Autonomous Surface Vehicle

Old Dominion University, Norfolk, Va

4th International RoboBoat Competition
8th – 12th June 2011
Founders Inn and Spa, Virginia Beach, Va

Table of Contents

Team Members

ODU faculty adviser	Dr. Gene Hou
Graduate Adviser	Justin Selfridge
Team Leader	Stanton Coffey
Project Design II members	Pat Gray
	Matt Cline
	Jack Nagle
	Sam Riahi
Project Design I members	Jeff Roper
	Brian Skoog
	Bobby Lee
	Stephanie McCarthy
	Paul Hart
	Drew Vaden
Electrical and Computer Engineers	Alex Chin
	Jason Armstrong
Elizabeth City State University	George Harris
	John Hayes

Overall structure

This years boat design is a pontoons boat. Above the two pontoons is the frame structure that everything gets mounted to. The things mounted to the frame are the motors, electrical box and the pontoons. The reason for this design came from learning mistakes from last years design. We built a metal boat hull with the frame internal to the hull. This provided problems because we had limited space inside the hull and made moving parts around and getting the proper weighting very difficult.

Frame

The frame of the boat is made from a series of fiberglass T-slotted bars. There are two bars that run the length of the boat, bow to stern, and three bars that run from port to starboard. Of those three bars, the bar in the stern of the boat we mounted both of the motors. On the other two bars in the bow of the boat is mounted the electrical box with forward facing cameras. The two long bars that run the length of the boat we use to mount the pontoons. The reason for using T-slotted bars is because of how easy it is to adjust and move things. When we first placed the boat in the water the weighting was completely wrong, so we just pulled the boat out and moved the electrical box and put the boat back in the water. The time it takes to re-adjust the weight is about five minutes.

Pontoons

There have been two models of pontoons built for our boat. The first model was built as practice pontoons, because we were having our final pontoons built by two people at Elizabeth City State University in North Carolina.

The first model pontoon was made by two foam pours, the same kind of foam used in surfboards. Each foam piece was four feet long by one foot high by six inches wide. These foam pieces were rounded smooth. Next, each foam piece was coated in fiberglass layers. The final step was to add mounting points to each pontoon. We did this by taking two fiberglass T-slot bars and inserting them in to the pontoons. We drilled six-inch deep holes in each pontoon and inserted the bars into the pontoons and filled with a fiberglass resin pour to anchor in each bar.

The second model pontoon was a hollow fiberglass design. The selected material for the pontoons is high thread count fiberglass matting with West System epoxy resin and hardener. Such material is extremely durable, lightweight, waterproof and resistant to water damage (corrosion), easy to work with, and affordable. The reason for not having a foam core was to cut down on weight and in the event of the foam getting wet and getting waterlogged and adding dead weight. The mount leg material is a square hollow aluminum bar. Each pontoon will have two mount legs and each leg is 1.5”x1.5”x24” and 0.125” thick with 0.375” holes spaced every 1.5” apart. Each pontoon is 10” in diameter and 5’ long. Each leg will protrude 14” out of the top of the pontoons. Six inches of leg is needed to mount the pontoon to the frame. The extra length on the leg is for possible future attachments to the boat. For this years boat there will be nothing mounted to the extra length on the legs. These pontoons will be used for multiple years to come so it was decided to keep the extra length on the legs.

Motors / Propeller Shields

Two Bass Pro Shops® Prowler® T30/30 Trolling Motors were used on our boat. The reason for using this motor is due to cost and size. The cost of this motor is only $100. The reason for two motors was from learned mistakes from last year's team. Last year there was only one motor and the method for turning was achieved with two water thrusters in the bow of the boat that would shoot water out to the sides to get the boat to turn. We decided that turning would be better and faster with two motors and just running one motor faster than the other to achieve a turn.

The propeller shields were designed to mount both to the motor shaft and the fin that extends below the blade. Norva Plastics in Norfolk built the propeller shields. The shields were made by quarter inch thick rolled plastic.

Electrical Box

The box chosen to house all of the electronics was a 16”x11”x6” plastic waterproof tackle box. This box was originally chosen because of it was waterproof and it could house all of our necessary electronics. After installing our electronics we found that we needed ventilation to keep our electronics from over heating. So some holes were cut in the lid. So the box is no longer waterproof, but still works great for holding our electronics.

