replace the current PIC architecture, but Microchip dsPICs were chosen over Atmel or ARM microcontrollers because the electrical team was already familiar with their use.

Certain high-bandwidth sensors are interfaced directly to the computer rather than through the master IC interface. These sensors are connected directly through USB to serial devices, which allow the computer to more quickly ask the sensors for their data. The sensors on separate interfaces are the Doppler velocity logger and the two separate inertial measurement units.

Actuator Control and Communication

Currently, the actuators on the system are the six thrusters and the pneumatic system. The

thrusters are controlled through a simple I^2C interface. Although the I^2C protocol allows for all the thrusters to communicate on the same 2-wire bus, we have isolated each thruster so that a failure or power outage in any individual thruster does not affect our ability to communicate with any other thruster. The pneumatic system is run by a single dsPIC connected through serial lines to the main hull. To actuate the pneumatic system, the small pneumatic relays are opened and closed to control the flow of pressure to and from the piston chambers.

Sensors

Tortuga IV has a variety of sensors to gather data about its state and surroundings. Inertial guidance sensors inform low level control algorithms which stabilize the vehicle and allow it to execute basic motions in the water. Sonar sensors and a pair of cameras are used to determine the relative location of various competition objectives.

Localization

Tortuga IV has three types of sensors that together provide enough information to localize the robot in all three dimensions. The first sensor is an absolute pressure transducer that provides a direct measure of the vehicle's depth while remaining insensitive to temperature and pressure changes inside the hull. The second type of sensor is the inertial measurement unit (IMU). *Tortuga IV* has two separate MemSense nIMUs each containing a three-axis magnetometer, accelerometer, and gyroscope. One IMU is placed at the vehicle's center of gravity (CG) to accurately measure the gravity vector and rate of rotation. The other IMU is at the top of the vehicle to isolate it from electromagnetic interference from the thrusters and other electronics. *Tortuga IV*'s absolute orientation is computed via the TRIAD algorithm [1]. Last year, we integrated a Doppler velocity log (DVL) which measures the velocity of the robot with respect to the sea floor. By integrating these velocity measurements and taking into account our orientation, we are able to localize our position in the horizontal plane.

Sonar

Tortuga IV uses a passive sonar system to localize an external "pinger." It finds the relative bearing of the pinger by comparing the time of arrival of a pulse to four hydrophones at known positions. This is known as a "time delay on arrival" or TDOA method of localization.

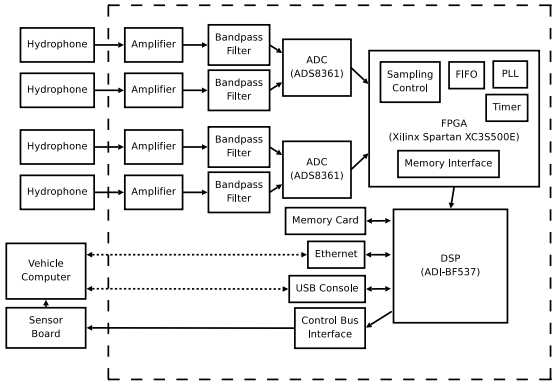

Figure 5: A block diagram of *Tortuga IV*'s passive sonar system

Signal processing

The audio pulses are read in by the hydrophones, and processed through a series of filters, passed into an analog to digital converter, and finally read in by a field-programmable gate array (FPGA). The FPGA interfaces with a Blackfin computer, which processes the information into a direction and magnitude. This is then sent back to the main computer where it is used by the software to find the pinger.

The Blackfin computer uClinux was chosen as the operating system for the DSP. uClinux provides numerous features that enable rapid development such as file system drivers, Flash card drivers, and a TCP/IP stack. It also provides a familiar coding environment for the developers of the sonar code.

Vision System

The vision system serves to give *Tortuga IV*'s software information about the locations or visual objectives in the view of the forward and downward facing cameras. The vision system input consists of two Toshiba Teli FireDragon cameras which stream video over FireWire to the Mac Mini. The two cameras are stored in housings separate from the rest of the vehicle's electronics (as shown in figure 6). The external housing provides a flat optical viewing surface for the cameras.

