Cedarville University RoboBoat 2014

Thomas Humbert (Team Leader), David Moeller, Greg DeVos, Joshua Kaster

Team Advisor: Dr. Timothy Tuinstra

ABSTRACT

The purpose of the Cedarville RoboBoat team was to build on last year's team in preparation for the Annual International AUVSI RoboBoat Competition. We reused the platform, thrusters, control hardware, and cameras from last year's boat. This year well evacuated the strengths and weakness of last year's boat and decided that we needed to focus on three main areas. The first were the core computing capabilities of the RoboBoat. We added new hardware to increase software speed and network reliability. Along with these hardware changes, all of the core software was rewritten to increase performance and remove dependencies. Our second area of focus was the navigational aspect of the competition, where we worked toward a consistent compass following, GPS waypoint following system and, reliable image following. The third and final focus of the team was the completion the challenges at the competition. We focused on the audio pinger challenge, automated docking challenge, and the obstacle challenge.

I. HARDWARE

1. Pontoons

This is the third year utilizing the pontoon style boat design. The pontoon style platform consists of two Echo Fisher pontoons from Venture Outdoors. These pontoons are composed of linear, low-density polyethylene (LLDP), which is much lighter than PVC. Each pontoon is 48” long by 12” wide by 10” high and weighs only 11 pounds. By removing a section from the original design in the middle of the seat and plastic welding the seat back together, the boat was made to fit within the three-foot limit.

2. Propulsion

The propulsion system consists of the same two Seabotix BTD-150 thrusters used the previous years. These thrusters were chosen because they are extremely small (6.927”x3.72”x3.673”), lightweight (1.58 *lbs* each), and easy to interface. At the maximum voltage, the thrusters can each produce 4.07 *lbs* of thrust per motor. Each thruster is mounted directly underneath the center of each pontoon. This design makes zero-radius turns possible using differential steering.

3. Electronics Enclosure

Our team also chose to reuse the electronics enclosure from previous years. The Nanuk 945 protective case was chosen because it is relatively light and the interior dimensions are large enough for all of our electronics.

4. Electronics Wiring

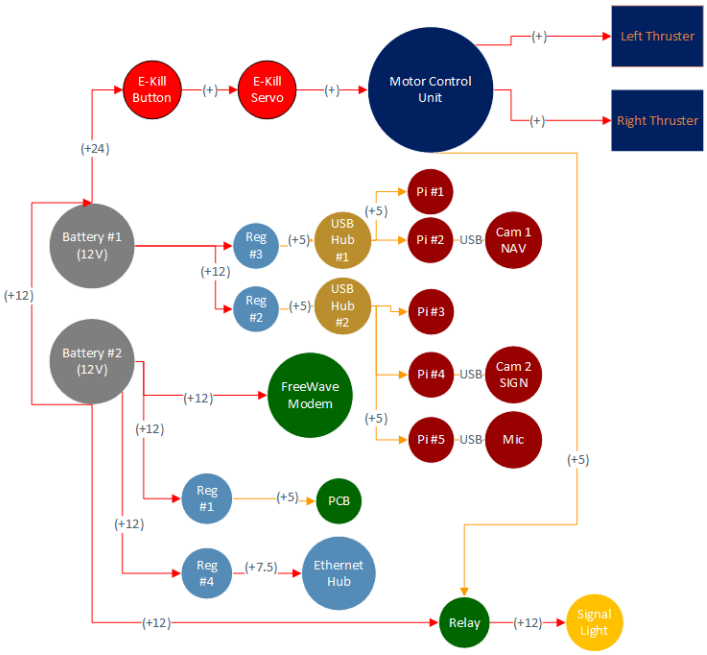

Figure 1. Electronics System Diagram

5. Global Positioning System (GPS)

Our GPS unit is an LS20031 receiver. This receiver was chosen last year. It is a Wide Area Augmentation System (WAAS) receiver with an accuracy of ±3 *m* (9.84 *ft*). This unit was chosen for its simplicity, in addition to providing sufficient precision. This year we decided to add a GPS to the boat. With two units we hope to be able to use an averaging algorithm to increase our precision.

