URI RAMboat 2012: A Flexible Architecture for the AUVSI RoboBoat Competition and Beyond

University of Rhode Island
215 South Ferry Road
Narragansett, RI, 02882

Team: Adam Arrighi, Andrew Bird, Tom DeRensis, Robin Freeland, Kevin Hopkins, Rick Kollanda, Hayden Radke, Ian Vaughn
Faculty Advisors: Robert Tyce and Harold Vincent

Abstract: URI's success at the 2011 RoboBoat competition served as an invaluable educational experience. Mechanically, RAMboat 2011easily sustained the rigors of competition, but its numerous computational and electrical shortcomings determined where the bulk of 2012's effort would be required. The goal of RAMboat 2012 was to develop a "smarter" robotic system, one which not only would perform well this year, but also could easily be expanded upon by future ASV teams.

Introduction:

At first glance the 2012 RAMboat platform seems very similar to previous competition vehicles. However, subtle mechanical changes as well as a complete software overhaul offer significant improvements over last year's vehicle. The electrical system is more robust and offers better cooling and the new modular software system is more efficient and intelligent than ever before.

Hardware:

Mechanically and electrically RAMboat 2012 draws heavily from previous years mechanical platforms. A nearly identical power distribution system, sensor payload, and propulsion source are employed this year. For a more in depth hardware investigation see *URI RAMboat 2011Design,* the 2011 journal paper. The following is merely a brief overview with a focus on the improvements to the robot.

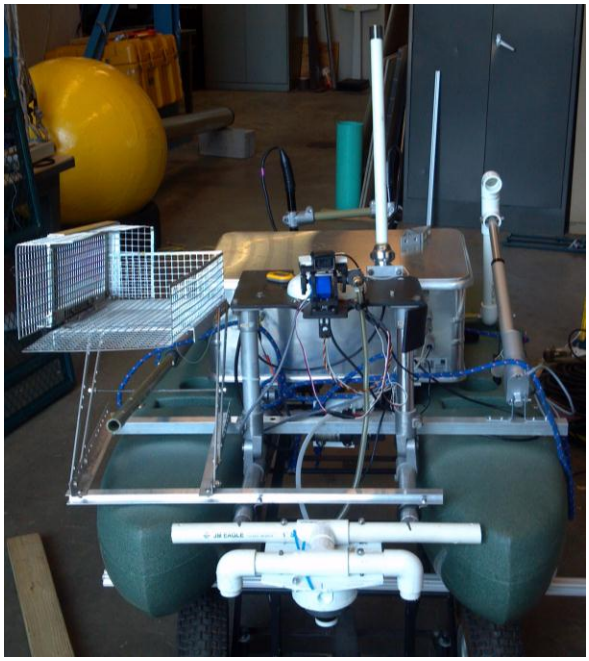


Figure 1:RAMboat 2012

Platform: Hobie Float Cat 60 hulls make up the base and flotation of RAMboat. The hulls are spaced out to the maximum width allowed by competition rules for optimum stability and offer a payload of 250lbs. Two longitudinal aluminum poles offer adjustable mounting for vessel hardware, a system which greatly simplifies trimming the vessel. A watertight, brushed

aluminum case is used to house the electronics. The electrical system is thermally coupled to the case effectively making it a large heat-sink. Additionally the reflective nature of the brushed aluminum reduces solar heating.

Propulsion is provided by 2 Sevylor 18lb trolling motors mounted at the stern and a SeaBotix thruster transversely mounted beneath the vessel at the center of drag. These thrusters are powered by 50A motor-controllers supplied by a bank of four 14.8V 10AHr Lithium Polymer batteries. The trolling motors are angled outwards at the stern of the vehicle to allow for greater maneuverability. By mounting the SeaBotix thruster at the center of lateral drag the vessel can evenly sway from side to side, this added dimension of control allows the vehicle to effectively perform station keeping and further enhances control.

Power Distribution is provided by a legacy power board which is used in both the ASV and AUV competition vessels. It provides fused and regulated power for all of the electrical systems, system voltage measurement, and optical isolation between the motor power and system power. Additionally it allows switching between computer and RC control.

Computers: Ram boat 2012 uses 2 FitPC single-board computers, one for mission control and the other for image processing. A Netburner 5282 is used as an interface between the servos, motor-controllers, and other electrical systems on the vessel and the FitPCs. These computers are connected via Ethernet, a 1 watt wireless bridge links RAMboat's network with the shore.

