University of Florida: Team NaviGator AMS

Daniel Frank, Andrew Gray, Kevin Allen, Tess Bianchi, Kipling Cohen, Daniel Dugger, Jake Easterling, Matthew Griessler, Sylvie Hyman, Matthew Langford, Ralph Leyva, Lucas Murphy, Jason Nezvadovitz, Anthony Olive, Blake Peterson, David Soto, Forrest Voight, Daniel Volya, Timothy Williams, Dr. Eric Schwartz, Dr. Carl Crane, Dr. Ira Hill, and Shannon Ridgeway

***Abstract*—NaviGator ASV is a fully autonomous surface vehicle (ASV) built to compete in the Association for Unmanned Vehicle Systems International (AUVSI) Foundation's 2016 Maritime RobotX Challenge in Oahu, Hawaii. The NaviGator ASV is part of a larger group of collaborative autonomous aerial, surface, and subsurface vehicles known as the NaviGator Autonomous Maritime System (AMS). This paper describes the NaviGator ASV's structural design, propulsion, power system, electrical design, software infrastructure, outreach efforts, and approach to completing the challenges presented in the 2016 Maritime RobotX Challenge.**

I. Introduction

The University of Florida's (UF) Team NaviGator AMS is a multidisciplinary group composed of undergraduate and graduate students from the departments of Electrical and Computer Engineering and Mechanical Engineering. This project is primarily sponsored by the Machine Intelligence Lab (MIL), which has nearly 20 years of experience in competing in the AUVSI Foundation's robotics competitions, including numerous championships in the RoboSub and RoboBoat Competitions. Due to the larger scale of the Maritime RobotX Challenge, MIL has partnered with the Center for Intelligent Machines and Robotics (CIMAR), a lab that has competed in three DARPA challenges and has extensive experience with developing highly intelligent large-scale autonomous vehicles. Between MIL's experience in autonomous maritime systems design and CIMAR's experience in software architecture design, Team NaviGator AMS feels that they have created a winning combination and look forward to competing in the Maritime RobotX Challenge.

II. Vehicle Design

This section of the paper will describe the hardware and software that was developed for this competition, as well as the motivations behind these choices. This will include descriptions of early iterations of hardware and software that may have failed, what was learned in that process, and how that knowledge was integrated to improve on the designs.

A. Mechanical Systems

The mechanical platform used for the NaviGator ASV is a modified WAM-V research vessel developed by Marine Advanced Research. Several of the mechanical modifications that the team has made will be detailed in this section. A computer-aided design (CAD) render of the NaviGator ASV is shown in Fig 1.

1) Propulsion: NaviGator ASV's propulsion system began as two forward-facing stern thrusters, providing the ASV with a skid-steer configuration. After a short time of testing, it became apparent that adding more thrusters and mounting them at an angle would simplify the vectoring of the thrust to achieve a desired motion, as well as adding the capability of lateral motion. The current configuration features two bow and two stern thrusters oriented at a fixed 45 degrees. This is a thruster configuration that the team used in the 2013 RoboBoat Competition with much success, earning first place. In addition to improved maneuverability, using four thrusters provides redundancy in the system, allowing the ASV to still have maneuverability even if either both bow thrusters or both stern thrusters fail. This feature was invaluable when a motor driver died minutes before a qualification run in the 2013 RoboBoat Competition. With a quick modification to the thruster mapper program, the ASV was able to operate with just three thrusters, saving the run. The major disadvantage of this configuration is that the fixed angles of the thrusters means that it is not particularly efficient moving in any direction. However, for the tasks that the Navigator ASV is designed to perform, maneuverability is significantly more important than efficiency.

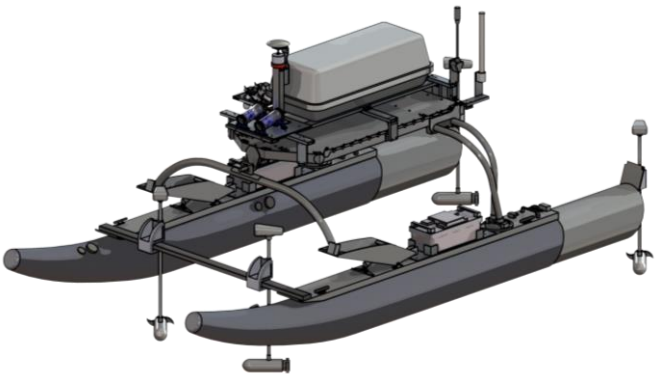

Fig 1. CAD render of the NaviGator ASV

Mounting the thrusters posed many challenges and required several design iterations, especially for the bow thrusters. For the ASV to be deployed from a trailer, the bow thrusters had to be either removed or raised during deployment so they would not collide with the trailer structure. The transom clamps on the trolling motors accommodated this function. 3D printed polycarbonate clamping blocks that interfaced with the

clamps on the trolling motors kept them fixed in place. While the mounts held the motors securely, the 3D printed parts began to crack and eventually failed. To solve this issue, the clamping blocks were machined from aluminum.

2) Sensor Mast: The need for a stable sensor platform is paramount in machine vision applications. The preliminary design utilized an 80/20 aluminum rail truss, which did not provide the required stiffness and resulted in smearing of the vessel's detection data. The initial sensor platform also did not raise the LIDAR system high enough to permit detection of obstacles in immediate proximity to the pontoons, a problem rectified in the final design.

As previously mentioned, the cameras, LIDAR, and GPS antenna require a rigid support. The need for an unobstructed GPS antenna guided the design towards a mast structure. For transport to the competition site, the assembly had to fit within the prescribed envelope of a Pelican Products transport case, requiring a modular assembly process. These target specifications led to a base-and-tree assembly, where the mast is simply welded to a plate that then fastens to the payload tray via a superstructure. For corrosion resistance and manufacturability, 6063 aluminum was chosen. To simplify the assembly process, fastener types were standardized. The mast is centered laterally on the ASV, which helps create a well-defined coordinate system that permits simpler software transformations. The sensor mast can be seen in Fig 2.

