An Interactive Slocum Glider Flight Simulator

Hans Christian Woithe and Ulrich Kremer
Department of Computer Science, Rutgers University
{hcwoithe, uli}@cs.rutgers.edu

***Abstract*— The Slocum Glider is a commercially available autonomous underwater vehicle used in the sensing of the world's oceans. Its manufacturer-provided simulator uses nearly identical electronics and components as installed in the production glider. The simulator is mainly used to develop and test new hardware and software components in a real-time setting, where for instance, a one-week mission of the glider takes one week to simulate. This limits the types and the complexities of algorithms that can be developed and tested on the vehicle.**

In this paper, we present our ongoing work on a simulation infrastructure used to fly and operate a Slocum Glider in a virtual ocean environment. We also introduce a new three dimensional companion visualization tool. The effectiveness of the infrastructure and its new visualization tool is illustrated in the context of three applications, namely glider education and training, the replay and analysis of glider missions, and the design and testing of algorithms to be used in future missions.

I. Introduction

Autonomous underwater vehicles (AUVs) have revolutionized the sensing of the world's oceans. The Slocum Electric Glider is such an AUV and belongs to class of vehicles which propel themselves using a buoyancy driven mechanism [1], [2], [3] instead of a propeller [4], [5], [6], [7]. The vehicle is a commercial product manufactured and sold by Teledyne Webb Research (TWR) [1].

The buoyancy driven flight is accomplished by the movement of a piston at the front of the vehicle and is referred to as the buoyancy pump. When the pump is fully retracted, the vehicle's displacement of water decreases and causes the vehicle to dive towards the ocean floor. Likewise, when the piston is fully extended, the glider's displacement increases and it is forced to rise towards the surface. The movement of an internal battery pack changes the vehicle's center of gravity and allows the fine tuning of the pitch angle of the glider. Along with wings and a controllable fin, the AUV is able to navigate through the ocean in a sawtooth flight profile at approximately 35 cm/s [8].

The manufacturer, along with the production of the vehicle, also produces a simulator for the Slocum Glider. The simulator contains the essential electronics needed to develop and integrate new software and hardware for the platform. These electronics are in a small container and it is affectionately called the Shoebox simulator. Because the Shoebox's hardware closely matches that of the vehicle, little or no additional work needs to be performed to bring the new components into production. This however is also a point of strict limitation. The glider's 16 MHz processor already struggles at times to run the control software, so advanced simulations involving, for example, a trace of ocean currents is not possible. This brings forward the issue of how well tested an algorithm can be if its testing framework is limited.

In previous work [9], we introduced our Slocum Glider simulation infrastructure. Unlike the Shoebox, it is more flexible and allows for direct specification of the environment that the vehicle should perform in. The simulator also runs faster than real time, allowing for a faster and more streamlined development environment. For example, a one-week mission can be simulated on commodity hardware within minutes or seconds depending on the detail of the simulation. During our work on the simulation environment, we recognized the need for visualizing the behavior and performance of a vehicle during its mission. To address this need, we have developed a companion graphical interface to our simulation environment. In this paper, we will (1) provide the background to our simulation infrastructure, (2) introduce a complimentary three dimensional visualization tool, and (3) illustrate their effectiveness through real world applications.

II. Simulator Background

The simulation framework for our new Slocum Glider is developed using the Python programming language. Python provides for rapid application development and has a vast set of libraries available to it. Such libraries include NumPy, SciPy and Matplotlib which are essential for the development of algorithms within the simulation environment and provide an open source and free alternative to Matlab.

As described in our previous work [9], the simulator is capable of using multiple speed models for the Slocum Glider. They include a speed model similar to the one present in the TWR's Shoebox simulator and a speed

distribution model we have empirically derived from years of flight time off of the coast of New Jersey. Experimental results in [9] showed that sampling the vehicle speed using this distribution can produce simulated missions similar in length (in time) to their real counterparts when supplied with the same waypoint list.