Hockey Puck Retrieval: A linear actuator deploys a rover (right) which uses a single beam IR sensor to find the hockey puck, then retraces its steps back to the ramp.

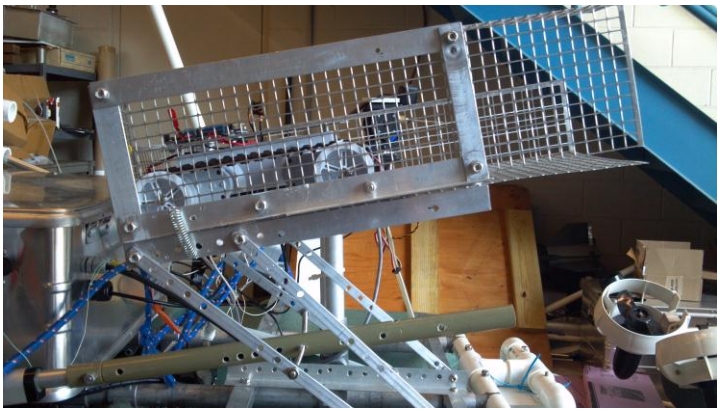


Figure 2: Hockey Puck Retrieval System

Sensors: The Boat uses four primary sensors, the Hokuyo Lidar, Microsoft HD-5000 webcam, phidgets IR sensor, and a Trimble GPS.

Software: Overview

An entirely new software architecture was designed and implemented for the 2012 RamBoat. Key goals for the new architecture included scalability, maintainability, and modularity. In particular, the mission software was split into a number of small, independent processes communicating over set of well-defined Lightweight Communications and Marshalling (LCM) structures. Individual processes are responsible for a single task, such as reading a specific sensor, computing a new set of motor thrusts, or searching images for buoys. The resulting architecture consists of a number of small programs rather than a single, large, extremely complex multithreaded program.

The explicit nature of the interface between processes makes the system extremely modular. Processes that require, for example, compass data, need only receive the correct message type and will operate as long as some other process is outputting that data. On the boat, the basic sensor data is typically supplied by sensor daemons. For testing, however, that data may be replayed from a log file or generated by a simulator. Indeed, such a modular approach leads to a relatively straightforward simulator design; the simulator is simply another process that takes in actuator commands and outputs sensor data.

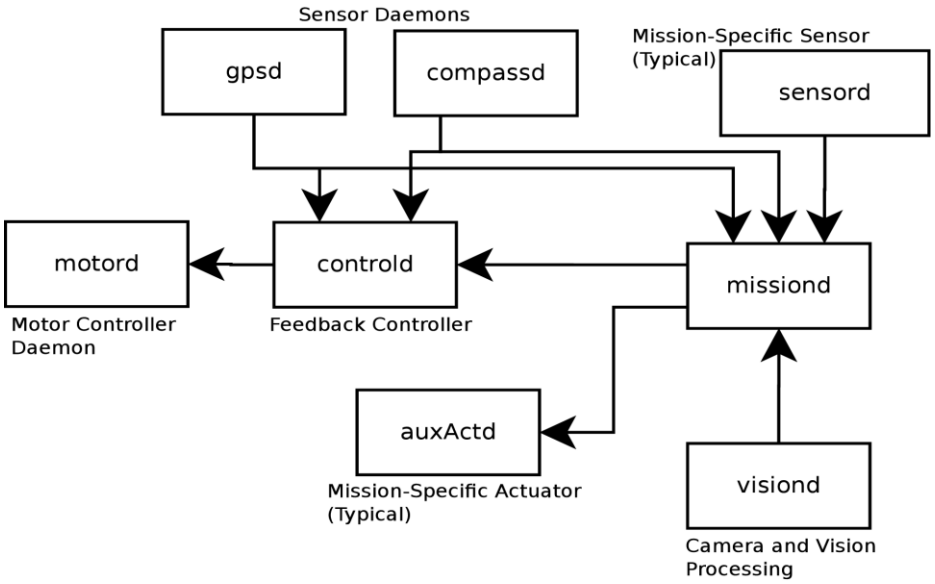


Figure 3: Software Architecture

Communication between blocks is implemented using the open source Lightweight Communications and Marshalling (LCM) communications protocol originally developed for the MIT DARPA Grand Challenge car. LCM provides low-latency communication between processes spread across several hosts using UDP multicast. Finally, LCM provides several helpful utilities including log creation, log playback, and message-sniffing (Huang 2010). Additional support utilities and libraries were provided using libbot (libbot, 2012).

