Autonomous Aquatic Agents

A. Calce, P. Mojiri Forooshani, A. Speers, K. Watters, T. Young and M. Jenkin

Computer Science and Engineering, York University, Toronto, Canada

Keywords: Autonomous Aquatic Agents.

Abstract: Constructing a collection of autonomous agents requires the development of appropriate experimental hardware platforms. Here we describe the process of re-purposing inexpensive radio-controlled (RC) electric motorboat as autonomous surface craft. Standard electronics components are used to interface with the RC boat electronics, and the vessels are augmented with GPS, vision, and a tilt-compensated compass to provide the necessary onboard sensing capabilities to enable point-to-point and target-based control of the vehicle. A ROS-based control and sensing infrastructure is used to operate the vehicles on-board while 802.11n communication provides communication off-board. Vessels have been operated successfully in both the pool and ocean environment.

1 INTRODUCTION

Unmanned surface vessels (USVs) have undergone a rapid growth over the last five years and we are now beginning to see devices in a range of different applications. Autonomous Surface Craft (ASC), also referred to as Unmanned Surface Vessels (USVs), have been developed for civil, military and research purposes. Proposed applications for autonomous surface vessels for civil applications include automated bathymetry, coastal surveillance, water sampling, and environmental monitoring (Caccia, 2006). Of particular importance is the ability of an ASC to operate as a mobile relay point between radio frequency transmissions in air and acoustic transmission undersea. In this role they can be deployed as part of a cooperative heterogeneous team including AUVs (Autonomous Underwater Vessels). This fact is a key point in many military applications (Manley, 2008). Other Military applications include harbour security, mine sweeping and various military missions(Manley, 2008; Caccia, 2006; Bertram, 2008).

Although a number of autonomous watercraft projects exist, much of the technical details associated with these projects is not well recorded and individual projects must often start again from scratch. Here we describe efforts to utilize standard tools from the RC and hobby electronics community to construct a fleet of autonomous surface vessels that expose ROS, a standard robot middleware (Quigley et al., 2009) to external users. The basic structures used in this project can be easily adapted to other applications and

Figure 1: The "minnow". The autonomous service vessel is constructed from a modified RC motorboat, and has been augmented with onboard electronics and sensors. An 802.11g network connection provides offboard communication.

the basic electronic components used are "standard" devices that are easily sourced.

2 VESSEL DESIGN

A basic design question in the development of any autonomous system is the size of the device. Size, measured in volume or mass, provides a range of constraints related to power, locomotive strategies, onboard sensor capabilities, onboard computation, communication opportunities and the like. This constraint is especially true in the development of an au-

tonomous surface vessel based on an RC platform. RC vessels are typically small, designed to be easily portable and with relatively short operational periods (typically under an hour, and often significantly less). On the plus side, RC vessels typically avoid regulatory issues associated with operating a "human sized" vessel autonomously. One critical question in terms of the re-purposing of RC vessels is the nature of the vessel power plant. RC watercraft are developed with electrical systems for onboard control and communication, and are available with electric, gasoline and nitrox drive systems. Each of these drive alternatives come with their own capabilities and constraints. Electrical powered systems are emission free and thus are easily deployed in indoor swimming pools, but they lack the extended range and maximum power output associated with gasoline and nitrox power plants. (A properly tuned nitrox boat can exceed 80 kilometers an hour in terms of speed.) Fortunately the basic control mechanisms remain unchanged over the various power plants, but the choice of plant type has a significant impact on maximum speed, available test environments and operational period.

For this project the "Blackjack 26" platform was chosen. The Blackjack 26 is a "ready to run" 26" fiberglass catamaran RC vessel powered by a single propeller and equipped with a single rudder (see Figure 1). The vessel is powered by a 1500 RPM brushless DC motor and utilizes a standard RC servo motor to provide positional control over the rudder. The boat comes equipped with a water-cooled 45A programmable Electronic Speed Controller (ESC) that provides control over the drive motor, power to the servomotor, and exposes itself to control circuitry as a standard servomotor. As shipped the vessel contains a pair of 7.2V sub-C battery packs that provide power for both the motors and the associated electronics. Normally the vessel is controlled by a remote radio controller, but this device was removed prior to re-purposing the vehicle.

The robot operating system (ROS) has emerged as the de-facto standard in terms of middleware for research robots. Rather than attempting to develop a custom software environment for the vehicle as much as possible of the software developed for the vehicle leverages existing ROS infrastructure. This choice enabled the project to leverage a significant open source code base but lead to a number of design decisions in terms of hardware and software infrastructure, as detailed later in this paper.