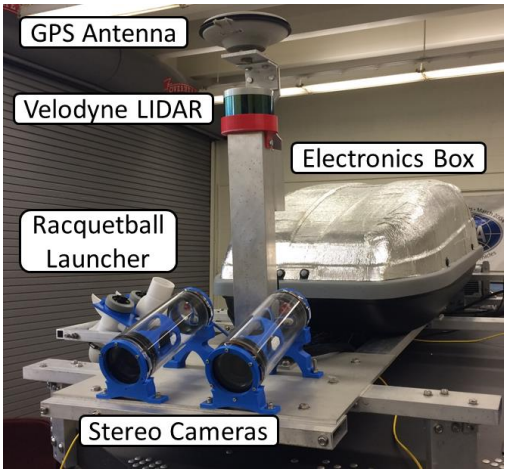

Fig 2. The NaviGator ASV's sensor mast

3) Electronics Box: NaviGator ASV's electronics are housed in a Thule Sidekick cargo box. The team originally considered commercial waterproof boxes, but began looking for other options due to their high costs. One student suggested the idea of using a cargo box after being inspired by family road trips they had taken when they were younger. While traditionally used to mount on the top of cars to provide additional storage, the cargo box was an ideal electronics enclosure due to its watertight integrity, aerodynamic form factor, low cost, and a side-opening mechanism that makes it very easy to access all of the electronic components.

The box's watertight integrity prevented the team from using air circulation for cooling. Instead, a combination of techniques are used to cool the box. First, an adhesive reflective covering was applied to the lid of the box to reflect heat generated by solar radiation. Second, the box has an active water cooling system that is used to remove the heat generated from the electronic components inside the box.

Fiberglass inserts were used to mount all of the components inside of the box. These inserts add rigidity to the relatively flimsy box and make it easy to add or remove components from the box. The components that need to be frequently removed, e.g., the hard drives, are attached to the fiberglass with Velcro. The rest of the components are attached with traditional fasteners.

4) Racquetball Launcher: A system for delivering the racquetballs into the target for the Detect and Deliver task was developed by breaking the challenge into subtasks that were solved independently. The two main subtasks that were considered were moving the balls into the target and feeding the balls to the mover. Several ideas for moving the balls were considered, ranging from a catapult to a robotic arm that would drop the balls into the target. Prototypes of several designs were built and tested. One design featured two counter-rotating wheels attached to the trolling motors that were once part of PropaGator 1, the team's submission to the 2013 RoboBoat Competition. The trolling motors were originally used as part of an early prototype, but since they were already waterproof, were effective at launching the balls consistently, and were readily available, the trolling motors were incorporated into the final design.

The ball launching mechanism was designed so that any type of ball feeder could be integrated into it. This allowed the team to test multiple types of ball feeder mechanisms, including a carousel and a linear actuator. A prototype of the linear actuator racquetball launcher can be seen in Fig 3. After the carousel mechanism was found to be prone to jamming, the linear actuator design was selected. Additional design criteria that were considered were the ease of loading balls and how quickly all four balls could be launched. Ball loading was addressed by designing a spring-loaded 3D printed ball magazine that is easily detachable. To achieve rapid-fire, the team developed a closed-bolt system. After a ball is loaded into the chamber, it is withheld at the minimum distance from the wheels to reduce the amount of time it takes to fire. In order to prevent premature firing, a retention lip is used.

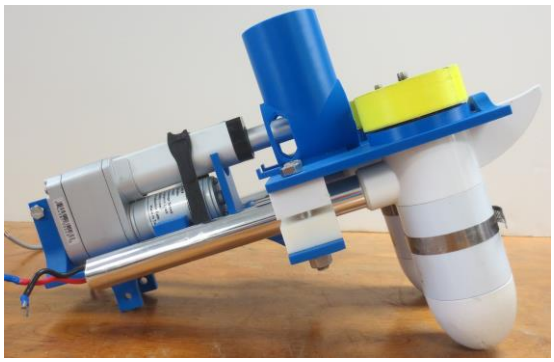

Fig 3. Prototype of the linear actuator racquetball launcher

5) Anglerfish: The Anglerfish is a remotely operated vehicle (ROV) designed to assist NaviGator ASV in completing the Underwater Shape Identification and the Find the Break challenges. Anglerfish [1], shown in Fig 4, is designed to be controlled by the NaviGator ASV via a 30 meter tether that provides both power and two-way communication to the ROV. The tether is also strong enough to be used as a method to recover the submersible.

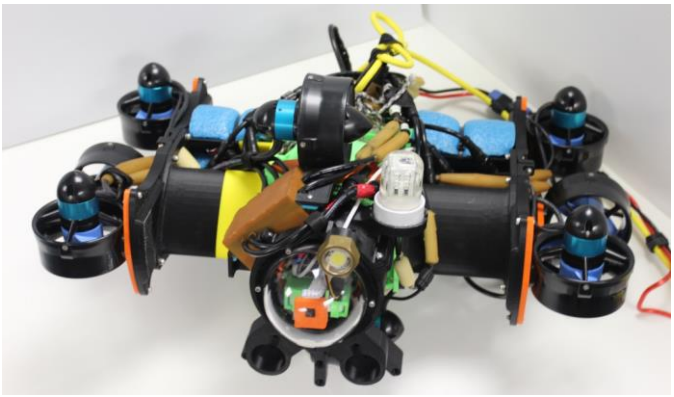

Fig 4. The Anglerfish ROV, a student-designed vessel created to assist the NaviGator ASV in the Maritime RobotX Challenge

The preliminary tasks descriptions of the competition stated that there would be a denied area above the Find the Break challenge. In this challenge, Anglerfish would be deployed from NaviGator ASV using a spool system. Once in the water, Anglerfish would descend to a pre-defined depth. Next, it would use a camera to search for the objects on the bottom of competition field. Once an object was found, it would report the object type and the object's position in order to complete the challenge. However, the latest version of the rules removed the denied area. This change on the constraints of the challenge opened the door for much simpler solutions, making Anglerfish obsolete.

6) Camera Deployment System: Once the denied area restriction was lifted for the Find the Break challenge, the team decided to use a simpler method to solve this task that did not include the use of Anglerfish. The solution that was developed is a deployable underwater camera that is able to observe the bottom of the course. This mechanism consists of a rigid beam that extends vertically into the water that is attached to the support beam between the pontoons of the NaviGator ASV. It is also possible to manually store and deploy the camera through a simple bolt system.

B. *Electrical Systems*

Robustness and simplicity were the primary motivating factors behind the design of the NaviGator ASV's electrical system. The team focused on these aspects in order to get a testable system built quickly and minimize any downtime due to electrical failure.

1) Power System: The salient features of the power system are the dual battery power supply and the power merge board. The NaviGator ASV's power requirements surpassed those of MIL's other projects in both the amount of power and required power source durability. Solving these challenges required looking outside the typical battery suppliers MIL used in the past. To be able to supply adequate power to four thrusters, computers, sensors, and communication hardware, the ASV uses two Torqueedo Power 26-104 batteries. Each battery powers two of the thrusters and contributes to the power rail that powers all other devices on the NaviGator ASV.