6. Compass

The Devantech CMPS03 Magnetic Compass Module has 0.1 degree of resolution and 3-4 degrees of accuracy. The headings are accessed via I2C; the device is connected to the laptop by a USB port through an I2C to USB module. The Devantech compass' 19200 baud rate allows reception of data from the compass in real time, making it useful for navigation. The compass is a key part of our design; all of our navigation is based off the boat's ability to follow a given compass heading.

7. Webcams

The Microsoft Lifecam Studio Webcam was chosen because it is very easy to interface with Matlab through USB. This webcam also has many RGB resolution options. We determined a resolution of 320x240 was sufficient to extract needed data while still being able to processes the data in a reasonable time. The white balance, exposure, hue, and saturation are all set on the Raspberry Pi by loading predefined values every time the main program runs. This is set by sending commands over SSH to the Raspberry Pis. The cameras are mounted with a custom 3D printed mount which allows for a consistent camera position from run to run but is still adjust the angle and the height of the camera so that we could find the optimal viewing angle. The camera mount also holds the polarizing filter for the cameras (Figure 2).

Figure. 2 Electronics System Diagram

8. Laptop

The Dell Inspiron N4030 laptop is the main processor for our system. The laptop runs Matlab and performs all of the vision and data processing. Programming in Matlab, with its built-in Image Processing Toolbox, was much simpler than using a less abstract language like C. In Matlab, we are

able to interface with our microcontroller, compass, GPS, webcam, and network antenna simultaneously while concurrently doing all image processing and navigation computations in a single program.

9. Raspberry Pi Network

The Raspberry Pi network currently consists of five Raspberry Pis and a central laptop all connected using UDP (Figure 3). We installed an Ethernet switch in order to link all the communication lines together. Each Pi sends back pertinent data over a specific UDP port for the boat laptop to decipher and adjust the thrusters and camera servo. This allows the Pis to work independently while the boat laptop continues to work through each event. As the events transpire, they pull the information from the Pis when it is time to read from them again. This also removes the dependencies that were causing last year's team to get caught in loops and relying on the GPS for the update rate.

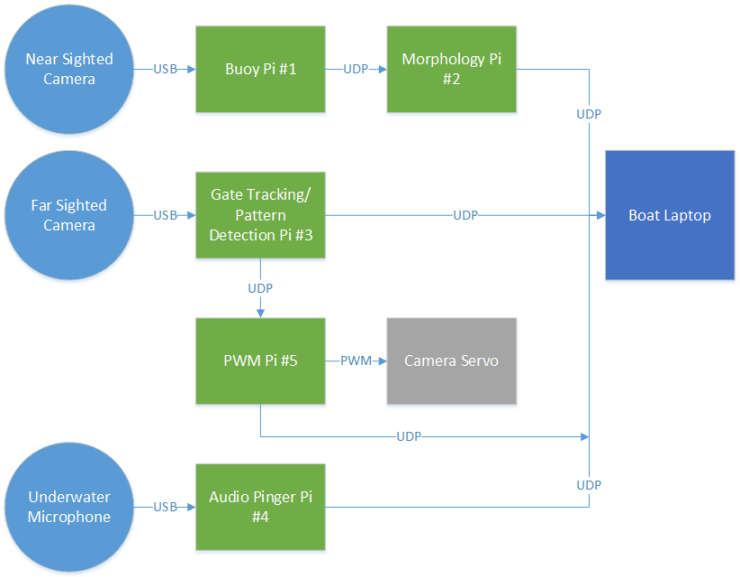

Figure 3. Data Flow of Raspberry Pi Network.

10. Microcontroller

The laptop provides data to control the thrusters. However, intermediate electronics are required. The Arduino microcontroller converts numerical values to pulse width modulation (PWM) signals for the motor controller. An Arduino was chosen because it is very common and has open source software. The Arduino also hooks directly to a virtual COM port on the laptop via USB.

11. Motor Controller

The Sabertooth 2x25 motor controller has two output channels capable of sourcing up to 25A. Another key feature of this choice is its ability to drive both thrusters independently, allowing for differential steering. This motor controller met the voltage and current specifications, contains built-in overcurrent and thermal protection. A servo throws a switch controlled by a channel on the remote control so the input to the motor controller can alternate between autonomous and remote modes.

