UM::Autonomy Serenity, A Rigourously Refined ASV

Anthony Bonkoski, Eric Rossetti, Michelle Howard,
Dorian Brefort, Dominique Kudzia, Tyler Olsen,
Steve Ratkowiak, Steve Orloff, Adrian Choy

University of Michigan
Ann Arbor, MI 48109

Figure 1: Serenity

ABSTRACT

Serenity is a fully autonomous surface vehicle with a pontoon hull design for maximum stability, essential to the electronic systems. This boat was designed, built and tested for the purpose of competing in the Fifth Annual AUVSI Autonomous Surface Vehicle Competition, where it will exhibit its ability by attempting the challenges that have been presented for this year's poker themed competition. The main focus of the team this year was to redesign the vision and electrical systems. This paper explains the changes and adaptations that have been made to our boat since last year's competition.

1. INTRODUCTION

Serenity is the UM::Autonomy submission to the Fifth Annual AUVSI Autonomous Surface Vehicle Competition. Serenity was designed to execute the challenges of the Poker themed competition. Serenity is a fully-autonomous vehicle and has taken several years to build if we take into consideration the design time of Wolvemarine, Serenity's predecessor. Because of this, we cannot discuss every facet of our system in this paper. Therefore we will focus on the improvements that we have made since last year's competition. In the past few years we have emphasized the design, build, test methodology in the construction of our entries. We used the same successful approach this year. Learning from our successes and weaknesses of the past, we developed our latest iteration, Serenity. While Serenity appears similar to its predecessor, Wolvemarine, it has many changes, including an entirely new electrical system, vision system and the addition of some very powerful sensors. We have also implemented a novel and ground-breaking buoy chanel navigation algorithm and have spent countless hours rigorously testing our navigation capabilities.

Throughout the remainder of this paper we will discuss these modifications and improvements, as well as other ones that have been made since last year. These changes have made Serenity our most reliable and powerful submission to this competition.

2. HULLS AND DECK

2.1 Hull Design

We kept the same hulls as last year, but painted them maize and blue to better represent our school colors. The design evolved from a small waterplane area quadruple hull which we modified to become a twin hull. We made these modifications to the hulls shape to allow for better performance on the water. We designed the hulls to be more streamlined to help with tracking and reduce the water flow separation from the hull, hence reducing drag. Also, we redesigned the bow of the hulls to resemble the bow of a pontoon vessel. This design allows for the hulls to be semi-planing and allow the autonomous surface vehicle to travel at much faster speeds safely. The redesigned hulls are very stable in pitch, roll, and yaw and thus provide a stable platform for the vehicles electronic systems.

2.2 Hull Fabrication

The hulls were fabricated last year by using a male mold fiberglassing process. Foam molds were fabricated using a CNC Router, and then two layers of fiberglass were added.

Figure 2: Hull in the paint booth

This year, the paint scheme was updated to resemble the iconic Michigan "Winged Helmet." An epoxy layer was added to fix some damage from the previous years, and then another clear coat was added to waterproof and protect the new paint.

2.3 Deck Design

The deck of this vessel is similar to previous years with a few major changes. One of the changes that was implemented this year was the installation of ballast points at each corner of the deck. This allowed us to add and remove ballast weight easily. Another major change that was added this year was a bracing system attached to the bottom of the deck. This system prevents the hulls from shifting under the deck.

3. ELECTRICAL SYSTEM

3.1 Design

We almost completely re-designed our electrical system from previous years. We decided to go with a modular design that will make it easier for this system to be adapted for future challenges. One part of the design that did remain the same as Wolvemarine's was the motor power system and our power hub. Serenity's system has a completely new wire routing scheme and we converted to a one-PC system. We also add several new sensors and servos to make Serenity much more capable.

3.2 Box Layout

This year we put a lot of our time and energy into redesigning our electrical box. One of the main goals of this design was accessibility. We wanted to have easy access to all of the components, to allow for easier troubleshooting and repair. We also wanted this box to be simpler and more organized. We designed this box to be shorter and flatter to give us the accessibility we wanted as well as to lower the center of gravity, to make the boat more stable in the water. Another major improvement with this system is, we switched from running our software on two computers to one computer. This reduces the dependence on network comunication inside the electrical box as well as eliminates many software synchronization issues. We designed this new box to avoid having any components attached to the lid of the box for waterproofing purposes as well as to keep wires from being pinched in the hinges.