The framework also makes use of bathymetry data from the National Geophysical Data Center's (NGDC) ETOPO1 model[10]. As with the real glider, this allows the simulated vehicle to be programmed to inflect a specified amount of distance off of the ocean floor. This is accomplished by an artificial altimeter which interpolates the bathymetric dataset using inverse distance weighting (IDW). The rate at which the dataset is sampled can be adjusted so it must not be performed at every iteration of the control loop. This allows the tradeoff of decreasing the time necessary to perform a mission while losing some accuracy or vice versa.

Integration of Coastal Ocean Radar (CODAR) [11] data from Rutgers has also been incorporated into the simulation environment. Data products are available in hourly increments and contain a grid of sea surface current vectors. Like the bathymetry model, the currents applied to the vehicle are interpolated using IDW. However, unlike the bathymetric dataset, new CODAR files must be loaded as the simulated vehicle progresses through time. This method has been used successfully in [9] to emulate a previous sea trial.

Products generated from the Regional Ocean Modeling System (ROMS) [12] and the Hybrid Coordinate Ocean Model (HYCOM) are also being developed into the framework's world model. These systems provide nowcasts and forecasts of the ocean environment. The slow speed of the Slocum glider makes it much more sensitive to ocean currents. For this reason we are particularly interested in the current predictions of these two technologies. Both models can estimate currents at multiple depths which will allow us to more accurately simulate the vehicle's flight. This is useful in the initial mission planning stages of a deployment, allowing one to accurately judge when and where to deploy. It may also assist in the selection of new target waypoints.

Planning an efficient path through a field of currents may be vital to the success of missions. However, making efficient use of the limited energy resources that are available in the vehicle is an equal challenge. In [9] we describe an infrastructure we have created to measure the power consumption of a subset of the Slocum Glider's components. These devices include main and emergency power, the fin and pitch motors, and the buoyancy engines pump and brake. This platform was successfully deployed on two sea trials. The collected data was analyzed and used to build several energy profile models of these devices as part of the simulation environment. For example, we have observed that the energy cost to perform an inflection increases linearly with depth. This is due to the increase in water pressure that the buoyancy pump must work against as the vehicle is exposed to deeper depths. As we increase the number of energy profiles we measure and integrate the total energy costs calculated by simulated missions becomes more accurate.

Providing an idea of the necessary cost required to perform a deployment may be helpful for mission planning engineers. A simulated deployment will likely always vary from an actual deployment because of the highly dynamic nature of the physical environment and the accuracy and resolution of forecasts. However, the framework should not only be used before the mission, but also actively during deployments. If the vehicle is not equipped with our measurement infrastructure, the energy used up to that point of the deployment could be estimated through log files sent by the glider. The remainder of the mission, which has not yet been flown, could then be simulated. Feedback of the energy used can then be given to the planning engineer to decide whether to maintain or alter the programmed flight scenarios.

In its current form, the simulator can only be programmed using the Python programming language. Generally, an object of class `Glider` is instantiated and loaded with the speed models `SimDist` and `SimShoebox`, described previously in [9]. The bathymetry and currents, and their effect on the vehicle, are then optionally loaded. Additionally, waypoint bearings and depth limitations can be set. When the initialization of the environment and the vehicle is finalized, an iteration of the vehicle control loop is called. This is followed by an update to the environment and finally, any other algorithms may run before the process is repeated by a call to the control loop.

Other methods to program the simulation framework are in progress. A parser for mission files used by the Slocum Glider has been created. The infrastructure needed to run these missions is however not yet complete as it has required the development of a control system more similar in nature to that of the glider's. A parser and compiler for our domain specific language is also ongoing work [13].

The simulation framework is a versatile tool which allows for the specification and behavior the vehicle and its environment. Its can lend itself useful in a variety of

(a) (b) (c)

Fig. 1. A heightmap (a), generated from NGDC's ETOPO1 model, is used to create a texture (b) for the 3D object (c) of the terrain in SimGUI.

applications, some of which are describe in section IV. For further insight into inner workings of the simulator we refer the reader to [9].

