Programmable Autonomous River Droid: Tracking and Data Fusion

Aaron Springut (IEEE 90691505) and Sam Courtney

Lafayette College

IEEE Branch Counselor: Professor Yih-Choung Yu
Programmable Autonomous River Droid: Tracking and Data Fusion
# Table of Contents

Abstract ........................................................................................................................................... 0

Introduction ..................................................................................................................................... 1

Drone Tracking and Data Fusion System ....................................................................................... 3

System Overview ........................................................................................................................ 3

Shore Sensors ................................................................................................................................ 6

Azimuth-Elevation Positioner ................................................................................................. 6

Infrared .................................................................................................................................... 7

Ultrasonic................................................................................................................................. 8

Boat Sensors ................................................................................................................................ 9

Accelerometer ........................................................................................................................ 10

Global Positioning System .................................................................................................... 10

Gyrosopes ............................................................................................................................ 10

Magnetometer ........................................................................................................................ 11

Filtering ..................................................................................................................................... 11

System Control and Data Acquisition (SCADA) Software ...................................................... 12

Calibration..................................................................................................................................... 13

Conclusion .................................................................................................................................... 15

Appendices .................................................................................................................................... 16

Table I – Definition of Variables ........................................................................................... 16

Section I – Error Analysis ...................................................................................................... 16

References................................................................................................................................. 20
Abstract

This paper describes the first part of a multi-year capstone design project undertaken by the Electrical and Computer Engineering class of 2012. The goal of the project is the development of a small autonomous boat that will be capable of precision navigation and data collection in a creek or river even in rapidly moving water. This year is focused on developing sensor systems that track the boat hardware and measure its position with an uncertainty approaching 10 cm root mean squared (RMS) at a range up to 25 meters. A software framework for system control and data acquisition is being developed with feedback for end user hydrologists.
Introduction

As urbanization and population around the globe increase, maintaining the quality and quantity of water supplies is becoming an important issue (1). There are a multitude of factors which influence water quality, including agricultural runoff, urban runoff, and sewage from manufacturing and industry. Because this is such a complicated and multidimensional problem, measurements to determine the quality of water are problematic. This complexity is only compounded in local river systems where pollution enters at a single location and is then carried down-stream. Additionally, distributed sources of contamination, such as runoff or absorption of airborne pollutants, can enter the river over a wide area. Determining the source of water pollution from a single measurement can be a difficult if not impossible job.

Ideally, multiple measurements could be taken along a stream and compared to obtain a complete picture of the water quality. Hydrologists must visit many sites on the river and manually take water quality readings, or set up a station at a single site to take continuous readings (2). An autonomous system would be excellent for this repetitive task. For example, taking flow measurements is one of the hydrologist’s most tedious tasks. The researcher must make flow velocity and depth measurements at multiple locations across the width of the river. Presently, this involves wading into the river if it is shallow enough, or stringing a probe from a bridge or other structure over the river. By traversing the river automatically while taking measurements, much of the work is eliminated.

In this paper we present such an autonomous boat system. Although several autonomous boat systems already exist, all of these systems are guided by global positioning systems (GPS), which limits the accuracy of the position measurements (3). For making hydrology
measurements such as stream flow, significantly more accurate positioning is necessary. Our system utilizes a fusion of both GPS and multiple other sensors such as ultrasonic rangers, optical trackers, compasses, accelerometers and gyroscopes to sense position more accurately (4). To achieve a higher accuracy, all of incoming position data is passed through a particle filter. Particle filters have been used successfully in tracking models in a range of robotics applications over the past several years (5) (6). This filter allows us to combine several different measurements along with a system model of boat position into a single, more accurate measurement of the boat’s states.