Figure 6: Camera housings.

The vision software makes use of the OpenCV image processing library. The vision system is split into a series of detectors, corresponding closely with the set of vision objectives. This year we have attempted to add in shape recognition in addition to the usual color detection algorithms to increase our reliability, but these have proven to be at the limit of our computing capabilities. The bin object recognition algorithms have also been updated and improved to more reliably identify the images in the bins.

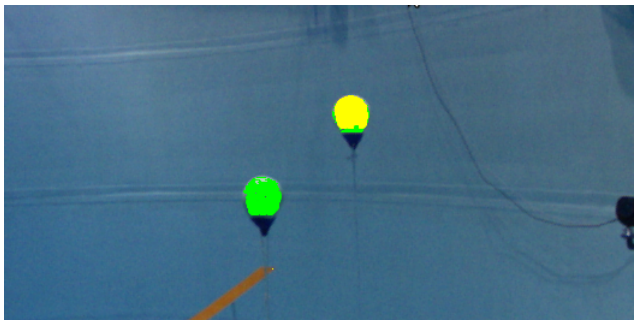

Figure 7: Debugging output from vision configuration tool.

Control

Designing an effective control system requires the careful consideration of nearly every aspect of the AUV's design. Starting with the careful attention to vehicle dynamics, the selection of accurate sensors and actuators, and the design of communication hardware to allow low-latency transmission of measurements and thruster commands, the foundation for a well-behaved vehicle is laid before any control algorithm is written.

Tortuga IV is designed with preference of maneuverability over speed, having control over all six degrees of freedom. In order to control all six degrees of freedom, we have chosen to incorporate six thrusters into the design rather than dynamically vectoring the thrust. Having six thrusters greatly simplifies the code to control all six degrees of freedom and additionally allows us to control all six degrees of freedom simultaneously. SeaBotix SBT153 thrusters were selected because they fit well with the dimensions of the robot while providing enough thrust to move at a reasonable speed. Additionally, these thrusters include integrated motor controllers and encoders which allow precise control over the thruster output.

The control subsystem software architecture has been designed with the primary goal of creating a modular object-oriented system

which can accommodate multiple control algorithms, allowing for quick switching between algorithms without recompilation. An object-oriented design significantly reduces the amount of logic required, converting semantic errors to syntax errors, which are much easier to identify. This makes development and testing of new control algorithms significantly easier and less error prone. This design also separates code into reasonably sized components, drastically improving the readability and making the understanding of logic flow much easier.

Depth Control

Tortuga IV primarily holds a fixed depth, so depth control has been implemented as a regulator as opposed to a tracking controller. The possible non-linearity that stems from the vehicle being positively buoyant can be treated as a constant disturbance due to the buoyancy design. Thus, any linear controller can be used as long as it is augmented with an integrator for constant disturbance rejection.

Several depth controllers have been implemented on the vehicle. PID, LQG, and model based observer controllers [2] are currently available to be selected at run time. Since the vehicle's buoyancy and trim characteristics are often altered for new actuators or adjustment of ballast, the robustness of the control algorithms is as important as controller performance. After comparing the various depth-control algorithms on the vehicle, a PID implementation was chosen.

Rotational Control

The rotational control algorithm is based off [3], a full three axis non-linear PD controller. The controller uses quaternions, making it singularity free (i.e. immune to "gimbal lock" problems). This rotational controller is a tracking controller and subsequently can be commanded to rotate the vehicle in any direction by either specifying a desired orientation or specifying a desired angular rate and integrating the desired orientation. A more advanced rotational control algorithm based on Egeland and Godhavn's adaptive attitude control algorithm [4] has also been implemented. As an adaptive controller, it can "learn" *Tortuga IV*'s inertia, drag, and buoyant moment properties while running to provide a significant improvement to the vehicle's attitude control.