2) Student-designed Electronics:

a) Power merge board: The power merge board is a student-designed printed circuit board assembly (PCBA). It uses two Texas Instruments LM5050 High Side OR-ing FET controllers as ideal diode rectifiers to balance and parallel the two batteries into one rail that supplies four output ports. This makes the system more fault tolerant to a failing battery, a feature used in normal operation to switch batteries out without turning the system off. One of the strengths of MIL is the ability to design hardware and software that can be reused on other projects and vehicles. This is the third vehicle for which this board design has been utilized. The design was originally created for PropaGator 1 and then used on PropaGator 2, both of which have competed in the RoboBoat Competition.

b) Passive sonar: The passive sonar is a student-designed PCBA that implements a signal conditioner with for four Reson hydrophones. The timing information from the hydrophones is put in a buffer and sent to the computer for processing. The hydrophone PCBA design has proven to be extremely versatile in that NaviGator ASV will be the sixth autonomous vehicle from MIL to use this passive sonar design in an AUVSI Foundation competition.

c) Kill system: The hardware kill system consists of two student-designed PCBAs and four off-the-shelf twist to detent kill switches. The kill system for the vehicle also has a software component. The kill board monitors the status of six kill sources. When any of the six sources request a kill, the kill board cuts power to the thruster motor controllers. The six kill sources are the four off-the-shelf switches that are mounted around the vehicle, a remote kill switch, and the computer. The remote kill switch operates over a 900 MHz radio link and displays the hardware kill status of the vehicle. The kill board is also used to control the NaviGator ASV's indicator lights and a siren used to ward off curious watercraft during testing.

C. *Software Systems*

The NaviGator ASV utilizes an open source software environment known as the Robot Operating System (ROS). The team has been using ROS for their entries to the RoboBoat and RoboSub Competitions since 2012 and the students in the lab are active contributors to the ROS community. ROS was chosen as the programming architecture for NaviGator ASV because it compartmentalizes specific pieces of executable code into nodes. This allows the team to use software that was developed in previous competitions to be used on the NaviGator ASV, greatly reducing development

time. For example, much of the stereo vision software that is used on the ASV, was ported directly from code that was developed last year on SubjuGator 8, the lab's latest entry for the RoboSub Competition.

1) Object Detection and Classification: The lowest level perception service available on the NaviGator ASV is the Occupancy Grid Server. Occupancy grids are a two-dimensional grid-like representation of the environment generated by the sensor suite available on the ASV. The generated map contains both the occupied and unoccupied regions in the environment. This information is provided to the server via any range-detecting sensor onboard. On the ASV, the primary range-detecting sensor is a Velodyne VLP-16 LIDAR. A LIDAR uses lasers to provide relatively dense range information of the environment. This information is then segmented by regions containing dense clusters of relatively close points. These bounding regions are treated as obstacles, and are placed in the occupancy grid. This information is then provided to higher level services such as the motion planner and Classification Server.

In the Classification Server, the points generated by the LIDAR are clustered into regions on the occupancy grid where it decides which of these distinct regions are objects. The ASV then looks at the bounding box of this object and classifies the object based on the dimensions of its bounding box. The software detects if the object has a prominent plane. If it does, then this information is attached to the object. These objects are then accessible to other programs through the use of a list of detected objects.

2) Motion Planning: Motion planning is the process of finding a sequence of inputs that bring a given system from its initial state to a goal state. In most cases, there are an infinite number of ways for the same goal to be achieved. Ambiguity among these solutions can be reduced in a useful way by adding two more criteria to the problem statement; optimality and constraints.

Typically, optimality is characterized by a scalar function that assigns a cost to every possible state-input pair. Optimal solutions are the ones which also have the smallest total cost. Constraints are characterized by a binary function which classifies every possible state-input pair as either being allowable or forbidden. Allowable solutions are the ones which are allowable at every step.

For the NaviGator ASV, a goal state is defined as the desired GPS waypoint and vehicle heading with zero linear and angular velocity. Succinctly, achieving a goal means coming to rest at some new desired pose. An allowable plan is abstractly defined as one that does not collide with another object, and an optimal plan is defined as one which finishes in minimal time. The exact way these criteria are mathematically formulated depends on the motion planning algorithm being used. Like any planner, it is necessary to define the dynamics by which the ASV's state evolves with time.

The spin speeds of the ASV's thrusters can be varied fast enough to be treated as the inputs to the physical system. The thrusters apply forces to the ASV, so that control is said to be at the acceleration-level, i.e., the ASV's position as well as linear and angular velocities must be included in its state vector. The model for the vehicle was determined using the marine surface-craft dynamics provided in [2].

The dynamics of this model include the inertial effects of both the ASV and the water it pushes with it, as well as a quadratic hydrodynamic drag model. Parameter values for the ASV were first approximated by applying known forces to the system and trying to match simulated velocities to actual velocities. The results of this parameter-tuning were later refined with rigorous system identification methods. Regardless, some model uncertainty is acceptable due to feedback in the control architecture, which will be explained in detail in the Motion Control section of this paper.

To solve the now defined planning problem, first the team implemented the popular differential dynamic programming (DDP) algorithm given in [3]. DDP does not directly invoke an allowable-or-forbidden function for hard constraints. Rather, obstacles have to be encoded in the cost function itself as areas of high cost. DDP starts with a guess at the optimal sequence of inputs, performs a forward-pass where it simulates the dynamics over that sequence of inputs, and then performs a backward-pass where it recursively perturbs each input value according to the local cost of its associated state.

Each local cost region is approximated as a quadratic, so essentially the algorithm is an iterative linear quadratic regulator (LQR) solver for an arbitrary global cost function. However, the cost function still needs to be twice differentiable, a restriction that led to some inadequacies. An example of each iteration of DDP being used to navigate a buoy field can be seen in Fig 5. It should be noted that what is shown is only the 2D positional cross-section of the full cost field, which is really defined on the full 6D state-space.