Figure 3: New Electrical Box

3.3 Cooling System

The new electrical box has a new cooling system that is much more efficient than the system that was used in previous years. This new cooling system consists of two fans, similar to last years, except the fans are located on the end of the box instead of the top. Another major difference is, both fans are facing the same direction, to double the airflow. The airflow now goes in the forward facing side of the box and out the rear facing side. This box has slits along the rear facing side to allow the hot air escape from the box. We also oriented the internal components of the box so that they would not block any airflow through the box. Also this year we chose a lighter colored box, to keep it from overheating in the sun.

3.4 Sensors and Servos

For dead reckoning purposes we rely on GPS, Compass, and Fiber-Optic Gyro sensors:

GPS: Garmin 16-HVS

Used primarily for velocity measurements to project the state forward

Compass: Ocean Server OS5000

Used primarily for pitch and roll measurements and to compute an initial heading for correlating GPS and Gyro measurement in our Extended Kalman-Filter SLAM.

Fiber-Optic Gyro: KVH DSP-3000

Used exclusively for yaw measurements as its superior drift rate of less than 1 degree per hour provides us with nearly perfect yaw estimation.

For perceptual purposes we use the following sensors (precise usages are explained in detail in Section 4):

Two Point Grey Fire-Fly MV CMOS cameras
Hokuyo UTM-30LX LIDAR actuated on a Dynamixel AX12 Servo
FLIR Tau 320 IR Camera for heat detection
MaxBotix Sonar sensor for underwater detection

We have added several servos to make our ASV much more capable. These include:

Two Dynamixel AX12 Servos for a pan/tilt water cannon aiming system
Two Dynamixel AX12 Servos for deploying a ramp and tethering an amphibious RC car

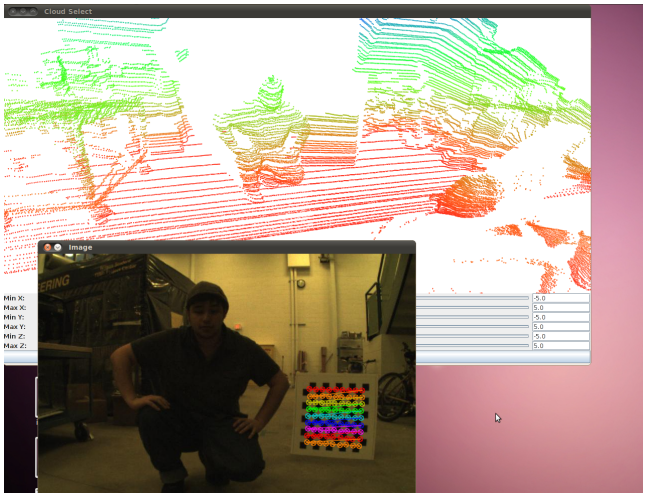


Figure 4: Point cloud visualization taken during a calibration session. Notice the checkerboard in the image and the corresponding plane in the point cloud. The goal of calibration is to isolate each and compute parameters for the transformation.

4. VISION SYSTEM

The basis for Serenity's vision system has its roots in both our 2010 and 2011 entries ("Mjolnir" and "Wolvemarine" respectively). Mjolnir relied heavily on image based detection while Wolvemarine relied heavily on 3D ranging data. By using inspiration from years past combined with some novel techniques, we have developed a very balanced and well-polished sensor fusion vision system. One could describe Serenity's approach as camera driven, while heavily supplemented by spacial 3D information.

4.1 Detection from Camera Images

A notable change from Wolvemarine is our increased reliance on camera-based detection. Wolvemarine attempted to detect all objects geometrically and used camera data only to discover object color. Serenity instead uses the image data to actively detect objects of interest.

We begin our buoy detection by performing a HSV based blob detection of each camera image. This process includes several filters and thresholds to reduce false detection rates. However, this alone is not enough to ensure reliable buoy detection. There are numerous situations that can cause false detections such as lighting conditions, vegetation on the shore, and pedestrians.

For the challenge stations, we have employed a variety of techniques including: HSV thresholding, blob dilation, line fitting, and other constraints. These will be further explained in the corresponding sections.

4.2 Point Cloud Generation

For spacial data we utilize a 3D lidar based system. Our Hokuyo UTM-30LX is actuated by a Dynamixel AX12. Using returns from both the servo and laser, we can construct a set of three dimensional range points, which we refer to as the point cloud. For a detailed explanation of point cloud construction see the corresponding sections in our competition paper for Wolvemarine.

For Serenity, we have optimized and improved the utility of our point cloud data. First, we mounted the lidar beneath the deck. This was done to eliminate returns from debris in the water and reduce the required rotation angle for the buoy channel. Second, we implemented a dynamic servo control system that allows the route planner to reconfigure the rotation range during a run. This new feature allows us to use a very small range for the buoy channel without sacrificing our ability to generate point clouds of challenge station tasks. By comparison, Serenity can produce buoy channel point clouds at 8 Hz while Wolvemarine could only produce them at less than 1 Hz. This improved point cloud frequency allows us to perform sensor fusion detections with both visual and 3D spacial information at a rate of 8 Hz!