These improvements allow our system to determine the boat location to a significantly higher degree of accuracy than current autonomous boats. This makes it ideal for use by hydrologists studying water quality in river environments. The result would be a significant reduction of the quantity of work and money required to survey a single sight on the river, allowing hydrologist to gather a larger number of river samples and get a more complete picture of river water quality.
Drone Tracking and Data Fusion System

System Overview

The overall system (Fig. 1) consists of a pair of tripod mounted shore-stationed sensors, a sensor interface box and user interface software, boat based sensors and, boat-beacons. Analysis predicts the unfiltered value of the shore sensors will be less than 20 cm RMS. With a particle filter it is anticipated that the 10 cm RMS accuracy requirement can be met at a range of 25 m.
The software modules act as the primary user interface for controlling the boat in automatic mode (with scripts or simple commands). The two tripod sensors are of the exact same design, containing a motor system and the shore-based portions of the infrared and ultrasonic sensor systems. The shore system (Fig. 2) also contains one end of the radio frequency (RF) datalink.

The boat hardware (Fig. 3) consists of two beacons used by the shore sensors and an ArduPilot Mega (7). An XBee link for telemetry transmission is in use as a shore-to-boat
communication device. The team has been adapting the hardware and firmware libraries provided to enable high-resolution positioning of the drone on water, and to communicate in duplex with the shore station. Our system tracks a variety of boat states to ensure the gathered data is of maximum utility (Fig. 5). These states include: absolute x and y position, absolute heading, absolute rotational velocity, and relative forward and sideways speeds. To allow independent control all of these variables independently of one another, the design is based off of a boat with four bi-directional jet-pump motors (Fig 4). This would allow the boat to be stationary even in moving water, while simultaneously maintaining its heading so sensors that rely on being oriented correctly to operate.

Fig. 4 Boat motor configuration

Fig. 5 Boat orientation diagram
The system is designed such that it is easy to deploy. The user must deploy the two tripods, mount the two sensors and attach the interface box. Then the system automatically calibrates and determines its local coordinate system (Fig. 1).

**Shore Sensors**

**Azimuth-Elevation Positioner**

To track the position of the boat and send sensor data back to the Shore Station, we use an Elevation over Azimuth positioner (AZ/EL). The AZ/EL positioner provides a mounting surface that rotates about two axes. Two interconnected servomotors achieve the desired motion. The servomotors interface with an ATMega328 microcontroller chip. Our team has constructed a custom ultrasonic ranging device and infrared tracker mounted on the head of the AZ/EL positioner (Fig. 6, Fig. 7). In addition a second printed circuit board (PCB) is mounted on the back of the positioner that contains an interface for power and communications, additional amplifiers, and signal processing hardware.
The ATMega328 microcontroller executes a control loop to keep the positioner pointed at the source of the infrared light on the boat. This also ensures that the co-located ultrasonic rangers are always pointed at the boat. We use the available Inter-Integrated Circuit (I2C) hardware on the microcontroller to transmit data and commands to and from the shore station. Using this interface, the positioner can report where it is pointing and how far the boat is. The AZ/EL positioner is mounted to the top of a tripod and upside down so the servo motors can make the best use of their available movement range. A cable from the shore station brings power and I2C communications to the sensor.

![Fig. 7 Shore sensor and pre-amplifier board](image)

**Infrared**

The boat-mounted infrared beacon consists of an array of 4 high irradiance, wide-angle, infrared light emitting diodes mounted on top of a mast with a switch modulating driver circuit. We use infrared light with a wavelength of 850nm instead of visible light. There is significantly less ambient infrared light, allowing us to differentiate our signal from other ambient light (8). Additionally, we modulate this beacon at 20KHz, to further isolate signal from ambient light.
The shore mounted optical sensor assembly consists of an infrared-pass optical filter with a lens that focuses incoming light to a quadrant photodiode mounted on the AZ/EL tracker (Fig. 7). To determine where the beacon is, we use a lens to focus the light from the beacon to a small circular area over the photodiode. When this lens is pointed directly at the beacon, the intensity on all 4 channels is equal. If the boat beacon is skew with the lens the incoming light will land disproportionately on one of the sensors. Using this information we can calculate the magnitude of the skew in both azimuth and elevation directions.

The quadrant photodiode channel is connected to an RLC band pass filter which is then followed by multiple amplification stages. The multiple gain stages allow us to turn on and off gain in order to prevent saturation. The amplification stages are connected to a peak detector circuit transforming the alternating current (AC) output of the photodiode into an analog direct current (DC) voltage. This voltage is converted into a digital measurement, which is then fed into an onboard microcontroller containing the control loop.