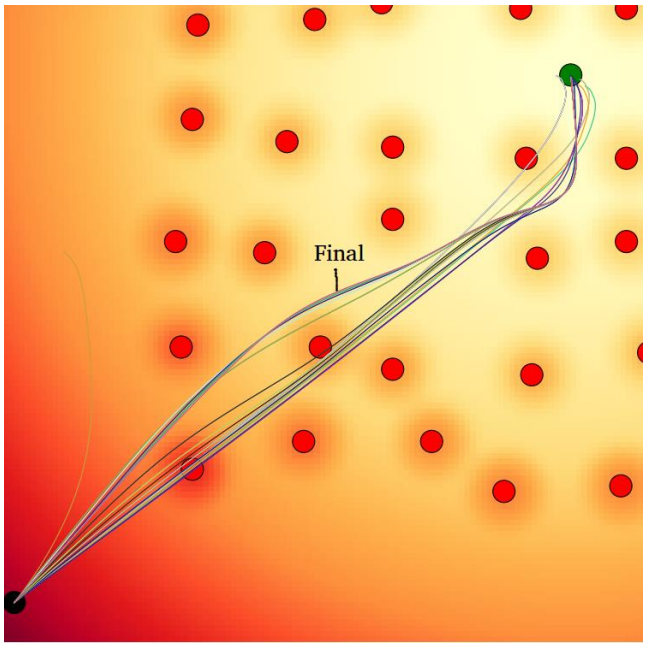

Fig 5. An example of each iteration of the DDP algorithm used for navigating a buoy field

Obstacles cannot be represented in the DDP cost function as steep, well-defined spikes or walls of cost. They have to be smooth enough to allow for stable numerical differentiation. For example, the buoy field cost function is defined in (1).

$$l = e^T Q e + u^T R u + \sum_{i=1}^{N} \left(\frac{A_i}{2} \left(\cos\left(\frac{\pi d_i}{r_i} \right) + 1 \right) w(r_i) \right) \quad (1)$$

In this cost function, e is the vector from the current state to the goal state, u is the control input vector, d_i is the distance from the current state's position to the ith buoy center, r_i is the buoy's radius, $w(*)$ is a 2D windowing function, N is the number of buoys, and Q, R, and A_i are weights on goal error, effort, and buoy nearness respectively.

By representing each obstacle as a cosine hump superimposed on a quadratic bowl centered on the goal state, convergence to the optimal solution was numerically stable. However, whether or not the plan truly collided with a buoy, i.e., by crossing too close into a fuzzy boundary, depended strongly on proper tuning of A_i. Fundamentally, the DDP solver guarantees local optimality but requires a very well-advised cost function to guarantee plans that do not actually collide. Another problem is its lack of flexibility. For example, a special new cost term has to be conjured up for each shape of obstacle that is expected to be encountered.

For a safer and more flexible planner, the team sought out an algorithm that can handle strict, well-defined constraints. The rapidly-exploring random tree (RRT) algorithm is highly efficient for this scenario [4]. The algorithm starts with a seed node at the ASV's initial state. It then randomly samples a state in the region of navigational interest. A nearness function is applied to every node currently in the tree, and then that node is extended or steered towards the random state following a policy function. The endpoint of that extension is added as a new node to the tree only if it is allowable, and the algorithm repeats. If an extension, or any intermediate state leading up to it, is not allowable, that iteration is simply abandoned. Once a node reaches the goal region, the tree is efficiently climbed from the goal back to the seed, and is classified as one solution to the planning problem. The best of the found solutions is defined as the one that takes the least amount of time. The goal region is likely to be reached because one can bias tree-growth towards it by shaping the probability density function from which random states are sampled.

Since every portion of the tree is only retained if it is allowable, all paths generated are guaranteed to obey every hard constraint. Additionally, constraints like obstacles can be easily encoded as an occupancy-grid look-up table inside the allowable-or-forbidden function. Most RRT implementations differ in what they do for the nearness and steer functions. The team primarily used the popular LQR-RRT* methods given in [5]. An example of the NaviGator ASV's RRT planning towards a goal region is shown in Fig 6. Obstacles are expressed as exact forbidden regions for which no portion of the ASV can enter. The tree only contains these allowable states, so safety is guaranteed. However, as randomness is intrinsic to the planner, it is only probabilistically optimal.

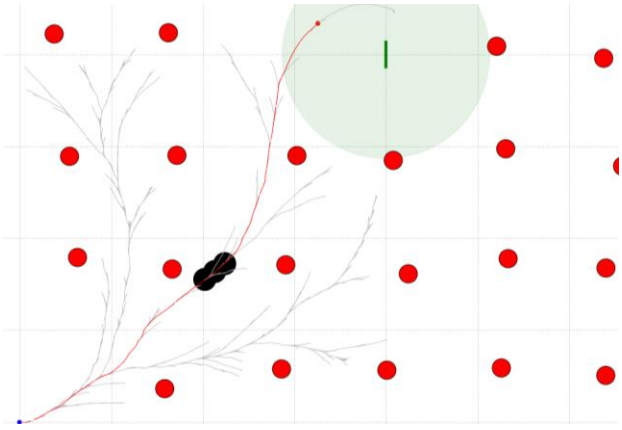

Fig 6. An example of the NaviGator ASV's RRT planning towards a goal region

After selecting the RRT algorithm for safety and flexibility, the final step was to integrate the algorithm with a real-time system. One of the biggest difficulties in doing this was dealing with a highly nonstatic environment. Obstacles spontaneously appear when they get in range of the perception system. This means that a valid path can suddenly become invalid with only seconds to spare. To make efficient use of time, the planner should always be planning the next move so that the RRT has more time to get a better solution. To handle this, the planner had to be made asynchronously interruptible, and a lot of plan-reevaluation and crisis-aversion logic had to be built in to elegantly deal with spontaneously appearing and/or moving obstacles that cross the ASV's current path.

The ASV's real-time ROS-integrated RRT algorithm being run for an arbitrarily drawn, complicated occupancy grid can be seen in Fig 7. Tree nodes can be seen in blue. The ASV was only given one second to plan its first move. It used its time during the first move to plan its second move, shown in red. While the paths generated using this method are safe and useful for solving the problem of navigation in the competition, the team is actively working on improved heuristics for smoothing out the paths.

Fig 7. The NaviGator ASV's real-time ROS-integrated RRT algorithm being run on an arbitrarily drawn, complicated occupancy grid

3) Motion Control: Since the RRT motion planner uses a model of the ASV, in principle it would be possible to employ

a model-predictive control architecture in which the ASV rapidly re-plans from its current state to steer it back onto the desired path. However, due to the randomness inherent to the RRT itself, such a method did not work well in practice. Thus, the team opted to make use of the sequence of states generated by the motion planner rather than the inputs to define the reference a feedback controller tracks.

First, a simple manually-tuned full-state feedback PD controller was used. Tracking along straight paths was nearly perfect with this alone, providing a positional steady-state error of less than 0.25 meters. However, along curves, a larger positional steady-state error of a few meters would always emerge depending on the curvature. Even the introduction of a standard integral term did not fix this problem.

The team figured that this was because an integral of the world-frame error alone would only be able to compensate for disturbances that are constant in the world-frame. Simulation revealed that the sources of the curved motion disturbances were centripetal-Coriolis effects and heading-dependent drag forces. A more intelligent integrator would be necessary to compensate for these state-dependent disturbances. Most marine and aerial systems accomplish this by using a model-reference adaptive control (MRAC) architecture. A block diagram of the MRAC controller used for the ASV is shown in Fig 8. In this diagram, y_{ref} is the current state in the sequence generated by the motion planner, u is the control effort choice, and y is the actual state.