III. Graphical Interface

Missions performed in the simulator can generate output about the vehicle as well as the environment. Typical focal points are on the glider's vertical profile (depth, pitch, roll) and geographic position. Much of these data can be graphed using tools such as matplotlib, Matlab, and the Generic Mapping Tools. We have nonetheless found the need to be able to visualize the Slocum Glider in three dimensional (3D) space for both simulated and real sea trials.

Google Earth is a powerful 3D tool that allows for the display of data on a world model [14]. It is used extensively in the deployment process, from the initial planning stages to tracking the vehicle's flight. Weather forecast images are often overlaid to aid in the selection of new waypoints. The simulator has also been retrofitted to create KML files. Despite its usefulness, it also has its shortcomings. First, under the current licensing scheme, any movies, KML/KMZ, or images from Google Earth cannot be used in any reports or presentations outside of our organization without the purchase of a license of Google Earth Professional. Secondly, we would like to be able to easily customize the 3D environment to allow for live interaction as well integration with our simulation environment.

To fulfill this gap, we have implemented an interactive graphical environment for the Slocum Glider that is able to mesh with the existing framework. In the simulation front, it can be utilized in the development and debugging of algorithms, for example in glider swarming and coordination. With regard to actual deployments, it can be applied to the whole deployment process from waypoint selection, like Google Earth, to a three dimensional inspection of a vehicle's flight, such as that RU27's when it suffered from biofouling [15]. We will refer to this graphical user interface as "SimGUI" although it is completely independent from the simulator.

The SimGUI makes use of Panda3D, an open sourced, BSD licensed, cross-platform 3D game engine with origins from Disney. It is a mature and stable engine used in commercial products. The core of the engine is written in C++, for efficiency, and its primary programming interface is exposed through Python. This allows a single language to be used throughout the entire simulation infrastructure. Although other game engines such as Ogre [16] provide Python bindings, they are not the focal point of the engine and so were not chosen.

The terrain information in SimGUI, as in the simulator, is based off a of dataset by NGDC. A heightmap, an image representing the height of the terrain, must first be generated. This is accomplished by interpolating the height of a physical location and mapping it to a corresponding pixel in the image. A heightmap generated using this process of the coast of New Jersey is shown in Fig. 1(a). The presented heightmap has been limited to maximum altitude of 25 meters and maximum depth of 125 meters. These limits were chosen to give enough detail about the immediate landscape for orientation, and to provide enough depth for our simulations of a 100 meter glider. Different parameters can be provided to the heightmap generation script, but details about the terrain may need to be sacrificed because most heightmaps are bound to eight bits of resolution.

Once the heightmap of the area of interest has been generated, a texture must be generated for the terrain. A set of images, such as sand, rocks and grass are repeated, or tiled until they have the dimensions of the desired image texture. Each of the tiled images are then blended

together in a process called texture splatting. In this process, a pixel's transparency, or alpha, is calculated for each image depending on its altitude. For example, a pixel representing a deep ocean depth should have little or no contribution of the grass texture applied to it. Fig. 1(b) shows the result of the texture splatting process performed using the NGDC dataset.

The heightmap is also used to generate a 3D object of the terrain. This is achieved in Panda3D using the `GeoMipTerrain` class. The terrain texture can then be applied to the 3D object. The resulting product, with lighting, is shown in Fig. 1(c). Water, as well as a sky box surrounding the environment have been created and can be seen by the reflection of clouds in the lower right corner of Fig. 1(c).

Depending on the graphical capabilities of the hosting machine, the level of detail (LOD) in SimGUI can be adjusted. The LOD of the terrain could be set to use a high number of polygons to shape the nearby scenery, while progressively decreasing the polygon count into the horizon as the distance from the engine's camera increases. Smaller textures for the terrain and the reduction of non-essential models can also aid in the programs performance but comes at the cost of the user's overall experience. If desired, SimGUI can optionally be loaded without a terrain object. This is certainly useful in non-coastal areas while simulating a shallow water glider because it is unlikely that the vehicle will ever reach the ocean floor.