**Ultrasonic**

In addition to the Infrared tracking system, the project includes an ultrasonic pinging system to determine the range of the boat from a shore tripod. This range measurement allows the system to calculate an x and y, giving two independent measurements for fusion into a more accurate position. Our system sends an ultrasonic ping to the boat, which then responds with another ping. The time delay from the first ping to the response is combined with a pre-calibrated speed of sound to calculate the boat’s distance.

The ultrasonic shore station circuitry contains two different ultrasonic transducers, one for receiving and one for transmitting, as well as driver and receiver circuitry (Fig. 7). The driver circuit consists of a 4 MHz oscillator which is divided down to a 40 kHz signal and used to drive
a dual, half H-bridge circuit running off of 18 volts which produces a 36 V peak to peak, 40 kHz signal across the transmitter.

The receiver is sensitive only to signals in the range of 38 to 42 kHz. These signals are then passed through two separate amplification stages (Fig. 8). The amplified signals are then sent through a comparator which triggers off of a specific threshold. This signal is then processed by a microcontroller to detect incoming pulses.

**Boat Sensors**

![Fig. 9 Boat processor, sensors, and RF link](image)
Accelerometer

A three-axis accelerometer is integrated on the ArduPilot Inertial Measurement Unit (IMU) shield, with communication handled by the open source libraries provided for the ArduPilot (Fig. 9). The accelerometer reports acceleration in the range of +/- 3g as meters per second squared. This data is both transmitted in the raw to the shore station for logging and used by the particle filter to help track the change from the boat’s previous position.

Global Positioning System

A GPS is sited on a breakout board developed specifically to work with the Ardupilot setup (Fig. 9). This board is connected directly to the front of the ArduPilot board, where it communicates using a dedicated Universal Asynchronous Receiver/Transmitter (UART). Communication with the GPS is handled by another open source library that can be found in the ArduPilot source code package. The GPS itself is, unfortunately, not accurate enough to be used alone. Specifically, it is capable of resolving to within a 3 meter radius of the unit 50% of the time - this means the system can establish a global positioning fix that is sufficiently accurate to be used for locating the operating site. We can also use the course speed and heading information provided by the GPS to help update positions, though it must be given a light weight to avoid the potential for larger errors.

Gyrosopes

There are two Gyroscope packages sited on the ArduPilot IMU board: one XY-axis gyroscope and one Z-axis gyroscope (Fig. 9). Their placement enables the tracking of the three different rotational axes on the main board, which allows us to feed the rotation in radians per second to the position filter. The gyroscope data is also used to calculate the system’s pitch and...
roll, which is used by the magnetometer libraries while calculating a heading when compensating for tilt.

**Magnetometer**

The final sensor provided in the ArduPilot package is an optional triple-axis magnetometer (Fig. 9). This is located on a small breakout board utilizing an I²C connection to the ArduPilot via the IMU. This magnetometer can be used as a compass for determining the heading of the boat, allowing for simplified navigation that doesn’t rely on guess-and-check movement. Unfortunately the triple-axis nature requires more involved firmware capable of accounting for roll and pitch to ensure only the yaw angle is tracked. The gyroscope data is extremely valuable for this purpose, allowing the library to calculate the corrected heading.

**Filtering**

Our system uses a particle filter to combine measurements from these sensors into a coherent and accurate picture of the boat’s state. Particle filters are part of a class of artificial intelligence algorithms called sequential Monte Carlo methods (6). In the simplest possible terms a particle filter keeps track of a range of possible system states called “particles” and discards those that are the least probable according to external measurements. Particle filters have been used in a wide variety of autonomous systems in recent years due to their advantages over more traditional methods such as Kalman filters (8). Most importantly, particle filters do not require that the system model be linear. Additionally, they are computationally simple allowing even a low power processor to execute one. Finally, as the number of system states being tracked increases the probability that the filtered system state is the true system state approaches 1.
Our model for the boat is highly nonlinear, as can be seen by an examination of the transfer function (Fig. 10). This non-linearity requires a form of filtering capable of working even with non-linear models. This ruled out a standard Kalman filter (although not an extended Kalman filter). Ultimately, we decided to use a particle filter because of its simplicity and ability to work with nonlinear models.