4.3 Calibration

Before data correlation can be performed, we need to obtain an accurate calibration between the laser and camera. This step was also performed by Wolvemarine, but with only limited success. For Serenity, we have corrected the issues in the calibration procedure.

Calibration requires a large set of image/laser pairs that can be used to compute both camera intrinsics and the rigid-body transform between the sensors. We use the extrinsics model given in [5] and the paramters are computed using the method described in [3].

To collect the laser and camera data, we use a large checkerboard that can be easily detected with OpenCV. We also developed a new 3D point cloud visualization to replace last year's point cloud selector (See figure 4). Using this interface, the user can view the selected region from many different direction to make sure he/she isn't accidentally selecting incorrect points that were being occluded. This new point selector as well as correction of a few bugs in point cloud generation has yielded the very accurate calibration our approach requires.

4.4 Spacial and Visual Data Association

With an accurate calibration, we are now able to transform 3D ranging data directly into each camera's pixel space. This can be done with a simple rigid-body transform followed by some corrections for lens distortion as discussed in [5]. We perform all visual/spacial data associations in pixel space. Depending on the object that is being detected, different constraints are placed on the correlations. These are explained in the below sections.

5. FEATURE DETECTION

5.1 Buoy Detection

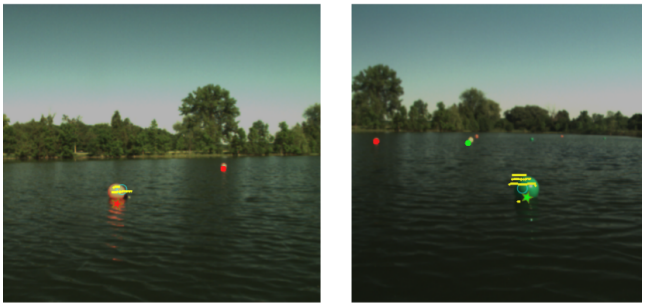


Figure 5: A Visualization of our buoy detection. The small red circles are blob detections. The yellow lines are points from the 3D point cloud. The colored stars indicate that laser points and blob detections have been correlated to produce a buoy detection.

For buoy detection we perform blob and point cloud correlation. By correlating blobs with transformed laser points within a reasonable radius and of reasonable quantity, one can develop a workable buoy detector. However, experimentally, this simple approach fails whenever the shore line is near. When this occurs, there are a plethora of laser points that will easily be correlated with blobs. From experiment,there tend to be many false detection blobs for the on-shore vegetation.

To reduce this vulnerability, we filter the point cloud data to discard large clusters of nearby points. We accomplish this goal using the Union-Find data-structure. We begin by reducing all points to an X-Y gridmap. We can then join nearby points to form several clusters. Next we can apply constraints on each cluster, such as size, to discard unlikely candidates. Clusters that are not discarded in the filter are then correlated with the image-based blobs as before. Because our laser range-finder doesn't return off the water surface, large clusters correspond to the shore, medium clusters correspond to buoys, and small clusters correspond to either noise or to debris in the water. By discarding all large and small clusters, we can be reasonably confident in the resulting correlation.

5.2 Poker Chip Station Detection

Detection begins by searching for linear orange blobs in the image. Each blob is checked for linearity using the minimum eigenvalue of the pixels' covariance matrix. Next, we fit a line to each linear blob and finally we search for two reasonably parallel lines. These parallel lines should be the ramp edges. Finally, using the lines, we estimate a rectangular region in which the ramp resides.

To compute the location and orientation of the ramp, in respect to the boat, we select all 3D laser points that fall within the rectangular region bounded by the fitted lines. These points can then be averaged to produce the location of the ramp, and we can perform plane and line fitting to compute the orientation of the ramp. To make additional guarantees on ramp existence, we require that there is a sufficient number of shore line laser points nearby (see section 7.2.1 on Shore Following)

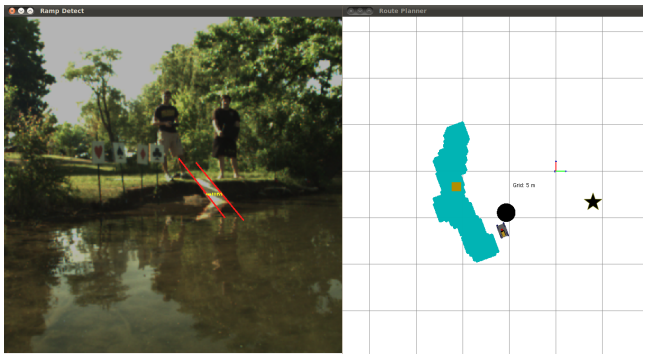


Figure 6: Left: Ramp detection Right: A visualization of the shore (in cyan)

5.3 Jackpot Station Detection

Detection of the jackpot station has been made a bit easier this year by redundancy. In previous years, there was only a single button and thus little contextual information to help reliable detection. Since there are now two buttons, we can use an approach similar to ramp detection to search for both simultaneously and thus have a better existential certainty.