As previously stated, SimGUI is independent of the simulation infrastructure and intentionally so. This allows the simulation and the graphical environment to be developed separately and leaves open the possibilities for simulators from other vehicles to be integrated into SimGUI. Three dimensional objects must simply be imported and their respective simulators must provide the necessary information to SimGUI to place the vehicle in the scene.

The simulator and SimGUI typically run as separate operating system processes, but can be run as one. Running as one process gives the benefit of simplicity in that the memory is shared between the two components of the infrastructure. However, this can also lead to performance and engine frame rate issues because the process will be busy computing the simulation instead of refreshing the scene. This can be particularly be the case if multiple vehicles are in the environment concurrently.

When running as separate processes, the components can exchange information through pipes, shared memory or other forms of interprocess communication. Using UDP/TCP it is also possible for the simulation node to be on a completely different node than that of SimGUI. A cluster of computers can also run vehicle simulations while communicating to a central coordinating node. The coordinator ensures that other nodes have synchronized the global environment and updates the display state of SimGUI to provide feedback of the distributed simulation.

Great care has been taken to keep SimGUI, like the simulator, flexible enough for a multitude of applications. A selection of the employments of the simulator and SimGUI is described in the follow section. These applications should provide an indication of the applicability of the infrastructure in context with the Slocum Glider.

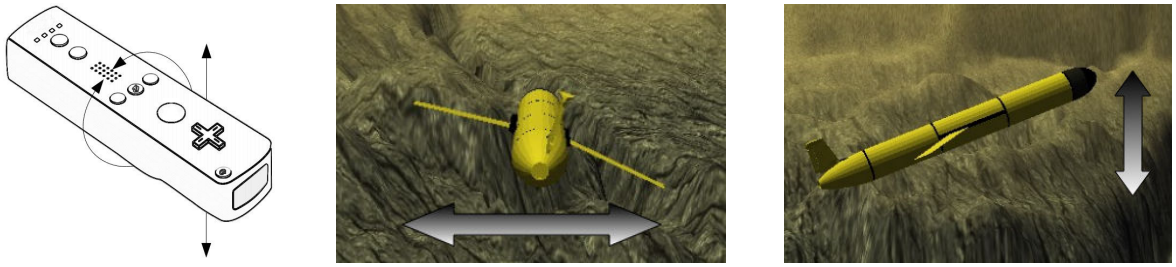
IV. Applications

The initial purpose of the simulation environment was to make use of energy usage models we have created from collected samples of sea trials equipped with our measurement infrastructure. Using theses models, basic flight segments were to be analyzed for their energy costs. The need for additional functionality was quickly realized if more complex virtual deployments were to be executed. Thus, the framework was extended to include bathymetry, currents, and a companion graphical interface. This multipurpose platform has found itself useful outside the realm of mission planning. This section describes how the simulator and SimGUI have been successfully used in eduction, the replay and analysis of missions, and in the development and testing of algorithms.

A. Education

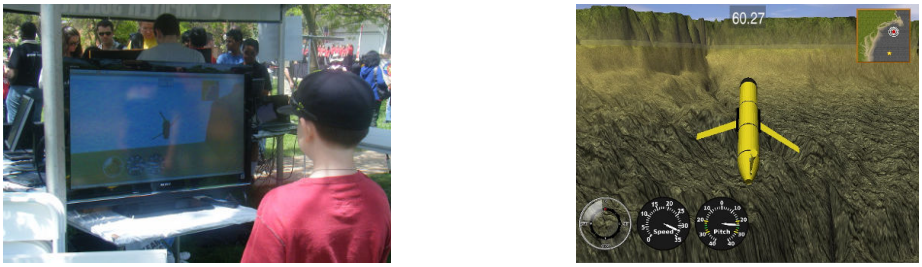
The simulator and SimGUI can coincide to produce a feature rich and flexible environment. It can augmented to aid in the education of those curious to learn more about the operations of underwater vehicles. To demonstrate the general dynamics of the Slocum Glider's flight, the infrastructure was modified to make use of the Nintendo Wii Remote controller. In addition to its buttons, the controller also contains motion sensors allowing an additional level of interaction with the console's games. The remote's popularity and familiarity with children and adults, along with a large support community, makes the Wii Remote an ideal tool to be integrated into the simulation framework for education. Use of the Wii Remote to control the glider allows the user to more fully understand the saw-tooth flight profile utilized by the vehicle. For the more experienced, the simulator and SimGUI platform can be used in the training of new mission planning engineers.