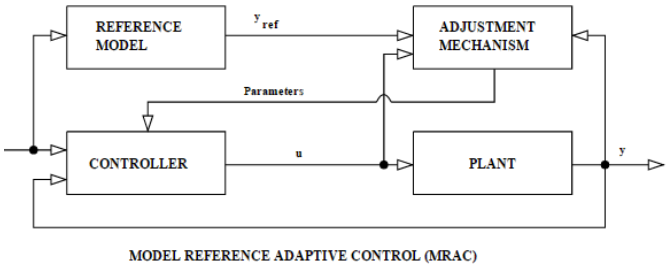

Fig 8. Block diagram of the MRAC controller used on the NaviGator ASV [6]

The team implemented MRAC using the motion planner as the reference generator, and a tracking-error based gradient-descent for the adjustment mechanism. The adjustment mechanism was derived by Lyapunov analysis and resulted in an effective controller of the form shown in (2).

$$u = Ke + Y \int \left(Y^T K_g e\right) dt \quad (2)$$

In this controller, u is the wrench consisting of the force and torque that should be applied to the ASV, K and K_g are the PD and learning rate gains respectively, and e is the state error. Y is the regression matrix for Fossen's marine surface-craft dynamics; it is the state-space model expressed as a linear operation on the unknown parameters. It is interesting to note that traditional PID is an MRAC architecture but with Y being the identity matrix.

MRAC works very well on the ASV, bringing steady-state error to negligible amounts in all cases without introducing oscillations. Additionally, it does not wind-up as much as an ordinary integrator when unexpected disturbances are applied, such as humans pushing the ASV, since it is trying to adapt specifically to drag and inertial effects instead of constant external forces.

Finally, with the controller outputting desired wrenches, the last operation needed is to map that wrench to a thrust command for each thruster. A surface vehicle would only need three thrusters to be holonomic, but with four, the ASV is more fault tolerant. This redundancy in the mapping can be solved as a regularized least-squares problem by evaluating a pseudoinverse [7].

4) Navigation and Odometry: The NaviGator ASV uses a student-developed Sylphase global positioning system (GPS) and inertial navigation system (INS) that is in the process of being commercialized by Forrest Voight, a UF student and member of Team NaviGator AMS. It primarily consists of a circuit board with a Spartan-6 field programmable gate array (FPGA), radio frequency (RF) frontend, inertial measurement unit (IMU), magnetometer, and a barometer. The FPGA performs the correlation operations that enable tracking of GPS satellites. All the sensor measurements and correlations are passed to a computer via USB, into a pipeline of software modules that track and decode the signals from the GPS satellites and then fuse measurements using an extended Kalman filter into an estimate of the ASV's pose in both absolute world and relative odometry coordinate frames. Last, the resulting odometry is transformed so that it describes the ASV's coordinate frame and it is then passed to ROS.

By using the sensors to aid the GPS solution and taking advantage of GPS carrier phase measurements, extremely precise relative odometry is possible, with noise on the order of centimeters over periods of seconds to minutes. This is the result of years of work, during which several iterations of the hardware were produced. The initial version of the hardware was a Beaglebone cape, but quickly moved to the USB/FPGA approach for ease of development and reduced CPU load. The current revision of the hardware is shown in Fig 9.

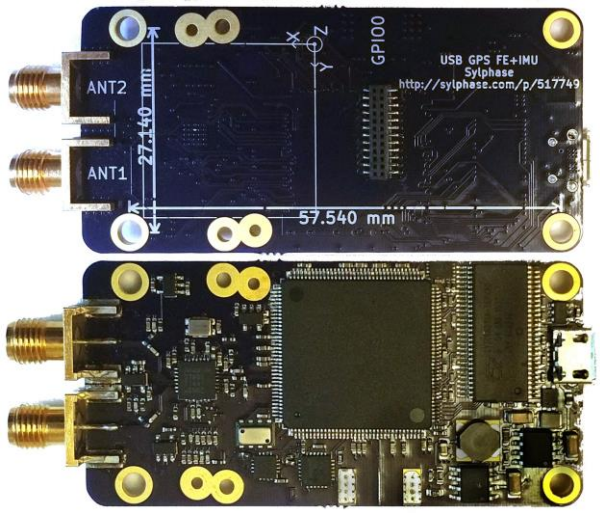

Fig 9. Current hardware revision of the Sylphase, a student-designed GPS/INS

5) Perception: In addition to the underwater camera, the

Navigator ASV is equipped with three additional color cameras, including a stereo pair. The stereo cameras face forward while the third camera, used specifically in conjunction with the racquetball launcher mechanism, faces starboard. Previous testing with cameras oriented horizontally demonstrated that much of the field of view was above the horizon. As a result, the white-balancing done by the camera drivers had a hard time dealing with the variability in lighting conditions. The team found that if the cameras were tilted down 15 degrees, it produced a better color consistency across a spectrum of lighting conditions and a more usable field of view.

The NaviGator ASV's visual system has the capability of generating depth images and point clouds using the stereo cameras. The ASV can generate dense stereo point clouds using a stereo processing ROS package which offers an implementation of the Semi-Global Block Matching (SGBM) algorithm. However, at the resolutions at which the ASV operates, this algorithm consumes a substantial amount of computing resources. Since the ASV has a Velodyne LIDAR that provides excellent point clouds without any on-board processing, it often operates with an inactive stereo point cloud generation pipeline. The ASV also has the ability to generate sparse stereo point clouds using image keypoints with a negligible consumption of resources. This algorithm relies on a modified version of the FAST detector and scans across epipolar lines for potential matches in left and right images [8]. Although it is very efficient, its biggest downfall is that in outdoor environments with vast and repetitive texture regions, the outlier rate is worse than that of the LIDAR.

6) State Machine: The state machine that is used in solving the challenges uses a directed acyclic graph (DAG) to decide which missions to complete at which time. Each mission is first defined by three key attributes: the other missions that it depends on, the objects that it depends on, and whether or not the mission should be re-executed. For example, the Scan the Code challenge does not depend on any other challenges, it depends on the Scan the Code object being recognized after it is executed, it should not be re-executed after it is completed. The state machine is constantly listening for new objects to be found. Once one is found, it goes to the parent missions in the DAG and evaluates if they are ready to be completed. If one of these missions is ready, it is executed. Once it is complete, the DAG is reevaluated for more missions to be complete. This continues until all missions are complete.

III. Design Strategy

This section describes the various strategies that the NaviGator ASV will employ for each of the challenges.