We begin detection by finding all red blobs in the image. We then filter these results by applying size and circularity

constraints. Circularity is enforced by using the eigenvalues of the covariance matrix. Next, we try to match similarly sized blobs that are on approximately the same horizontal line. We then improve this static result by tracking these blobs between frames. This approach allows us to discard most noise and ensure a reliable detection. Since, these buttons are too small to produce laser returns, we utilize our shore map laser points to compute an approximate location of the buttons.

5.4 Cheater's Hand Station Detection

For the Cheater's Hand station, we are prepared to employ a couple different strategies, depending on competition dynamics. We intend to use the cards on the sign as unique features and perform either Template Matching, Nearest Neighbor Classification, or SURF (Speeded-up Robust Features) feature extraction. By requiring that all cards be adjacent and aligned, we can be certain that these features correspond to the station in question. We will use the same technique as the Poker Chip station to capture laser points and compute position and orientation.

When we are reasonably close to the sign, we will use the blue square to localize our aiming. We can extract the square in a way similar to ramp edge detection. Namely, we will extract blue lines and attempt to fit them into a square.

5.5 Hot Suite Station Detection

We will treat to Hot Suite station much like the Cheater's Hand station. Using a variety of feature detection techniques, we will search for each sign individually and then require a grouping of the individual detections. Using the calibration between the cameras and our IR Camera, we will correlate and choose the hottest sign for reporting. Incidentally, there are no special route planning required for this task, allowing us to send a wireless message to the ground station as soon as detection is complete!

6. SLAM

Serenity utilizes an Extended Kalman-Filter SLAM feature-based mapping system [4]. Since there are few things to localize on in the competition pond, we use a feature-based mapper and fall back on dead-reckoning when feature detections are not available for localization purposes. Due to the inclusion of the highly accurate Fiber Optic Gyro (FOG), we now use the FOG returns exclusively for heading information. On initialization, we collect a set of compass and fog observations and compute a "globalization" constant for the FOG measurements. This allows us to use both GPS and FOG data in a global frame for the purpose of dead reckoning.

For building data correspondences between buoy detections, we use the recursive Joint-Compatibility Branch and Bound algorithm [2]. Since challenge station features are unique, there is a known correspondence and data association is trivial.

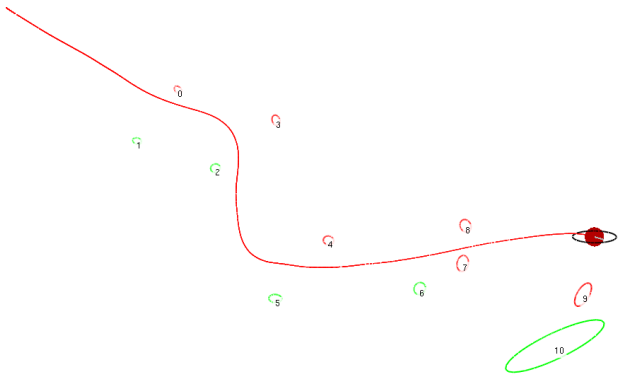


Figure 7: A Visualization of our SLAM map. Covariance is shown using an ellipse. Large ellipses correspond to high uncertainty.

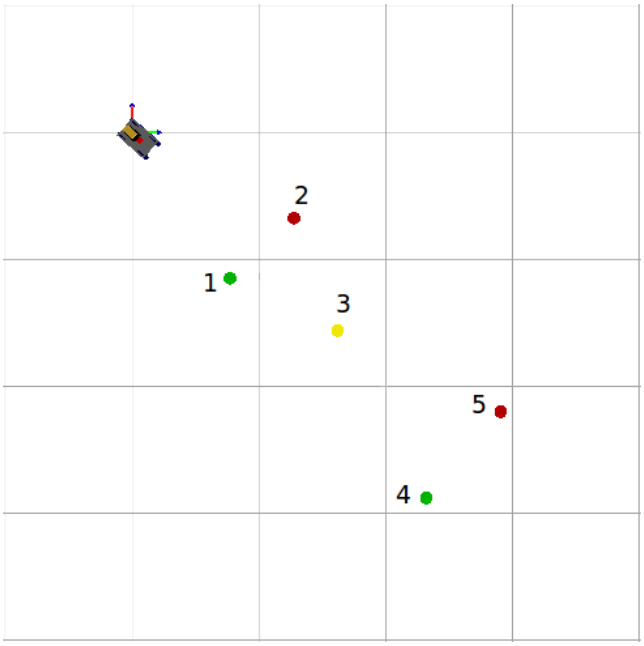
We have also added some capabilities for map correction. These include: detection and eliminantion of duplicate features and detection of map corruption. By maintaining a good map of the competition environment, we are afforded many advantages that are not possible with a simple short-term mapping technique, namely, adaptive buoy channel navigation. This will be discussed at length in the following section.