The remote communicates with the computer hosting our infrastructure using the wireless Bluetooth technol-



(a) (b) (c)

Fig. 2. The Wii Remote has been integrated with SimGUI to control a virtual Slocum Glider. The roll of the remote (a) translates to a roll on the vehicle (b). Likewise, the glider's pitch (c) is defined by the pitch of the controller (a).



(a) (b)

Fig. 3. SimGUI, retrofitted to work with the Wii Remote, was put on display at Rutgers Day 2010. Students and families were able to learn the basic principals of flying a Slocum Glider through an interactive mission (a). Feedback about the glider is provided through heading, speed and pitch gauges as well as a minimap showing the location of the vehicle (b).

ogy. For a workstation not equipped with Bluetooth, USB Bluetooth adapters are readily available. The CWiid library [17] and its python wrappers were used to develop a program to acquire data from the controller for our simulation infrastructure.

The acquisition program periodically polls the library for updates of the Wii Remote; any pending messages are returned. Before any data are used, the remote must first be calibrated. This is accomplished by calculating an average offset by collecting several hundred messages while the remote is untouched on a flat surface. This offset is then used to properly calculate readings when the remote is in active use. Because of the controller's high fidelity and update rate, tens of samples are also averaged together to compose a single aggregated reading. This would otherwise induce an undesirable oscillating effect on the vehicle since fluctuations of the hand are captured by the remote. Readings for Wii controller's pitch, roll and home button are serialized and sent to a simplified simulation component via UDP.

The simulator, upon receiving updated pitch and roll information, performs an iteration of its control loop to update the vehicle's state and position in the environment. The control loop's speed model is a modified version of `SimShoebox` [9]. The AUV's speed over the ocean floor is calculated based on the pitch angle of the remote. With the pitch and speed known, the glider's change in depth can be calculated using trigonometry. The roll component of the update is mapped to the rotation of the glider's battery pack which corresponds to a roll in the vehicle. The glider's progress made in the environment is then computed. Finally, the necessary information of the simulated AUV's newly determined state is given to SimGUI.

The SimGUI interface updates the graphical representation of the virtual glider. The vehicle's roll and pitch will match that of the Wii Remote as shown in Fig. 2. Although the Slocum Glider no longer uses the roll of the battery pack as a steering mechanism, steering the vehicle through the rolling of the Wii Remote is more intuitive to an inexperienced user who is unfamiliar with gliders. We may, in the future, opt to instead control the fin by detecting the yaw angle of the remote.

The system, as described, allows users to fly the glider in a depiction of a real environment. This educational tool was put on display, along with the actual vehicle, at the Computer Science booth at Rutgers Day 2010. Because of the public's familiarity with torpedoes, it is

often assumed that Slocum Glider is propeller driven and that the propeller was removed when put on display. After an explanation of the nature of the buoyancy engine, we have found it useful, especially for children, to let them take Wii Remote and experience the vehicle's flight dynamics themselves as shown in Fig. 3.