A. Find Totems and Avoid Obstacles

The Find Totems and Avoid Obstacles task requires that the LQR-RRT* controller, the Occupancy Grid Server, and the Classification Server are all working in tandem. The Classification Server continuously runs in the background attempting to classify bounding box instances that are provided by the Occupancy Grid Server. For this challenge, classified instances of totems are provided to the mission as shown in Fig 10. A path that accounts for all obstacles currently visible is planned, with an additional step of circling the closest totem to the ASV. Since the order in which the NaviGator ASV is supposed to circle each totem is provided, the mission attempts to match this knowledge with what has already been classified by the Classification Server. This process is then repeated for each remaining totem visible on the field. Due to the nature of the motion planner, obstacles that come in and out of the frame of view are automatically accounted for in the planned path, even in scenarios where an obstacle may appear while circling one of the totems.

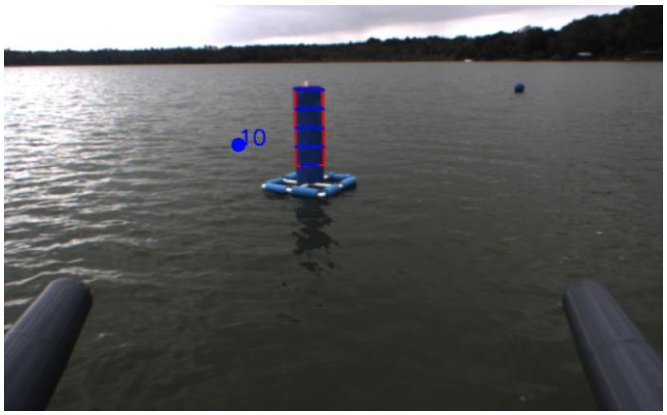

Fig 10. Classification Server detecting a blue totem for the Find Totems and Avoid Obstacles challenge

B. Identify Symbols and Dock

The NaviGator ASV initializes this challenge by first strafing alongside the docking bays, recording the shape and color of the three symbols identified and storing them globally for future use. The computer vision used in this challenge underwent two major trial designs before the current design was implemented. First, a color frame received from one of Navigator ASV's cameras was thresholded into three binary image frames, each with its own hue thresholds intended to create one frame for each of red, green, and blue symbols. After a contour approximation, each contour underwent a series of tests: calculating the number of sides in an approximated polygon to the contour to help detect triangles and cruciforms; the ratio of the contours' perimeter and area which will always be a constant ratio for perfect shapes; and the internal angles between the sides of the contour. Despite the use of the image's hue saturation values (HSV), this strategy did not prove effective given the varying lighting conditions. The next progression, and the one implemented for the competition, uses Canny edge detection on a grayscale image and performs the same geometric tests. Once the shape has been classified, the corresponding pixels in the color image are used to determine the closest hue of red, green, or blue. Another solution considered was one of OpenCV's feature detection algorithms, such as Speeded-Up Robust Features (SURF). Unfortunately, such algorithms did not perform well with the simple features present in the

challenges' symbols.

After the three symbols have been identified, a smooth outline of the docking bays, approximated from the occupancy grid, provides the mission with the location of the center of the three bays. The program then queries the mission planner for the correct symbols in which to dock, then moves to the bay corresponding to the appropriate symbol while relying on obstacle avoidance to not collide with the dock.

C. Scan the Code

The Scan the Code mission begins by first using the Classification Server to classify the light buoy. The NaviGator ASV then maneuvers to the light buoy in a way that places the sun behind the ASV. For example, if it is 11:00 AM, the ASV would drive to the east of the sign in order to get better color readings. A bounding box of the image is then obtained using back-projected LIDAR points. The ASV then runs a Canny edge detector and edge analysis on the image. This edge analysis involves finding vertical lines, and choosing a square of interest to recognize the color based on the positions of these lines. Finally, once the ASV successfully obtains the sequence of colors using this method, they are reported to the team-provided judge's display.

D. Underwater Shape Identification

For completion of the Underwater Shape Identification task, a PointGrey Firefly MV camera in an underwater housing is used as the primary perception sensor. Using a 2.3 millimeter lens, a 0.33 inch CCD sensor, and assuming an average water depth of 4 meters, the camera is able to view an area of approximately 8 x 5 meters under ideal conditions, when observing from the surface of the water. For finding the objects of interest, the LQR-RRT* path planner generates a spiral search pattern that radiates outward from the center of the provided quadrant. The search area is bounded by the dimensions of the quadrant, in this case 40 x 40 meters. While the ASV is executing the search pattern, the vision software is looking for the given shape. The black rectangle is detected by using Hough transforms to find intersecting lines. This in turn provides a bounding area in which to find any of the given shapes. The shapes are classified using Canny edge detection and a host of geometric tests. This whole process is repeated for the second desired quadrant and the results are posted to the team-provided judge's display.

E. Find the Break

Much like the Underwater Shape Identification challenge, the same sensor system is used to count the number of breaks in the search area. Geometric tests coupled with Canny edge detection are used to identify the shapes. For each detected marker, its pose relative to the NaviGator ASV is found by using principal component analysis. This information is tracked for all of the detected markers. The ones that mark the start and end of the sequence and those which have to be counted are denoted. Once this process is completed, the count is reported to the team-provided judge's display.

F. Detect and Deliver

NaviGator ASV begins this challenge by first identifying the target platform using the Classification Server. Once found, it begins a circular search pattern around the platform, maintaining the shape perception camera pointed towards the platform. The search ends when the target with the correct shape and color are identified or the mission time runs out. The same vision algorithm used to classify the shapes and colors of the targets used in the Identify Symbols and Dock challenge is used for this task. At this stage, the LIDAR is used to approximate the normal from the target's plane. The ASV then moves a fixed distance from that plane, orienting itself parallel in order to launch the racquetballs. The Navigator ASV then switches to a station holding behavior, keeping this orientation and distance to the target while it launches each of the four racquetballs in sequence using a simple timing based control of the racquetball launching mechanism. An image of the ASV performing this task can be seen in Fig 11.

Fig 11. The NaviGator ASV performing the Detect and Deliver task

G. Acoustic Pinger-based Transit

The team implemented an acoustic pinger locator using an array of hydrophones and mathematical multilateration techniques. The NaviGator ASV continuously records the sound heard by each of the hydrophones onto a circular buffer. When the amplitude of the sound crosses a threshold, it transfers the contents of the buffer to the computer after a predefined amount of time. Difference in time of arrival (DTOA) measurements are made by taking the time offset where there is a minimum in the running sum of absolute differences between a reference signal and non-reference signals.