7. ROUTE PLANNING

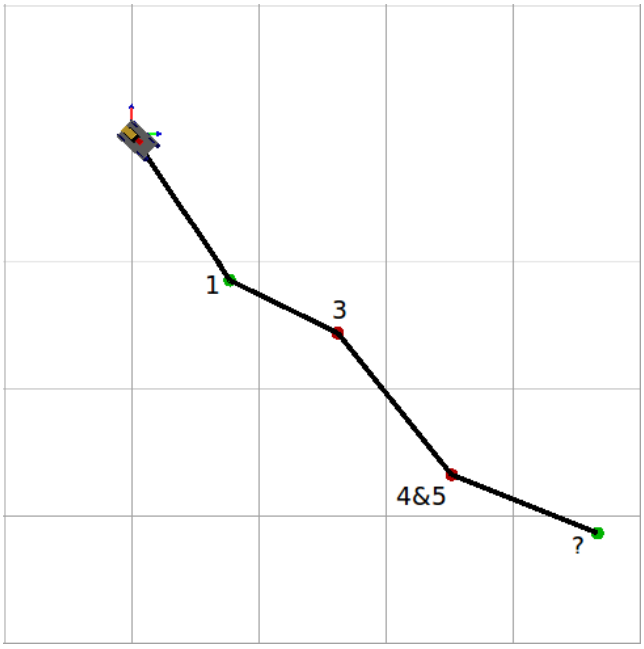
7.1 Speed Gate and Buoy Channel Planning

Having a reliable map allows many improvements to be made over simple instantaneous detection navigation. In short, each unique buoy that is detected and mapped tells something about the location and orientation of the buoy channel in general. This suggests that the location of the channel can be inferred directly from the buoy detections. Moreover, using some subset of the buoys, it should be possible to fit a line down the middle of the buoy channel. If a line can be accurately fitted to the buoy channel, navigation is reduced to the well-studied problem of line-following. This technique amounts to a higher order of planning, where buoys are considered in context to nearby buoys as opposed to a local statically-determined plan.

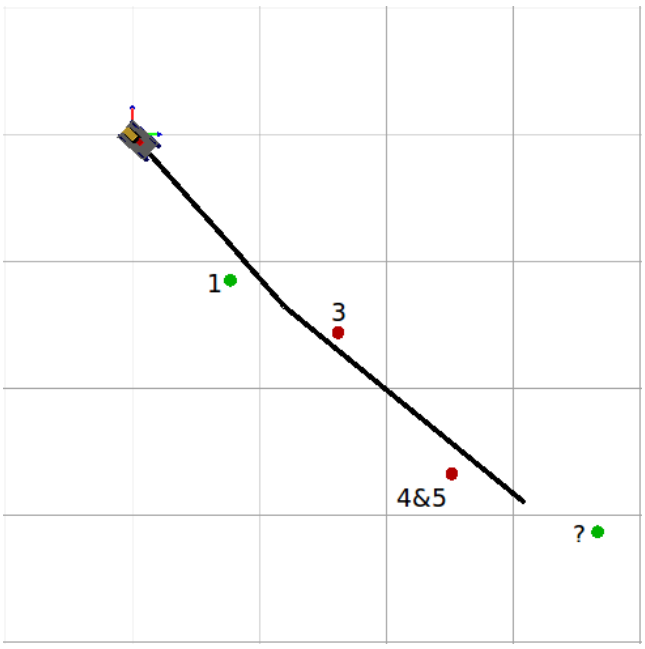
Next, we will show how such a goal can be achieved. We will assume that there is a low-level system that actively avoids obstacles (see section 7.3). Suppose we reduced the problem of buoy channel navigation to a problem of node traversal of a connected graph, where the nodes are buoys. We might construct such a graph by starting at the origin and drawing an edge to the nearest unconnected buoy. In this manner we can connect the entire set of known buoys. We now have a rough first-order approximation of the buoy



(a) Ground Truth of the buoy channel.