12. Remote Control

Last year, a Futaba 7C 2.4 GHz remote control was chosen. We continue to use this remote due to its spread-spectrum technology. This feature makes the remote very reliable. Even in highly networked areas, like the competition conditions, we are able to connect easily and maintain a solid connection.

II. IMAGE PROCESSING

1. Color Detection

After brainstorming and developing several different means of detecting the colored buoys, we developed a system that first identifies the circles created by the buoys on the lake. The Audio Pinger and Obstacle Course both utilize buoy/color detection in some way. Using a metric, we only look at the circles that have the greatest confidence. Once detected, these circles are analyzed using thresholds from the HSV range and categorized into the different color bins desired to find. We then run a blob analyzer on it to find the area and the centroid of the desired color. This is sent back over UDP to the main laptop to drive toward.

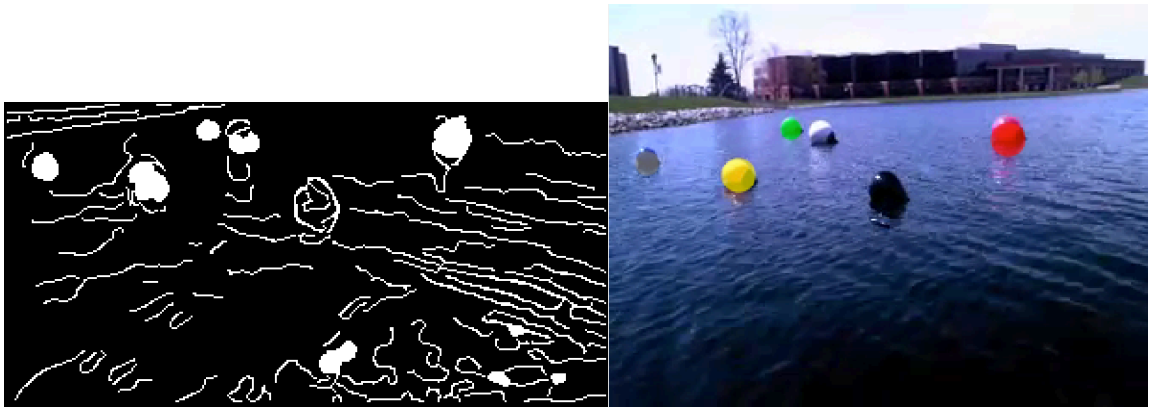

Figure 4. (a)The video feed after morphological operations to increase the strength in the shape of the buoy. (b) The resulting video feed with colored circles overlayed where it sees the buoys.

2. Pattern Detection

Part of the Automated Docking Challenge consists of being able to detect three different shapes, a circle, a triangle, and a cross. To do this we use the intensity of the current frame from the camera and run edge detection on this. Then we invert it so that the insides of the shape become enclosed blobs which blob analysis can pick up. This removes a lot of noise since many of the other edges do not make enclosed blobs. We then use the bounding box of the blob detected to pull out the selection from the original image. This provides a higher level of detail plus only run our shape analysis on the needed pixels saving on computational power.

Our shape analysis consists of corner detection for all the shapes. If it has the required number of corners and that the blobs are on the same horizontal level, then each individual shape has its own set of requirements. For the circle, triangle and cross, we use geometrical calculations to test for the radius, area and perimeter of the blob to see how close they resemble the actual shape. If the blob passes these tests, the data is then transmitted back to the laptop. The output of this algorithm can be seen in Figure 4. With our current system we are able to achieve 5.2 frames per second which is acceptable for this challenge.

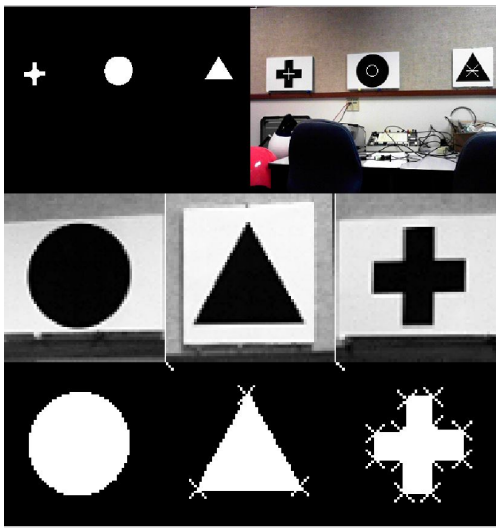

Figure 4. (Top Left) Possible Blobs. (Top Right) Raw Camera Feed with symbol-matched marker. (Center) Region of Interest for Each Symbol. (Bottom) Binary Image of Pattern with Corner Markers.