Translation Control

Due to the DVL, we can perform closed-loop control for translation. We have selected a PID tracking controller to take feedback from the DVL and follow a trajectory in the horizontal plane. This control algorithm has been shown to perform well even without extensive tweaking during times when the robot configuration is frequently changing. Our future plans include exploring feed-forward techniques to improve the controller response.

Motions

The ultimate goal of the control system is to allow the execution of complex motions. To acheive this we have chosen a design where the artificial intelligence / planning subsystem generates a trajectory for the controller to follow based off vision data that has been processed by the estimation subsystem. This avoids the problem of fluctuating measurements impairing the performance of our controllers, which would occur if the controllers were run directly off the raw vision data.

Software

Overview

The software architecture on *Tortuga IV* is designed around a few core concepts: modularity, extensibility, and configurability. Focusing on these tenets has helped reduce development and testing time while improving software quality. We have selected C++

as the core language because of its object-oriented facilities, good performance, and easy integration with other languages. All performance sensitive code is written in C++, aside from the drivers, all of which are written in ANSI C. Code that is not performance-sensitive is written in Python because it is more expressive than C++, leading to shorter development times, and does not require compilation after modifications are made.

Modularity

Modularity comes about naturally from the use of object-oriented design. The software is divided into subsystems, shown in figure 8, each of which represents an abstraction around a particular task. Each subsystem makes use of abstract classes, each of which serves as an interface to a different physical device or algorithm, reducing both the amount of code that must be written and the number of locations where modifications must take place to integrate new devices or algorithms.

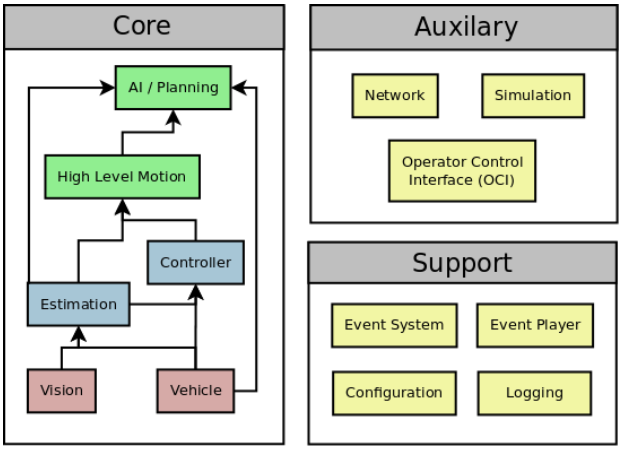

Figure 8: A Diagram of *Tortuga IV*'s software subsystem

Extensibility

Extensibility is essentially a measure of the amount of work required to implement enhancements. Even a modular system is not necessarily extensible. Extensibility requires foresight of what future enhancements will need to be implemented. Since the software on *Tortuga IV* is under continuous development and new algorithms are frequently being added, careful attention has been paid to make the design general enough to incorporate algorithms that require slightly different input. This extensibility is partly attained through our event-driven architecture. We have implemented a custom publisher/subscriber system using the Boost C++ libraries as a backend. Event-driven architecture reduces the coupling between subsystems as the location in the code where an event is published can be changed without requiring changes to the code of the subscriber.

Configurability

Configurability is an extremely important design aspect. Our configuration system allows us to select different algorithms and tune the parameters of our algorithms without the need for recompilation. Being able to change the behavior of compiled code without the need for recompilation is most important when testing and tuning our algorithms. Compilation is one of the most processing intensive operations and consequently requires a significant amount of battery power. By reducing or eliminating the need for compilation while testing and tuning, we greatly extend the amount of time that *Tortuga* can be operating in the water without the need to change or recharge batteries. We have selected YAML as a configuration language because it is human readable and writable. It also integrates very well into Python and C++.

Development Practices

In order to facilitate development in a team environment, we use Subversion to track revision history, Trac for issue tracking, and CMake as a build system generator. These tools reduce development times, allow multiple developers to work simultaneously on the

same code, and provide a system in which to solve bugs systematically while providing documentation of the solution. The build system generator allows different environments to easily build the code with minimal work required from the developer.