From DTOA measurements, the NaviGator ASV is able to calculate the heading to a pinger using the Bancroft algorithm for multilateration [9]. However, the ASV's motors produce frequencies around 20 kHz that drown out the sound of the pinger. Whenever the motors stall, the hydrophone array receives accurate DTOA measurements, providing a heading to the pinger. The equation of a line through the ASV's current position with the heading attained by the team's multilateration algorithm is archived any instant that the motors are stalled and the ASV is within audible range of the pinger. As the ASV drives around the course completing other

missions, it will acquire more estimates of lines on which the pinger lies. Whenever the ASV commands a locate-pinger command, it calculates the least-squares solution to the system of accumulating line equations. This solution is the best estimate of where in the course a specific sound source lies.

Once the heading to the entry gate pinger is located, the NaviGator ASV will align and maneuver through the gate with the active pinger. Throughout the entire mission, the Classification Server will be running concurrently with other software in order to classify the gates and black tower buoy. Once the ASV has successfully entered the correct gate, it will locate and circle the black tower buoy. The ASV will then end this challenge by locating the heading for the pinger associated with the exit gate and then drive through it.

IV. Experimental Results

This section discusses the simulated and physical testing strategies that the team used to prepare for the competition.

A. Simulator

The Gazebo simulator was used to test code that required inputs to change in response to its own output. While basic perception code can be tested on a recorded video, code that performs an action based on the perceived image requires the image to change based on that movement. Gazebo was chosen because it is well supported by ROS and has been used successfully in past MIL projects. Gazebo enables data to be generated based in a virtual environment and then published on the same ROS nodes that it would be published to on the physical hardware. In other words, this makes the process totally transparent to the software and allows the same code to be run in either the real or virtual world, with no modification. This is an extremely valuable tool because it takes a significant amount of time and effort to bring the NaviGator ASV to a lake for field testing, whereas the simulator can be launched in a single command at a workstation.

Due to the time constraints of the competition, the primary objective was to build a virtual sandbox world that contained each challenge in a pre-defined location. A basic virtual environment was created with a sea floor plane and a water surface plane above it. The mechanical systems team created 3D models of each field element that matched the specifications in the challenge preliminary task descriptions document. Textures were applied to the models and they were placed into the virtual environment by defining their locations in a sandbox launch file. This allowed each challenge to be attempted, but the static state of the environment prevented testing edge cases or even slight variations.

The secondary objective was to define the parameters of each challenge, such as the success and failure conditions, and randomly generate each challenge based on them. This allows different positions, rotations, and combinations of shapes and colors for the symbols to be created on the fly. After that, a system that allowed multiple challenges to be generated at a time was constructed. This basically came down to a packing problem wherein the course challenges had size parameters and all of the challenges had to be fit into the maximum allowed area. The challenges also had to have proper connections so that the mission system on NaviGator ASV was able to use information from one challenge to complete the next challenge. This enabled the simulator to generate a possible course based on the preliminary competition rules and verify that the ASV could successfully complete that particular course scenario. A sample screenshot from the simulator can be seen in Fig 12.

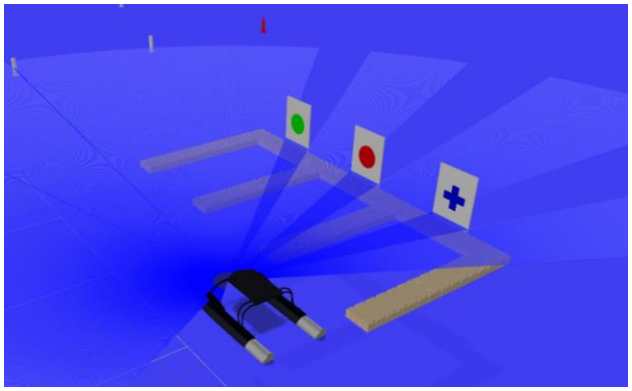

Fig 12. Simulated NaviGator ASV attempting the Identify Symbols and Dock challenge; note the simulated LIDAR beams emitting from the ASV

B. Field Testing

In addition to testing in the simulator, NaviGator ASV underwent significant lake testing. Over 130 hours of in-water testing were carried out in the form of day-long tests in the months leading up to the competition at a lake near UF. Lake testing offered real-life environmental factors that simulation cannot accurately provide, such as wind and current disturbances, various lighting conditions, and inclement weather.

Field testing also offered a chance to test the mechanical systems of the ASV, such as actuators like the racquetball launcher, the strength of team-manufactured components, and the efficiency of the computer cooling system. The frequency and duration of testing helped to expose hardware failures that may have gone unnoticed until the competition. For example, the original sensor mast placed the Ubiquiti omnidirectional Wi-Fi antenna less than two inches away from the Velodyne LIDAR. During field testing, the team found that the LIDAR was returning noisy data. However, when testing in the lab, the LIDAR data looked fine. Eventually the team determined that the only difference was that a wired connection was used to connect to the ASV while working in the lab, as opposed to the Wi-Fi connection that was used while field testing. It turns out that the Wi-Fi signal from the antenna was adding noise to the LIDAR data. Moving the Wi-Fi antenna further from the LIDAR solved the problem. This kind of issue would never have arisen during simulation. The detection of this and other flaws during testing prevented what would have been catastrophic failures during the competition.

C. Field Element Construction

In order to take full advantage of the realistic testing environment that the lake provides, field elements similar to

those that will be used in the competition were constructed. The field elements were designed to be simple in construction and easy to deploy. Many of the elements were made of a PVC pipe frame that allowed for modular construction and easy assembly and disassembly. Buoyancy was provided by foam sheets and pool noodles fitted around the PVC pipes. The simplicity and light weight of the course elements allowed for quick and easy setup and teardown of the course using only a few team members in a kayak. As an example, the Identify Symbols and Dock platform that the team constructed and used for testing can be seen in Fig 13.

Fig 13. The Identify Symbols and Dock platform that was constructed to aid in the field testing of NaviGator ASV

V. Outreach and Sustainability

The labs of Team NaviGator AMS have been providing outreach to a number of Native American communities over the past several years. The lab's efforts were recognized at the 2015 RoboBoat Competition by being awarded the Outreach Award as well as a check for $1000. This money was invested into more outreach activities for Native American communities. In particular, the majority of the money was used to purchase supplies for a science, technology, engineering, art, and mathematics (STEAM) camp for students of the Citizen Potawatomi Nation (CPN) in Shawnee, Oklahoma. Each day of the week-long camp, students learned about a different element of STEAM under the context of an overarching project that involved the evaluation of a historical site that is of importance to the tribe, as shown in Fig 14. The curriculum was developed and administered by the community. The camp framed the different STEAM disciplines as being something that have always been part of the CPN's culture. This was done in order to help the students see the relevance of STEAM in their own lives so that they be more willing to pursue STEAM related careers.