III. NAVIGATION

1. PD Controller

Because our thrusters generate different output thrusts given the same input voltage and because wind and other disturbances can throw the boat off course, a controller was designed to compensate for these problems. This controller will allow the boat to follow a given heading regardless of physical issues by giving a feedback from the compass. A PD controller gives an

approximation of the derivative of the error multiplied by a constant to the thruster value as well as the proportional value discussed earlier.

2. Speed Gates

To navigate the speed gates at the beginning of the course, heading information from the compass is used to navigate the speed gates in in minimal time without error. A Matlab function was developed that causes the boat to go in the direction of a given compass heading using the PD controller previously discussed. Using this function, the boat is simply pointed in the correct direction initially and follows that heading for a given amount of time before entering channel navigation mode.

3. Compass Navigation

Through testing, we know that our electronic compass is very accurate, and likely the most reliable sensor we have on the boat. Because of that, we wanted to base all of our movement algorithms on compass heading following. We had a PID algorithm and constants that we also inherited from last year's team, and decided to keep it in this year's design. After the main program was restructured to run its update code at a consistent rate, all of the problems that last year's team had with the algorithm went away, since the inconsistent update rate were making the PID calculations inaccurate. The boat can now follow a given heading with little to no detectable oscillation.

4. GPS Navigation

Another essential aspect to the boat's navigation on the lake is the ability to follow and arrive at GPS waypoints. Our general algorithm for doing this is to grab the boat's current coordinates and calculate the desired heading toward the desired waypoint. To improve the usability of the boat we added a waypoint editor with click-and-go functionality. This allows the user to quickly change and save new waypoint routes with a few simple clicks. The biggest problems with the GPS following that we have encountered have come from the inaccuracy of our GPS unit. We gathered data by walking around campus with the unit, and had trouble coming to a conclusion about the data's reliability. The coordinates can be reliable enough to use over short periods of time, but have a definite wander over longer periods. We investigated several ways to fix this but none of them were do able with our time frame or budget. Our final algorithm for GPS following involves updating the desired heading every two seconds to prevent drifting. In order to arrive at a GPS waypoint, the boat must calculate the distance between the current coordinates and the desired coordinates to be less than 6 meters.

5. Image Following Navigation

Image following is another critical part of our navigation. When the desired feature on a video frame is identified, the centroid of that feature is used to determine a new heading to follow. A heading differential is determined using the camera's known field of view and the number of pixels the centroid is from the center of the image. The heading differential is added to the boat's current heading to give the new heading.

IV. CHALLENGES

1. Obstacle avoidance challenge

To solve this challenge we have two separate models running at the same time. A stationary nearsighted camera that is used to avoid the small buoys in the field (the black and yellow buoys) and a gate tracking, farsighted camera used to know the location of the exit gate (the red, white, and green buoys).

The first task of the obstacle challenge is to dodge the buoys scattered about in the field. We accomplish this by running Color detection on the frame and separating the black, and yellow pixels into their respective bins. Then the bins pass through morphology and blob analysis, acquiring the centroids and areas of each blob. These centroids and areas are passed back to the boat. The boat then makes decisions based on this information.

While trying to solve the challenge the problem arose of knowing where our specific exit gate was especially when the boat was turned to dodge a buoy. With our GPS not being accurate enough we turned to image processing. We decided to have a camera attached to a servo that would track the exit gate giving the boat the ability to update the compass heading of the exit gate.

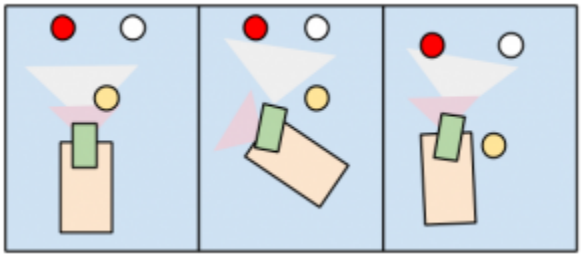

Figure 5. (Left) While Boat (tan) aims for Gate, it approaches obstacle (yellow). (Center) Boat dodges obstacle. Camera (green) tracks Gate. (Right) Boat realigns to Gate after successfully dodging obstacle. Gray triangle represents Gate Tracking Camera. Pink triangle represents near-sighted buoy.