\[
\begin{bmatrix}
\text{xVel} \\
\text{forAcc} \\
\text{yVel} \\
\text{sideAcc} \\
\omega
\end{bmatrix}
\begin{bmatrix}
0 & \cos(\theta) & \sin(\theta) & 0 & 0 \\
-\frac{f_g}{m} & 0 & 0 & 0 & 0 \\
\frac{m}{\sin(\theta)} & 0 & \cos(\theta) & 0 & 0 \\
0 & \frac{m}{\sin(\theta)} & 0 & -\frac{f_g}{m} & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\text{xPos} \\
\text{forVel} \\
\text{yPos} \\
\text{sideVel} \\
\theta
\end{bmatrix}
+ 
\begin{bmatrix}
0 & 0 & 0 & 0 & -1 \\
1 & 1 & 0 & 0 & 0 \\
\frac{m}{m} & m & 0 & 0 & 0 \\
0 & 0 & \frac{1}{m} & \frac{1}{m} & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
F_1 \\
F_2 \\
F_3 \\
F_4 \\
R_{vel}
\end{bmatrix}
\]

Fig. 10 System state-space model (see Table I for definitions)

**System Control and Data Acquisition (SCADA) Software**

The laptop software is the single point of comprehensive control for our entire system. Every time the software is started it enters an operational calibration state at which point sensors are calibrated for current operating conditions. The boat’s initial position will then be found by scanning the sensors over the operational swath. From this point on the software will use readings from the trackers to determine the position of the boat.

The boat is continuously tracked and sensor information is logged. The graphical user interface (GUI) has two areas for displaying sensor measurements – one section for sensors relevant to the boat’s status, such as location, heading, speed and internal temperature; the other for any relevant payload hydrology sensors such as depth, pH, water temperature, and turbidity.
All of these sensors have their information logged in separate CSV files to allow the ability to graph in external graphing software with no additional overhead.

Fig. 11 Main GUI

The software supports three autonomous modes of navigation as well as manual mode. Freeze mode which holds the boat’s current position in the water. Ferry mode where the user can specify a pair of x and y coordinates as a destination for the boat. Scripting mode allows the specification of a combination of ferry and freeze commands. When in autonomous navigation mode, an emergency stop sends the boat into a freeze state to prevent it from being damaged. It also has an emergency ‘return to origin’ mode. Manual control mode overrides all other modes of operation via a separate radio frequency remote control. This is meant as a failsafe as well as a more direct means of control for the boat.

Calibration
A fundamental aspect of the system is the ability to measure the position of the boat using our sensors. To ensure measurement accuracy, the various subsystems must be calibrated. In this process we take raw measurements from our sensors and compare them to standard systems of measurement. We then decide what modifications to make our measurements so that they match the standard. There are three main aspects to our system calibration: primary calibration, operational calibration, and verification calibration.

Primary calibration takes place before the system is shipped to the end-user. In this stage, the instrument is modified so the measurements are internally consistent and as close as possible to the standard. For our system the AZ/EL is calibrated to determine the angle zeros as well as the angle to pulse width modulation time ratio.

Operational calibration is done directly prior to use. This stage is used to eliminate errors caused by outside factors that will affect the system’s measurements. The range finding technique requires knowing the speed of sound, but the speed of sound can vary greatly depending on temperature and other factors. Therefore, our system must determine the speed of sound empirically before use. It calculates this value by sending pings back and forth over a known distance and calculating the time of flight.

Verification calibration is used during the operation of the system in order to ensure the measurements received from the system are still valid. This can also be called “sanity checking” and usually entails checking to see if the measurements are within possible limits. A simple example is checking to see if a time measurement is negative. Since measured time cannot be negative, the system can clearly see something has gone wrong and can warn the user of erroneous data. The main check for our system is ensuring the baseline between the sensors is
always less than the sum of the distances from the sensors to the boat. If the sum is greater than the baseline then the system is reporting a configuration that is not geometrically possible.