The platform on display took users through a interactive mission. The objective of the deployment was to fly to a list of waypoints, from a previous deployment, marked by floating buoys. The mission, starting near Marthas Vineyard in Massachusetts, brought users near the shore and the edge of the continental shelf, then towards Tuckerton, New Jersey for the final recovery. A miniature map on the screen displays the environment, the location of the vehicle and the next target waypoint. This is useful for navigation as the glider is displayed in a third person perspective, where the camera is behind and slightly above the vehicle. SimGUI was also equipped with heading, speed and pitch gauges as shown in Fig. 3(b). The heading gauge aids in the navigation, while the pitch and speed gauges can provide the user with information about how to accomplish an optimal flight profile. Users learn that long and deep saw tooth glides at 26° make the most progress towards the waypoint. The speed of the simulated vehicle was increased so that the mission could be accomplished in a few minutes. Students, children, and adults were all able to learn the basics of glider flight using this tool. An especially enthusiastic kindergartner was curious to know when it would be available for their Wii console.

After the basic dynamics of a glider's flight are understood, the infrastructure can also be used to teach flight planning procedures. First, as is typical in deployments, a suitable day must be chosen based on weather conditions. Once the virtual date has been determined, the trainer must specify the dataset to be used by the simulation infrastructure. As in [9] this can include, but is not limited to, coastal radar and bathymetry.

Next, when the vehicle is placed at its point of departure, the trainee follows protocol and performs a sequence of short test flight segments. Given the green light, the glider's target waypoint can be given along with the parameters specifying when its next surfacing should occur. The dive segment is then simulated, with the vehicle surfacing at approximately the time specified. The AUV's position with an overlay of the most recent ocean conditions, such as CODAR, can be displayed in SimGUI. After analyzing the situation, the trainee may opt to change mission parameters or have the simulation continue as formerly instructed.

The energy models that are part of the simulation framework provide the trainee with an assessment of the amount of energy used during the virtual deployment. The mission could then be executed again by the trainee with a different set of waypoints based on the intuitions gathered from the first run. Results of the simulations could then be compared and scored. The trainer, with their additional experience, could conceivably also show the trainee a more suitable and energy efficient alternative flight path that still satisfies the objectives of the mission.

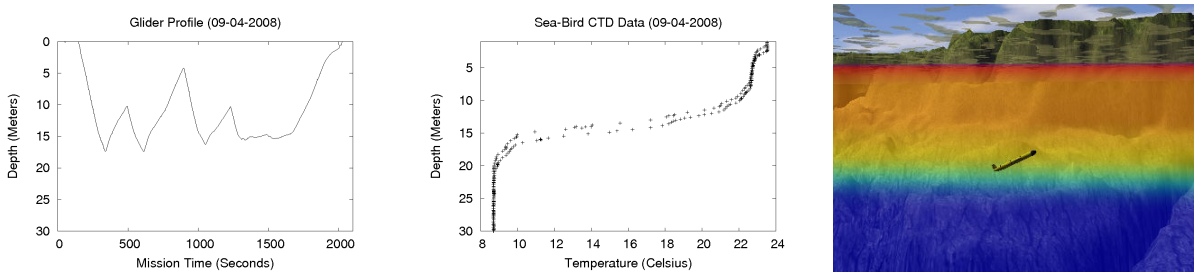
The simulator and SimGUI lend themselves useful in the education of those yearning to learn more about the Slocum Glider. The Wii Remote controlled virtual glider enlightens users of the basic flight dynamics of the vehicle. The platform can also contributes to the training of those ready to advance to becoming flight engineers.

B. Replay And Analysis

The presented infrastructure is not limited to being an educational tool. Experienced users can also use the infrastructure to replay and analyze past and current deployments. The plotting of data is indispensable, however it can at times be difficult to correlate the vast amount data at once. We have used the infrastructure to visualize the tracking of a thermocline and begun the process of replaying the transatlantic mission of RU27's where its flight suffered from biofouling.

A thermocline is a layer of water where the temperatures changes drastically in the water column, typically within several meters, and is often associated with phytoplankton. Phytoplankton is responsible for much of the oxygen produced in the Earth's atmosphere. Scientific instruments to gather information about phytoplankton are available for the Slocum Glider. To improve the measurement of the plant life, it may be useful for the vehicle to gather readings only near points of interest, such as that of thermoclines. In [13] we describe the tracking of a thermocline using a Slocum Glider.