(b) 1st order approximation of buoy channel location. This buoy channel line very roughly approximates where one would expect it.



(c) Smoothed buoy channel line by allowing buoys #1 and #3 and buoy #4&5 and ? to be paired. Notice how accurate the buoy channel line is even in the face of such poorly identified buoys. Also, notice that this approach is completely independent of color as it uses spacial and angular metric to perform channel line fitting.

Figure 8: A Comparison of Approaches to Buoy Channel Navigation. Figure (a) shows the ground truth of the buoy channel. All other figures show how the channel is perceived. Notice that the boat fails to identify buoy #2 and miss-colors buoy #3 causing miss-pairing. Also, notice that the boat thinks buoy #4 and #5 are the same buoy and the boat hallucinated and an additional buoy is denoted as '?'

channel. If we attempted to navigate this graph, we would indeed navigate the channel, albeit rather humorously.

A buoy channel line is a composition of several line segments l_i which can be parameterized by angle and magnitude: (θ_i, d_i). From observation, a correct buoy channel line would be one that has a minimal error in the change of angle between the different line segments.
Mathematically:

Channel Error: $\sum_{i=2}^{n} \|\theta_i - \theta_{i-1}\|$

To make the channel line reasonable, we iteratively merge (pair) nodes by replacing them with a midpoint node. To accomplish this, the $pairable$ function is defined to decide when it is appropriate to pair various nodes. This function may pair buoys using a variety of possible heuristics such as color, distance, angle, etc. After merging a pair of nodes, the channel error is computed using the above summation and the result is compared with previous channel errors. The nodes are paired using a greedy approach, which, experimentally, appears sufficient.

This approach relies heavily on pairing nodes that will reduce the angular error. It's crucial that $pairable$ is reasonable with its pairing decisions. If $pairable$ is too liberal with pairing, the result will become a single straight line. If $pairable$ is too conservative, the resulting path will result in a zig-zag behavior. Neither of these are desired, although arguably neither is a complete system failure. Thus, it is important to choose a reasonable heuristic for pairing. Experimentally, we have found that the following approach works remarkably well:

1) Only allow buoy nodes to be merged (paired), essentially a midpoint node cannot be merged with any node.

2) Set a reasonably liberal distance threshold such that any node further than the threshold distance cannot be paired.

Note that the above heuristics do not utilize buoy color to make pairing decisions. An interesting side-effect of this algorithm is the ability to infer the buoy channel without color information. Using spacial and angular information, we can reliably estimate the location with only a subset of the buoys that denote the channel.

We have used this approach for every successful buoy channel navigation this year. It has been shown to be incredibly dynamic and adaptive to uncertain situations. Figure 8 gives an example of one of our early navigation attempts where the perceptual conditions were incredibly poor. It is important to note that Figure 8 represents actual data and that navigation was actually successful while being able to see only a fraction of the buoys.

7.2 Challenge Task Planning

7.2.1 Shore Following

To allow Serenity to search for challenge stations along the shore, we have implemented an occupancy gridmap-based shore mapping system. We first construct an X-Y gridmap from the point cloud data. The gridmap is then dilated to join as many nearby grid cells as possible. We then can find reasonably large grid cell clusters and reduce them to only edges.

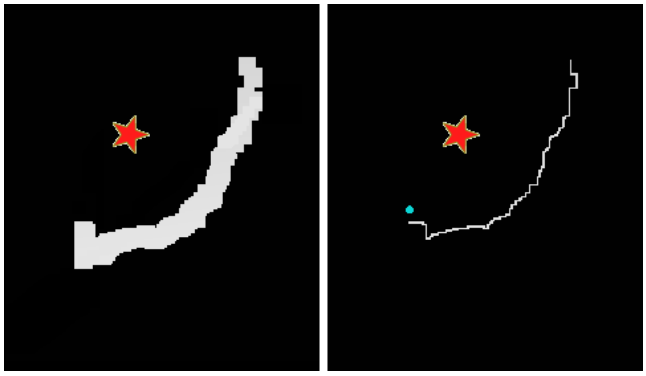


Figure 9: A Visualization of our shore detection algorithom. On the left: The dilated gridmap of all the laser points from the shore. On the right: The shore grid reduced to just the edge. A line can now be fitted to the edge. Notice the cyan circle: this circle is a waypoint computed for the purpose of shore following.

Next we eliminate all edges that would be occluded from our boat (these discarded edges are a result of the gridmap construction and dilation). At this point we have a single chain of gridmap cells that mimic the contours of the corresponding shoreline. As a final step, we perform a Least Squares line fitting to the resultant edge and choose a waypoint along the line.