Currently, the implemented processes fall into one of five categories: sensor readers, actuator writers, feedback control, vision/perception processing, and overall mission command. Sensor readers read sensor data and provide it over LCM to all other processes, while actuator writers take command values over LCM and use them to control actuators. Feedback control implements a simple output feedback control loop by computing new motor values from the available sensor data. Vision processes analyze images for objects of interest such as buoys, and output the results to the mission manager. The overall mission daemon processes the available data from all these sources and determines how to direct the controller and auxiliary actuators to best accomplish the mission goals.

Subsection: Vision

Vision processing was performed using OpenCV running on a dedicated FitPC2i running Ubuntu 12.04. The OpenCV library provides a number of useful imaging processing primitives such as color segmentation, feature descriptors, and so on. In addition, a number of useful utility functions for capturing, displaying, and saving images are also provided. Images are read directly from two Microsoft HD-5000 webcams into the processing software. This approach was deemed to be the simplest and most reliable. Images may also be output over LCM for logging and debugging. A Hokuyo planar LIDAR with 3m range provides additional ranging information.

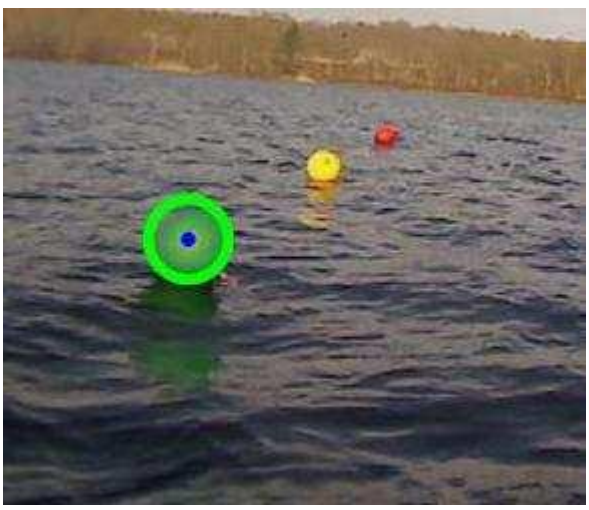
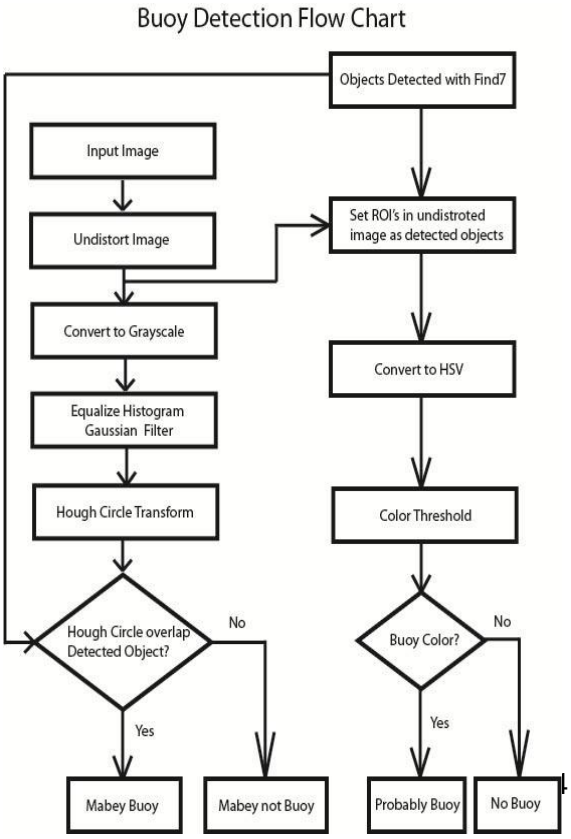