We then run the Color Detection on the frame and only extract the white pixels and the other color. Since we will never need to see red and green at the same time we were able reduce the computations needed by only looking at either red or green (this color will be referenced as other).

The white and the other bin are run through morphology and blob detection passing to our gate tracking logic only the required centroids. Then based on the location of the midpoint between the two buoys and the previous duty cycle we are able to adjust the duty cycle to center the blobs. We only move the camera a little bit each frame since the image frames are processed faster than the servo can move. Since the Pi have general purpose input output (GPIO) pins we are able to output the pulse width modulation (PWM) for the servo using a model developed by Joshua Hurst. We designed and 3D printed a camera mount which holds the servo and allows the servo head to rotate the camera. This allows the camera to always be centered on the correct exit gate.

Then the duty cycle is converted into degrees and is sent to the boat so it can update the compass. This allows the boat to know where our specific exit is at all times even so that when it dodges a buoy is has a direction to go in. Currently, we have tested the PWM tracking model with promising results in the lab, but further development is needed and we have not yet been able to test this on the lake.

2. Autonomous Docking Challenge

After the shapes are detected on the Pi, The three shape centroids are sent back to the laptop, where they are used to determine boat's movement. The algorithm used to convert the centroid values to thruster values uses a variation of the PID compass following that we use for all of our navigation systems. The x-coordinate of the desired shape's centroid is scaled into a difference in heading according to the measured field of view of the camera. That change in heading is added to the desired heading every second if the current heading differs too much from the desired heading, or if the pixel location differs too much from the center of the image.

3. Acoustic Beacon Positioning

The underwater pinger challenge has several buoys with one of them emitting a chosen signal between 25 kHz and 40 kHz. We are required to find which buoy is emitting the frequency and then report the coordinates of the result.

We are completing this challenge by first using the color detection to find one of the colored buoys and then systematically going from one buoy to the next until all five colors are checked. Once at the buoy we will sit and listen for the frequency.

Using an underwater microphone, we run the input through a Raspberry Pi Simulink Model to filter out all signals except the desired frequency. After running through the filter, the sum of the result is added and mapped. This magnitude tells us how close and how strong the signal is. The closer we are to the pinger, the higher the magnitude.

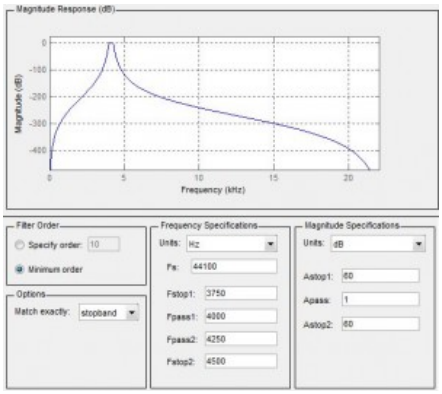

Figure 6. Customized Filter Design for Acoustic Beacon Positioning Challenge.

In order to compensate for the Raspberry Pis limitation of 44.1 kHz sampling rate, we are using the roll-down effect of the ultrasonic frequencies to identify the signal. For example we can listen for a 25 kHz signal and hear its alias under 4.103 kHz. If the HDMI monitor is hooked up to the Audio Pinger Pi, it will show the frequency it is seeking and the scope displaying the magnitude of the desired frequency. This model has been successfully tested under the audible frequency range. In the overall GUI, there is a slider that shows the magnitude of the filtered signal.

V. CONCLUSION

After analyzing the team's state from last year, we decided to remake all the coding and make it much more modular and less dependent on return values. The overall code is now event driven, utilizing Raspberry Pis to help delegate the tasks required for navigation and detection. After the rule changes, our greatest confidence is in the Pattern Detection challenge. We have attempted the Audio Pinger and Obstacle Course, but that is heavily dependent on our newly developed color/buoy detection system. We believe we will be able to navigate to the different challenges using GPS waypoints and compass headings.