**Conclusion**

It is our expectation that this portion of the multi-year project will be capable of successfully tracking the boat with a high degree of precision. This portion of the project will be completed by the end of the semester and will be ready for integration with the efforts of future capstone design teams. At the end of the three year project a fully functional autonomous boat will be capable of measuring using end user provided sensors in rapidly moving water. The relative simplicity of the system will allow hydrologists to replace more tedious manual testing with autonomous tests.
Appendices

Table I – Definition of Variables

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>xPos</td>
<td>x position</td>
</tr>
<tr>
<td>xVel</td>
<td>x velocity</td>
</tr>
<tr>
<td>forVel</td>
<td>forward velocity</td>
</tr>
<tr>
<td>forAcc</td>
<td>forward acceleration</td>
</tr>
<tr>
<td>yPos</td>
<td>y position</td>
</tr>
<tr>
<td>yVel</td>
<td>y velocity</td>
</tr>
<tr>
<td>sideAcc</td>
<td>sideways acceleration</td>
</tr>
<tr>
<td>sideVel</td>
<td>sideways velocity</td>
</tr>
<tr>
<td>m</td>
<td>boat mass</td>
</tr>
<tr>
<td>θ</td>
<td>angle</td>
</tr>
<tr>
<td>ω</td>
<td>angular velocity</td>
</tr>
<tr>
<td>α</td>
<td>angular acceleration</td>
</tr>
<tr>
<td>F1/2/3/4</td>
<td>motor forces</td>
</tr>
<tr>
<td>RVel</td>
<td>river flow velocity</td>
</tr>
</tbody>
</table>

Section I – Error Analysis

To convert azimuth, elevation and range to Cartesian coordinates we can use a simple spherical to rectangular transformation as defined below:

\[
x[r, \phi, \theta] := r \cdot \cos[\theta] \cdot \sin[\phi]
\]

\[
y[r, \phi, \theta] := r \cdot \sin[\theta] \cdot \sin[\phi]
\]

\[
z[r, \phi, \theta] := r \cdot \cos[\phi]
\]

To determine the standard deviation of these Cartesian coordinates from the standard deviation of our measurements we must know the partial derivatives of these functions with respect to range(r, rho), elevation(φ, phi) and azimuth(θ, theta):

\[
dxdr[r, \phi, \theta]:=\cos[\theta] \cdot \sin[\phi]
\]

\[
dxdp[r, \phi, \theta]:=r \cdot \sin[\theta] \cdot \cos[\theta]
\]

\[
dxdt[r, \phi, \theta]:=-r \cdot \sin[\theta] \cdot \sin[\phi]
\]

\[
dydr[r, \phi, \theta]:=\sin[\theta] \cdot \sin[\phi]
\]

\[
dydp[r, \phi, \theta]:=r \cdot \sin[\theta] \cdot \cos[\phi]
\]

\[
dydt[r, \phi, \theta]:=r \cdot \cos[\theta] \cdot \sin[\phi]
\]
When we multiply the partial derivative by the standard deviation in the variable that it was with respect to we find the standard deviation in the Cartesian system as a function of rho, phi and theta. We can then take the root sum square of each of the three sources of error to determine the total standard deviation of error in a single Cartesian direction. This is then plotted over our operational swath (semi-circle) (Fig. 12).

Fig. 12

From this we can determine the total error by taking the root sum square of both x and y Cartesian errors (Fig. 13).

Fig. 13
We can perform this same procedure with two sensors with a 10m baseline by taking the inverse sum of the inverses of the standard deviations (Fig 14).

![Total Standard Deviation of Error Over Operational Swath(m)](image1)

We can also calculate the offset of two sensors with and offset in rho, phi or theta by comparing the true x position to the x-position that is calculated with the offset. These errors can be extended to total Cartesian error through the same method as above and averaged for two sensors (Fig 15).

![Total Offset Error Over Operational Swath(m)](image2)
Using the values from the both sensors, we can calculate the total RMS error (Fig 16).

Finally, using computational minimization techniques we can find the maximum error over the operational swath, which in this case it is 10.36 cm.
References