The main component for the electronics is the laptop. The laptop is used for video processing, decision-making and the signal transmission to run the motors. All code written on the laptop is in C++. The cameras used are two basic Logitech web cameras. The cameras are mounted on the top of the electrical box in the bow of the boat. With two cameras we can achieve a 90 degree field of view. When the laptop has made a decision on how the motors should be run a signal is sent to the GPIO-24. The GPIO-24 is used to send out a PWM, pulsed width modulation, signal that is required to run the speed controllers that go to the motors. The power supplied to the GPIO-24 comes from the battery in the laptop. In an attempt to save the battery life of the laptop we send the PWM signal to an IFI robotics controller and turn off the GPIO-24. The IFI robotics controller gets its power from a JetSki battery within the electrical box and will continue the PWM signal. The PWM signal then gets sent to two 12Volt Victor 884 speed controllers by VEX. The speed controllers get their power from the 12Volt JetSki battery. Our code can run the motors in both forward and reverse.

Vision

Why vision is important

For autonomous vehicles to successfully interact with their environment, they must be able to gather an accurate assessment of their surroundings. Many sensors can give an autonomous vehicle the distance to objects or how fast the vehicle is moving, but this type of information does not help the vehicle understand what it is targeting and it does not tell the vehicle whether it is an object of interest or not. For an autonomous vehicle to truly interact with its environment, it must be able to discriminate between objects of interest and those that present obstacles. Drawing this distinction requires information about an object's shape, color and/or movement. Gathering this type of information is only possible through a robust vision system.

Vision General Description

A vision system for an autonomous vehicle is typically comprised of one or more cameras that collect screen shots of the vehicle's perspective, and then it sends those frames into a computer algorithm that applies image processing and data extraction algorithm routines. Unlike most sensors that monitor a single parameter, the vision system for an autonomous vehicle is a complicated task because it requires a certain amount of intelligence to process the incoming data. Whereas, most sensors provide a single value to monitor (range finders, GPS, compass, etc) the video stream consists of several thousand pixels composed of three values to represent the color. Instructing the machine how to process the data, and to make sense of the seemingly random numbers is the largest hurdle for any vision algorithm. Furthermore, all sensor data contains a certain amount of noise in the output. For single stream sensors discarding values outside a certain range, or implementing a running average will suffice to weed out the errors.

But for the vision system, the noise is often contained within the objects the system is designed to detect. Differentiating the noise from the useful data is no trivial task. The final, and often most significant, challenge is the processing time required to run the algorithm. Most streaming sensors can input their values very quickly; however, a typical vision system takes time to gather and perform calculations on the millions of values contained in a single image. If the vision system takes too long to perform the calculation, then the autonomous system will not be able to make decisions in real time.

Vision Problems Specific to ASVs

As if implementing a vision system for an autonomous vehicle was not challenging enough, the requirements for an ASV present additional challenges unique to these types of vehicles. Because the vision system is implemented on a boat, it is expected that the vehicle will have a certain amount of uncontrollable rocking and pitching. Based solely on the video input, the boat will be unable to determine whether the object is moving or whether the boat is rocking. This presents a unique problem as the vehicle attempts to track objects. Another hurdle common to ASVs is the glare and/or reflection from the water's surface. High intensity sunlight is difficult to process because the computer perceives that region as "white" rather than the actual color of the object. Also, on calm days the reflection off the water's surface can deceive the vehicle into falsely detecting objects.