Members of the team have also been mentoring students from local high school FIRST Robotics Competition teams for over five years. They also helped convince the UF engineering college to provide scholarships to all of the seniors on these teams to encourage them to study engineering at UF. These efforts have served two purposes; providing opportunities for the next generation of students to learn about engineering as well as addressing lab sustainability by providing a steady stream of highly-qualified students to work in the robotics labs at UF. This year the team had three incoming freshman work on the NaviGator ASV who had been mentored by students in MIL prior to attending UF. Although they are young, these students have been some of the most productive members of the team. They were put in charge of developing the software and hardware to complete the Detect and Deliver Challenge. They were able to complete this challenge so well, they were also put in charge of the Identify Symbols and Dock Challenge. The lab is excited to have these new students and looks forward to seeing what they can accomplish in the upcoming RoboBoat and RoboSub Competitions and beyond.

Fig 14. Potawatomi students working to evaluate a historical site while learning about STEAM

VI. Conclusion

This paper presents the University of Florida's autonomous surface vehicle, NaviGator ASV, for use in the 2016 Maritime RobotX Challenge. Sacrificing speed for maneuverability, the vessel's four thrusters give the ASV an additional degree of freedom when compared to traditional skid-steer vessels. The novel use of an automotive cargo box for housing electronics created an open layout design that allowed for easy access and rapid repairs. An iterative approach created a strong software foundation that was exhaustively tested with over 130 hours of in-water testing. Team NaviGator AMS is ready for the 2016 Maritime RobotX Challenge due to extensively tested software, simple mechanical design, and robust electronics.

Acknowledgement

Team NaviGator AMS would like to acknowledge everyone who has supported the team throughout the year, including the University of Florida's Herbert Wertheim College of Engineering, the Electrical and Computer Engineering department, the Mechanical and Aerospace Engineering department, as well as the labs of MIL and CIMAR. The team would like to extend an appreciative thank you to their advisers: Dr. Eric Schwartz, Dr. Carl Crane, Dr. Ira Hill, and Shannon Ridgeway.

The latest Team NaviGator AMS developments can be found at www.NaviGatorUF.org.

APPENDIX: SITUATION AWARENESS

The successful integration of any new technology requires that the general public feels comfortable using it. Sometimes companies quell initial skepticism with clever marketing. However, what really allowed society to accept cars, microwaves, washing machines, etc., was the spread of high-level understanding. For example, the average person may not know the intricacies of designing an engine, but the average person does know that the engine burns gasoline to make motion.

Even this kind of extremely basic conceptualization is critical to trust, because users need to have an idea of what to expect with the technology. If someone did not have the slightest idea about what a car is, nothing would stop them from wondering if this mysterious contraption might explode as soon as they got into it. Right now, robots are facing this type of skepticism; to many, they are mysterious contraptions. With accessible high-level overviews, it is possible to get the general public who already know that a car has tires, an engine, and brakes, to know that an ASV has a range sensor, a motion planner, and a state machine. Then something as basic as lights indicating the ASV's decision state would give people confidence in knowing what the robot will do.

However, robots that use machine learning can be unpredictable, even for their programmers. For example, if a robot's motion planner was learned, one can only hope that the situation the robot is in fits well enough with the patterns it extracted during training, such that it behaves as expected. For a robot that utilizes machine learning, there may no longer be a well-defined state machine that can be mapped to indicator lights.

Fortunately, for machine learning and adaptive algorithms, there is almost always a way to quantify confidence. For example, classifier algorithms typically have some measure of how strongly the input matches what the system already understands, say with a discriminant value in supervised learning or a clustering validation index in unsupervised learning. For neural networks, one can use the backpropagation error to generate a measure of confidence. Even for state estimators like the Kalman Filter, one can take the current state covariance as inversely proportional to some measure of confidence. The list goes on and on. The team's idea is to make the robot's confidence in its own decisions available to everyone around it. If the robot has low confidence in a particular scenario, it will effectively display the emotion of confusion, perhaps with a blinking indicator or a sound. If everyone knows at all times how confident the robot is in itself, i.e., a high confidence level in a situation that the robot has seen many times before in training, then everyone will know when they should be relaxed and when they should have their hand on the kill-button.

On the NaviGator ASV, the team implemented a concurrent learning (CL) controller as part of a research project for the University of Florida's Nonlinear Controls and Robotics lab. This controller blends machine learning and control theory by using a novel update law for batch linear regression that allows for simultaneous system identification and Lyapunov-stable control [10]. Unlike most adaptive controllers which are Markovian, this algorithm uses a history stack of states and efforts to enable actual convergence of system parameter estimates. The ASV ran this algorithm under typical conditions without bizarre disturbances to allow the parameters to converge to nominal values. With those values recorded, the team could then compute and view a measure of how off-nominal the current parameter estimates were at any given moment. If, for example, a thruster began malfunctioning, the model would start to diverge from nominal and a software alarm would be raised, alerting the users.

REFERENCES

[1] A. Gray and E. Schwartz, "Anglerfish: An ASV controlled ROV," presented at the 29th Florida Conference on Recent Advances in Robotics (FCRAR), Miami, FL, May 12-13, 2016.

[2] T. I. Fossen, in *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley & Sons, 2011.

[3] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," presented at the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, May 31-June 07, 2014.

[4] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," TR 98-11, Computer Science Dept., Iowa State University. [Online]. Available: http://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf.

[5] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," presented at the 2012 IEEE International Conference on Robotics and Automation (ICRA), St. Paul, MN, May 14-18, 2012.

[6] Wikipedia.org, "Adaptive control." [Online]. Available: https://en.wikipedia.org/wiki/Adaptive_control.

[7] University of California, Berkeley, "Regularized least-squares problem," from *EECS Instructional and Electronics Support*. [Online]. Available: https://inst.eecs.berkeley.edu/~ee127a/book/login/l_ols_rls_def.html.

[8] K. Schauwecker, R. Klette, and A Zell. , "A new feature detector and stereo matching method for accurate high-performance sparse stereo matching," presented at the 2012 IEEE/RSJ International Conference on Robotics and Systems (IROS), Vilamoura-Algarve, Portugal, October 7-11, 2012.

[9] M. Geyer and A. Daskalakis, "Solving passive multilateration equations using Bancroft's algorithm," in *Proc. Digital Avionics Systems Conference*, 1998, pp. F41-1–F41-8.

[10] Z. Bell, A. Parikh, J. Nezvadovitz, and W. Dixon, "Adaptive control of a surface marine craft with parameter identification using integral concurrent learning," to be presented at the 2016 IEEE Conference on Decision and Control (CDC), Las Vegas, NV, December 12-14, 2016.