2.1 Electromechanical Issues

A surface vessel like the Blackjack 26 is essentially a two degree-of-freedom (DOF) device. There is a rudder and throttle, both of which are controlled by 5V signal lines. The necessary control signal to these two inputs is provided by a radio receiver in the stock RC boat. A key step in re-purposing the vessel as an ASC is providing computer control of these two degrees of freedom. We chose to use an Arduino microcontroller to provide this interface. The Arduino (μC) was chosen for a number of reasons related to ease of use, ability to provide the necessary control interface to the underlying hardware, and also the existence of an Arduino ROS package that allows ROS nodes to be executed on the Arduino directly and to be made visible to standard ROS nodes operating elsewhere.

The Arduino microcontroller provides an interface to a tilt-compensated compass and provides reference voltages and PWM control signals to the rudder servo and the ESC controlling the throttle. The Arduino also provides power to the rudder servo. Normal power from the drive system provides power to the ESC. The Arduino is connected via USB to an onboard BeagleBoard SBC which provides higher-level control as well as providing power to the Arduino.

Each of the motor controllers onboard the robot requires a reference ground and a 5V control line. These are easily mapped to the Arduino's digital out and reference lines.

Although the use of the Rosserial package greatly simplifies integrating low-level control of an actuator into a collection of ROS nodes, it is not without its limitations. The existing version of the library limits the number of publishers and subscribers on the Arduino. Nor are all primitive types supported in messages that can be passed. Furthermore, the connection between the Arduino and its host computer is via a USB connection, and this places limits on how much, and how quickly, data can be passed from code running on the Arduino and that running on the host device. (For example, sending long string messages is not to be encouraged given the limited 280 byte buffer size available.)

2.2 Computation

The limited volume and mass characteristics of the Blackjack 26 places extreme constraints in terms of potential onboard computation. Furthermore, the decision to build a ROS-compatable device requires that the onboard computer run some version of Linux. Although a number of small form factor devices meet all of these requirements, the decision was made

to proceed with a BeagleBoard MX platform. The benefits of using a BeagleBoard have been demonstrated against several other platforms small form factor boards (see (Watters et al., 2011)). The Beagle-Board runs a quasi-standard version of Linux, provides a reasonable number of input and output ports – including onboard video – and is reasonably inexpensive. Furthermore, it supports a number of different power inputs, including being able to be powered by a USB power source.

The BeagleBoard was configured to run Ubuntu Linux and Robot Operating System (ROS) (Quigley et al., 2009). When on, the BeagleBoard serves as a link between the base station computer (when instructions are sent to, or information is requested from the robot) and the low level systems of the robot through an 802.11g connection. The control of the robot's single servo, electronic speed controller, and 3D HMC6343 magnetometer compass is delegated to the Arduino which is connected to the BeagleBoard via a USB connection. Connected directly to the BeagleBoard are on-board camera and GPS modules. Any commands to alter the motion of the robot are either generated on the Beagleboard or received by the BeagleBoard and directed to the Arduino Uno. The camera selected for use on the robot was a Logitech Quick Cam Pro for notebooks. This camera provides images up to 2.0Mpixel at 5fps and is UVC compliant (allowing for simple interfacing within the Ubuntu environment). The GPS selected was the BU-353 USB GPS Navigation Receiver that connects to the BeagleBoard through an RS232 connection which propagates the GPS data in NMEA format to the Beagle-Board.

The process of configuring a BeagleBoard to run ROS was not without its difficulties. The operating system most fully supported on the BeagleBoard family of single board computers is Angstrom, and initial development relied on this OS. ROS is not officially supported on the Angstrom OS, and efforts to compile ROS for the Angstrom-BeagleBoard combination did not lead to a robust ROS environment on the platform.

In order to provide a supported OS for ROS, a stripped-down version of Ubuntu ARM was deployed on the BeagleBoard. This provided a rather straight-forward installation of the majority of the ROS environment and its associated support libraries. ROS Electric Base and Ubuntu 11.04 (Natty Narwhal) were used as these were the most "cutting edge" compatible versions for the BeagelBoard platform at the time.

2.3 Sensing

Given the experimental nature of the vessel being con-

structed a wide range of different sensors have been built into the vehicle including GPS, video and a compass.

GPS. A ROS GPS node interfaces with the *BU-353 USB GPS Navigation Receiver*, parsing the data stream (in NMEA 0813 V2.2 format). Currently the node parses only the GPS system fix data (*$GPGGA*). These strings contain information on the latitude, longitude and altitude as well as associated data on fix quality and time of fix. The parsed latitude / longitude data is converted into a $+/-$ degree with positive values going North (Latitdue) and East (Longitude). The latitude and longitude are encapsulated in a custom gps message and are available to the rest of the system. If the GPS fix quality is zero an error latitude / longitude value are returned.

Front Facing Video Cameras. A USB Quick-Cam Logitech webcam with 640x480 resolution is mounted onboard the vessel. This video source can be used both by a remote user to teleoperate the vehicle, as well as by ROS software to control the robot using visual landmarks. Given the Linux nature of the onboard computation and the use of ROS, the process of capturing video onboard the robot is accomplished using the standard Video4Linux library and images are transferred as a single ROS sensor_msgs::Image message. Other nodes, including nodes off-board the vehicle can subscribe to this published image stream.