After completing the buoy channel, Serenity will begin to follow the shoreline until a challenge station has been detected and mapped within a reasonable certainty.

As a conclusion, we can also use the shore mapping as an alternative approach to filtering laser data for buoy detection as discussed in section 5.1.

7.2.2 The Poker Chip

When the poker chip station is detected and mapped, Serenity navigates away from the shore to a more ideal vantage point (computed using the ramp's position and orientation). As Serenity approaches the ramp, instantaneous ramp detections will be used to perform PID alignment to the ramp. With the camera configuration, it's easy to know when the ramp has been reached. There should be a large linear orange segment on the edge of each camera.

When Serenity reaches the Poker Chip Station's ramp, our docking sequence will begin. This phase results in lowering a ramp mounted on the front of our ASV actuated by a Dynamixel AX12 servo. Using the servo's load feedback, we can detect when our ramp has been deployed.

We have strategically mounted our amphibious RC car on the edge of the ramp for ease of deployment. As soon as our ramp has made contact with the station's ramp, the RC car

has only about a foot to drive to reach the station's ramp and begin exploring for the poker chip. For ease of bringing the RC car back to the boat, we have employed a tethering system made possible by an additional Dynamixel AX12. Either when the car has successfully captured the poker chip or when failure is imminent, Serenity will be able to easily pull the RC car back to the mothership and continue the voyage.

7.2.3 *The Jackpot*

Serenity will approach the Jackpot station in a way similar to the Poker Chip, although arguably simplified. After the Jackpot station is mapped, Serenity will line up for an approach and begin navigating towards the station using the location in the map. As the station nears, instantaneous detections will be used to issue commands to the PID controller. By keeping the targeted button between the two cameras, we can align the boat to hit the button. When Serenity is sufficiently close, our sonar sensor will be used to check for the existence of the underwater buoy.

7.2.4 *The Cheater's Hand*

As discussed in the feature detection section, we will use the blue square to detect the precise location to aim our water cannon. Correlating with the point cloud, an (x, y, z) aiming coordinate will be extracted. We then compute the water cannon projectile to determine the appropriate angles to set out pan/tilt water-cannon system. The pan/tilt driver also provides status feedback so we can determine if the target is beyond the aiming range. We will use this feedback to reposition and correct as needed.

7.3 Obstacle Avoidance

By delaying obstacle avoidance to a later planning stage we are able to significantly simplify planning. We allow the individual planning systems to command any desired waypoint with little concern about how to reach such a locations. Our obstacle avoidance system then constructs an occupancy grid map. This map is dilated to allow the boat to be treated as a single point and than the Wavefront algorithm is used to compute the shortest path. We then select a fixed point on the path as a lookahead waypoint and we order the low-level navigation systems to this location.

8. UTILITIES

We use a variety of utilities to improve software development and make our entire software stack work together in harmony. These include a network message passing system, process manager, visualization library, and special calibration application. We discuss some of these tools below.

8.1 LCM: Lightweight Communications and Marshalling

LCM is a message passing system developed by MIT for the Darpa Urban Challenge in 2007 [1]. This system is specially designed for low-latency communications by using a simple UDP-based communication scheme. By defining specialized message types, bindings for C/C++, Java, and Python can be automatically generated and all messages can be automatically marshalled. This allows us to write high-performance code in C and take advantage of Java for high-level code and for visualizations. LCM also provides logging and playback capabilities which have proven to be invaluable for debugging.

8.2 Bot-Procman: Process Management

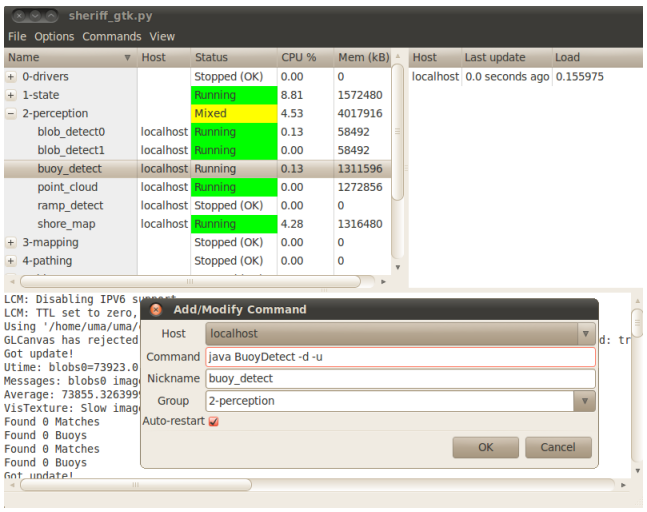


Figure 10: Bot-procman

One of the large failures in the execution of Wolvemarine was difficulty in launching due to the high number of processes that were required for our entire system to work. We utilized a process manager that was developed in-house, but was still a bit light on features and buggy. In addition, the process manager was a late addition to the design and had poor overall integration.