Debugging and Tuning Tools

There are several debugging and tuning tools which help speed development and analysis of the software. The first is the event player which reads logs of received events during a dive operation and can replay them so developers can more closely examine what occurred during a dive. This system enables developers to more easily debug errors received during a dive. Another tool is the real-time state visualizer, which enables any networked computer to plot time series of any aspect of *Tortuga*'s sensor suite. This immediate feedback is invaluable for tuning estimation algorithms and controllers.

Operator Control Interface

Tortuga IV's dive operations are supplemented by an Operator Control Interface (OCI) program designed to allow simple interaction with all of the subsystems while displaying telemetry data in a modular fashion. The OCI is composed of multiple panels, each of which display one category of telemetry, such as orientation or artificial intelligence state. The layout of these panels can be reconfigured during runtime to display as much or as little information as the user desires. Individual panels also reconfigure themselves based upon the number and types of devices present on the vehicle. Interaction with the subsystems is accomplished via an integrated Python interpreter.

Simulation

The OCI can also be run concurrently with our Simulator, shown in figure 9. This program simulates *Tortuga IV*'s hardware as well as the competition environment, completely replicating a testing environment. Additionally, the simulator is able to display the view of the on-board cameras, thus enabling integration testing of almost all of the software systems without time-consuming physical dive operations. The simulator, through the use of these mock subsystems, enables concurrent development of the AI and subsystems it is dependent on.

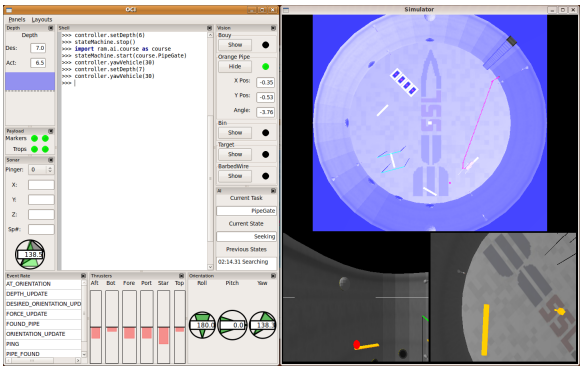

Figure 9: GUI and Simulator

2012 Modifications

This year, the majority of the changes have been in allowing the AI to make better use of new motions in the control section, utilizing the Doppler velocity logger more fully. Many of the software states have been updated with additional failure states, to correct when *Tortuga* has lost sight of the objective or otherwise unable to adapt to changing conditions. These changes should lead to more reliable operation of the software when the conditions make it difficult for *Tortuga* to determine its state.

1 Acknowledgements

Robotics@Maryland would like to thank its sponsors, without whom *Tortuga IV* would have never left the drawing board. Within the University of Maryland we would like to thank the Institute for Systems Research, the A. James Clark School of Engineering, the Space Systems Laboratory, the Student Government

Association, the Maryland Robotics Center, and the Departments of Computer & Electrical Engineering, Aerospace Engineering, and Computer Science. We would also like to thank L-3 Communications, SAIC, MEMSense, Sidus Solutions, TC technologies, SubConn, Advanced Circuits, Teledyne RD Instruments, the NSF ECCS division, Northrop Grumman, USBFireWire, BAE Systems, and Lockheed Martin.

References

[1] M.D. Shuster and S.D. Oh. Three-axis attitude determination from vector observations. *Journal of Guidance, Control, and Dynamics (ISSN 0731-5090)*, 4(1):70–77, 1981.

[2] S. Skogestad and I. Postlethwaite. *Multivariable feedback control: analysis and design*. Wiley, Chichester, 1996.

[3] B. Wie and P.M. Barba. Quaternion feedback for spacecraft large angle maneuvers. *Journal of Guidance, Control, and Dynamics*, 8(3):360–365, 1985.

[4] O. Egeland and J.M. Godhavn. Passivity-based adaptive attitude control of a rigid spacecraft. *IEEE Transactions on Automatic Control*, 39(4):842–846, 1994.