Figure 4: Buoy Detection

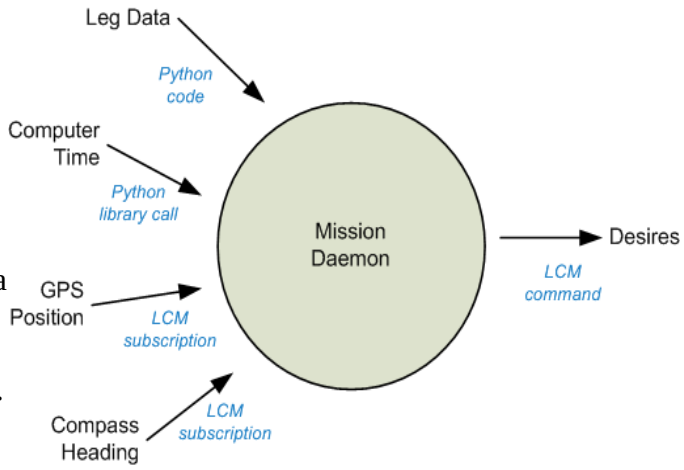
OpenCV's processing primitives are used to implement the overall mission-relevant processing tasks. An example processing flow chart for buoy detection is shown at right. Colors are typically isolated using HSV, which is less sensitive to small differences in lighting. In addition, there has been a great deal of recent work to improve robustness by exploiting the known shape of many of the targets provided by the competition.



Individual detectors may be activated or deactivated by the mission software to avoid unnecessary processing and improve framerate. A brief descriptor of each detected object is sent to the mission computer over LCM.

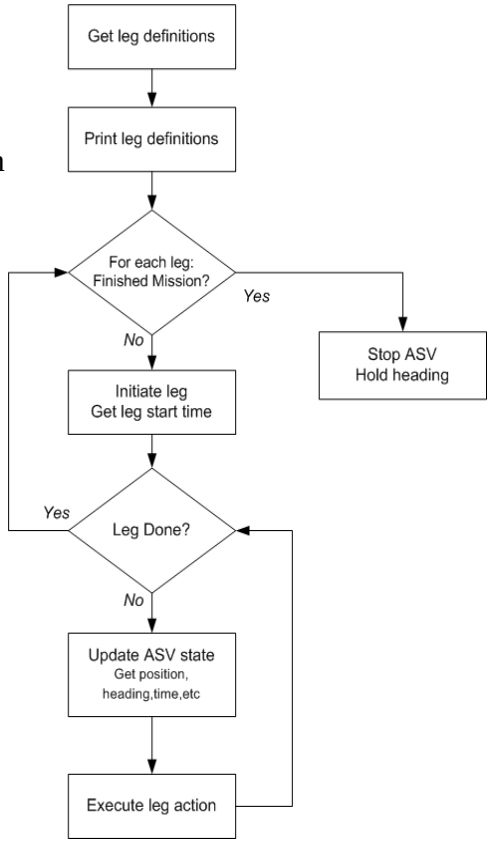
Subsection: Mission

The mission daemon is responsible for exercising high-level control over the vehicle's subsystems in order to accomplish specific tasks. The overall mission is represented as a list of "legs" to accomplish in order. Individual legs may be quite simple, such as holding a speed and bearing for a fixed period of time, or quite complex, such as the legs that use vision and GPS data to traverse the buoy course.



Each leg is represented as a python subclass. The mission then deals with each leg as an abstract entity. Common functionality, such as terminating the leg at a specific time, are included in the mission daemon to ensure that legs terminate correctly even should they crash. Furthermore, the mission daemon does not require any knowledge of the details of a leg. Additional leg types may therefore be added by simply specifying a new class in the leg table, making the whole system extremely flexible.

Legs are written as a python script that is passed to the mission daemon and compiled at runtime. This method was selected to make the leg files as flexible as possible. Leg tables can be as simple or sophisticated as required; any script that outputs a correctly-named list of objects that implement the required functions will work as a leg table.



We would like to thank the following sponsors:
URI College of Engineering
Raytheon
URI Department of Ocean Engineering
Battelle
Pacific Crest

Reference:

A.S. Huang and E. Olson and D.C. Moore. *LCM: Lightweight Communications and Marshalling.* International Conference on Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ. Pages 4057-5062. 10.1109/IROS.2010.5649358. 2010.

libbot: Set of libraries, Tools, and Algorithms for Robotics Research. http://code.google.com/p/libbot/. Retrieved 28 May 2012.