This year we've moved to the well-tested and proven bot-procman that was developed alongside LCM for the Darpa Urban Challenge. This process manager has many advantages such as: the usage of LCM as a backbone to manage processes across multiple machines, the ability to group processes for greater organization, and well-polished interface.

8.3 Vis: Visualization Library

One of the most crucial aspects in autonomous robotics is having a clear understanding of how your system is reacting to the environment and why. With only a simple $printf$ style debugging scheme, this goal is difficult if at all possible. Thus, developing versatile and intuitive visualizations is perhaps the most useful tool a roboticist can employ. For this purpose we use Vis, a 3D visualization library developed by the University of Michigan April Labortory (april.eecs.umich.edu). Vis is a Java library built around OpenGL that allows the user to construct very useful visual-

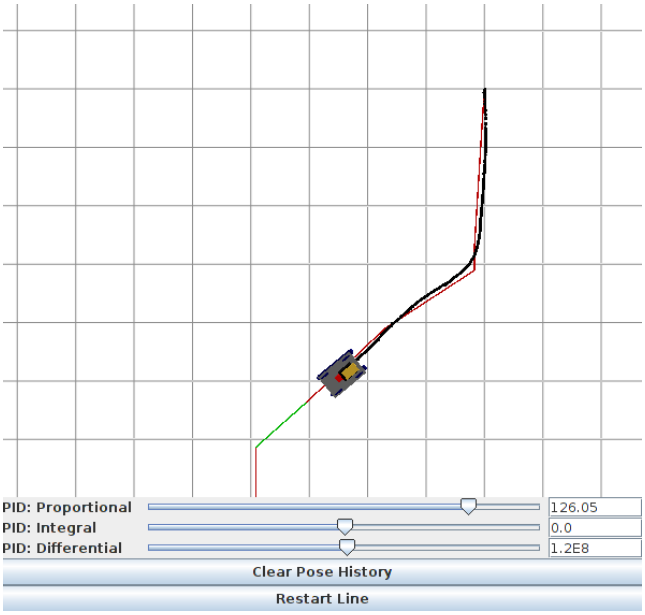


Figure 11: A screenshot of our PID tunning application. The desired path is red while the actual path is black. Notice the sliders at the bottom: these allow PID terms to be adjusted in real-time

izations with surprisingly little effort or background knowledge. Figures 4-9 and 11 were all constructed using Vis.

8.4 PID Tunning

One of the processes that prove frustrating to most developers is the process of PID tuning. From our competition showing with Wolvemarine, our team is clearly not without these troubles. Some part of these frustrations stem from the uncertainty about how the PID system is performing. It can be incredibly challenging to decide which PID term to adjust without having appropriate ground truth to compare against.

To help ease PID tuning, we have developed a special application to assist us. Our application allows the user to draw a poly-line of the desired path for the ASV to travel, and then it plots the actual path in real time. The application also allows the user to adjust PID terms within the application itself. These features enable quick PID tuning and allow the user to compare different settings quickly and effectively.

9. CONCLUSION

This year, we began with a goal to improve all the rough areas of past designs, and redesign and reinvent where necessary. We heavly stressed the importance of rigorous testing in an effort to construct an entry that performed as best as possible. We worked hard to remove as many failure modes as possible and to design a system that was easy to deploy and easy to debug by using visualizations and other software systems.

By rejuvenating our hulls, redesigning our electrical system, reworking our perception and planning techniques, we have developed a very refined autonomous vehicle. To this date Serenity has successfully navigated over ten buoy channels and our work on shore following and challenge station detection has shown much promise. We unequivocally believe that Serenity is UM::Autonomy's best Roboboat entry to date and we eagerly await this year's event.

10. REFERENCES

[1] A. Huang, E. Olson, and D. Moore. LCM: Lightweight communications and marshalling. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[2] J. Neira and J. Tardos. Data association in stochastic mapping using the joint compatibility test. *Robotics and Automation, IEEE Transactions on*, 17(6):890 –897, dec 2001.

[3] G. Pandey, J. McBride, S. Savarese, and R. Eustice. Extrinsic calibration of a 3d laser scanner and an omnidirectional camera. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*, volume 7, Leece, Italy, 2010.

[4] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[5] R. Unnikrishnan and M. Hebert. Fast extrinsic calibration of a laser rangefinder to a camera. Technical report.