Tilt-compensated Compass. Although considerably more expensive than their non-tilt-compensated counterpart, a tilt-compensated compass is essential on a surface vessel where wave action will ensure that the vehicle is rarely horizontal. Few tilt-compensated compasses exist with a USB interface, and thus it was necessary to decode the signal from the device in order to capture the vessel pitch, roll and yaw. Given the existence of the Arduino onboard the vessel to control the actuators, the Arduino was purposed to monitor the compass as well. The minnow publishes a `vesselStatus` message that contains the current commanded position of the rudder and throttle along with its pitch, roll and yaw. This custom message is defined on the Arduino and its definition is propagated to the rest of the ROS network through the Rosserial node definition.

In addition to these sensors, the onboard computer is equipped with an 802.11g wireless card to enable communication with an off-board controller.

2.4 Robot Control

Controlling software for the minnow robot is provided in ROS. In order to ensure the widest possible interoperability of the ROS nodes across platforms, wherever possible the ROS nodes have been implemented in Python. This choice reduced development time substantively on the minnow robots as the time required to compile ROS code on the BeagleBoard is not insignificant. Note that the other alternative – cross compiling on a host machine – requires the presence of an appropriate host machine to be deployed along with the robot platforms.

2.5 Vehicle Testing

The minnow USV's performed well in several pool trials as well as open water trials off the coast of Holetown, Barbados in January 2012. During these trials several things became apparent. First, the hull was able to take on the extra payload and run quite efficiently. At 20% throttle the boat was able to navigate fairly choppy waters and outpace human swimmers. A straightforward PID controller was constructed in ROS to provide bearing following. One issue that we expected to take considerable time during the field trials was tuning this PID controller for the open water conditions. Surprisingly, the tuning of the PID controller is less critical in open water as opposed to a pool or pond setting as the rough surface acts to damp the system making changes in rudder position take time to accumulate into the overall vehicle trajectory. In all, the vehicle was successfully tele-operated visually and commanded to follow a compass heading during several trials both controlling the rudder position directly as well as controlling the compass heading through the PID "Helmsman" controller.

3 DISCUSSION AND FUTURE WORK

During the trials the need for a heart-beat (still-alive) signal was made evident. The initial prototype would maintain it's last command until completion, received another command or until it ran out of power. This command structure would become problematic if the boat for any reason lost contact with the base station due to driving too far away from "home" or due to a wave blocking the signal. This latter problem was experienced during heavy swells where the minnow "lost sight" of the 802.11 base station. Modifying the ROS command structure to require a heart-beat in the command structure allows for the boat to return to

some default action (i.e. set the throttle to zero) when communication is lost.

On a more forward-looking theme the use of ROS as the core software architecture running the boat makes this project very expandable. We are looking forward to both leveraging the work of others on this platform (through the use of ROS nodes) as well as using multiple vehicles to perform a common task. In particular, ongoing work is exploring coordinated operation between a larger surface vehicle Eddy (a Clearpath King Fisher) and a fleet of minnow-class robots. Eddy is equipped with a differential GPS system as well as a long range radio communications channel. This would enable the rest of the fleet (comprising of the cheaper minnows) to relay communications through the larger surface vehicle and perform longer range missions as well as benefit from more accurate positioning through the larger craft.

It is also anticipated that we will be performing joint actions with the AUV Kroy (Dudek et al., 2007). In these experiments we anticipate having one (or more) surface vehicles performing functions from the surface (perhaps finding places of interest for environmental modelling or surveying) with the large Kingfisher tracking visual targets on Kroy and commanding Kroy's actions.

REFERENCES

Bertram, V. (2008). Unmanned surface vehicles - a survey. Technical report, Skibsteknisk Selskab, Copenhagen, Denmark.

Caccia, M. (2006). Autonomous surface craft: prototypes and basic research issues. In *Control and Automation*, pages 1–6.

Dudek, G., Giguere, P., Prahacs, C., Saunderson, S., Sattar, J., Torres-Mendez, L.-A., Jenkin, M., German, A., Hogue, A., Ripsman, A., Zacher, J., Milios, E., Liu, H., and Zhang, P. (2007). AQUA: An amphibious autonomous robot. *IEEE Comptuer*, pages 48–53.

Manley, J. E. (2008). Unmanned surface vehicles, 15 years of development. In *Oceans*, pages 1–4.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source robot operating system. In *Proc. of the Open-Source Software Workshop at the International Conference on Robotics and Automation (ICRA)*.

Watters, K., Minwalla, C., Liscombe, M., Lio, H., Thomas, P., Hornsey, R. I., Ellis, K., and Jennings, S. (2011). Characterization of an optical collision avoidance sensor. In *IEEE CCECE*.