Vision Main Tasks

For the AUVSI competition, the main tasks for the vision system are to identify objects of interest (buoys, markers, etc) and then deliver data to the navigation code so it can plot the boat's trajectory. The two main criteria used to distinguish between objects are the object's color and the object's shape. The buoys are marked as either red, green, yellow, or blue, and the shape of the object is used to distinguish the speed gates (rectangular shape) from the navigation cannel buoys (circular shape). By detecting both an object's shape and color, it is much more likely to accurately and correctly detect an object, and to eliminate false readings (red object that is not the correct shape). Object detection is the first task to be performed, but the navigation code still needs information to plan the boat's path. The vision system should be able to give some indication of the object's position and distance. Once the object is detected the vision system must determine where the object is located on the screen and how large the object is. The position of the object can be used in several ways. First, each column of the video screen approximately corresponds to an angle of the object with respect to the boat. This is a good starting point to determine how far "off-center" the boat is from the object. Similarly, each row of the video screen can provide basic information about an object's distance. Based on a perspective view, imagine the horizon as forming a horizontal line across the video screen. The objects of interest are all located close to the surface of the water and thus they will be located below the horizon line. From this we know that closer objects will appear lower in the screen than farther objects. Unfortunately, this technique cannot be used to reliably determine an absolute distance to the object because the boat is constantly pitching forward and back, which constantly changes the location of the horizon line. However, even as the boat pitches, the relative distance between objects does not change, so this technique can be used to determine which objects are closest to the boat, even if it does not convey the exact distance. Even though the previous technique does not provide distance to an object, a different approach can be used to gauge the distance of an object. Because the size of the buoys is known in advance, the system can evaluate the size of the object on the computer screen and calculate an estimate of the object's distance. For example, a 9" buoy will always be 9" no matter how far away the boat is. But the image on the screen grows larger as the buoy moves closer to the camera. The diameter of the buoy on the screen can be used with basic trigonometry to calculate the distance of the physical object.

Vision Implementation

The vision system for ODU's ASV was implemented with two web cameras, Microsoft Visual C++ 2010, and open source computer vision library called OpenCV, and a multichannel video stream library called videoImage. The webcams capture the available images and the videoImage library is used to tell the computer to work with multiple video streams. Once the images are captured, OpenCV is used for the image processing. The frames are converted from an RGB, red-green-blue, format into an HSV, hue-saturation-value, image. The pixels in this image are compared to several threshold values that bound the colors of interest. If a pixel meets the predetermined criteria for a particular color, that pixel is highlighted in a separate binary image. OpenCV then uses the outlines (or contours) of all the binary images. Each contour is then evaluated based on its size and shape. Any contours that do not pass the shape criteria are discarded. The contours that pass the shape criteria are then evaluated for their size and position, and this data is passed on to the navigation code.

Vision Conclusions

Vision processing is a complex task; however, it also has the potential to deliver the most data about the ASV’s surroundings. It not only provides data about objects to track, but it also allows the boat to make intelligent decisions based on the types of objects it detects.

Logic

The logic code is the final step to determining what the motor speeds need to be to get the boat to go where it needs to go. The logic code takes the outputs of the vision code, runs those through many if-elseif statements, and then returns values to set the motor speeds at.

The goal of this year's competition is to attempt tasks 3.1 – 3.4 (strength test, start gate, speed gate, and navigate the harbor). First the code is broken up into the different sections based on what we want to attempt. In our code there are ten sections: leave the dock, find the start gates, navigate the start gate and speed gate, find the navigation channel, navigate the channel, navigate the channel, make a U-turn at the blue buoy, return navigate the channel, return find the speed gates, return navigate the speed gate and start gate, and return to remote control mode.

The boat will operate in two different speeds. When the boat is in the speed gate section it will be moving faster and when the boat is in the navigation channel it will move slower. This is because in the speed gate section the buoys are spaced farther apart and it will allow us to move faster and have more time to make corrections. While in the navigation channel the buoys are closer together so we will need to move slower so we have more time to make adjustments.

The buoys that we will encounter on the course are big green, big red, small green, small red, small yellow and small blue buoys. The vision code will tell us what of those six buoys we see and don't see. Of the buoys we see, the vision code will tell us how far off center the buoy is from the boat center and how high it is in the picture frame. This will be all the information we need to navigate the course.

In the faster speed gate section we will be calculating the size of the gap between the larger speed gate buoys and how far off center, the error, the boat is from the center point between the two buoys. As the Gap appears bigger in our picture frame we know the boat is getting closer to passing through the buoys so we slow the boat down to give us some more control. The error is used to determine the difference in the two motor speeds, which will give us the right amount of turning to get us back on track.

In the slower navigation channel we bring the motor speeds down to the slowest possible speeds that still give us the ability to rotate and move. In the navigation channel there will be four buoys to look for, the small red green yellow and blue buoys. At any one time we could see none of them or we could see all of them or any combination in between. This makes sixteen different cases of what we could see. Once we determine what we see we enter that specific case. Within each of the sixteen cases there are many if-elseif statements for where the buoys are on the screen. Once it enters an if-statement a motor speed will be assigned to make corrections to what the cameras have seen.