The tracking algorithm used in the sea trials was purposely simplistic. The glider used to perform the algorithm was equipped with a Linux single board computer (SBC) with new vehicle control software [13]. The sea trials were thus short with a buoy attached to the AUV in case of failure. Because the deployments were so brief, the algorithm made use of only the most recent temperature and depth readings. This was to ensure that readings like the initial dive profile do not perturb the vehicle's view of the water column. The temperature measurements collected from the initial dive are sometimes discarded because the sensor is affected by being at the surface; the sun, for example, can heat the sensor causing warmer than actual temperatures to



(a) (b) (c)

Fig. 4. A flight profile of a Slocum Glider tracking a thermocline (a) in a water column as measured in (b). The water column and the flight of the vehicle can be displayed for analysis in SimGUI (c).

be reported for several meters. As the vehicle ascends or descends, the algorithm searches recent readings to detect if a temperature threshold has been met over a certain depth. The temperature and depth thresholds are predefined parameters and are specified *a priori*. When triggered, the SBC instructs the vehicle to fly in between the depths that caused the trigger. Other thermocline tracking algorithms, [18], [19], have since been developed that may make use of but do not necessarily require these temperature and depth parameters to be specified.

The resulting flight profile of the glider tracking a thermocline using said algorithm is shown in Fig. 4(a). The temperature of the water column during the sea trial is presented in Fig. 4(b) and was captured using a Sea-Bird profiling sensor. The thermocline is present at approximately 10-18 meters where the observed temperatures change dramatically. The AUV successfully detected the thermocline and flew the depth range that triggered the specified parameters. At approximately 700 mission seconds, the glider did not observe a dramatic temperature change in recent readings and thus returned to its previously instructed minimum depth. This was not caused by a fault in the thermocline detection algorithm and could have been prevented if we made more of the flight data available to it. As stated, however, the sea trials were short and we wanted to avoid the initial dive readings which could possibly corrupt the whole dive segment. After this mishap, the thermocline is again found and tracked.

This thermocline tracking mission has been imported into SimGUI for visualization. The data captured by the Sea-Bird profiling sensor was interpolated to produce a temperature water column texture. The texture was then applied to a 3D plane and position in the environment. Like Google Earth, overlays in SimGUI can be applied to the ocean surface. It is also possible for overlays to be placed in the water column as is shown in Fig. 4(c).

Before a mission can be replayed in the environment, the vehicle's data files must be readied. The binary log files are converted into text and filtered for the necessary data fields. The data are then used by SimGUI to visualize the virtual vehicle. Fig. 4(c) depicts the flight of the thermocline tracking mission. The glider flies the profile of Fig. 4(a) with the water column temperature overlay of Fig. 4(b). The vehicle can be seen hovering at approximately 15 meters towards the end of the segment. This was likely due to the vehicle's density in the water or being tethered to a buoy. This phenomena is not unusual and can be readily observed in other deployments.

The replay feature may also be instrumental in determining why and how a glider's flight went astray. In the case of RU27, when crossing the Atlantic from New Jersey to Spain, it experienced biofouling primarily of gooseneck barnacles. The biofouling caused the vehicle to spin, especially when climbing. The SimGUI infrastructure could have been used to view the spinning in 3D space to help ascertain the cause. RU27's flight is in the process of being imported into SimGUI so that a visual record of the effect of its biofouling exist. In the case of future missions, we hope to use the platform to analyze these types of scenarios and to compare them against those previously observed.

C. Algorithm Development & Testing

The simulation framework is especially pertinent to the development and testing of algorithms for the Slocum Glider. The thermocline tracking algorithm described earlier was not developed using this development environment. If the infrastructure had been in place, the implementation of the algorithm would have been more robust because the simulator grants a more realistic scenario for rigorous testing.

The dataset used to test the thermocline tracking algorithm in [13] was a synthetic thermocline loosely

(a) (b)

Fig. 5. A triangle swarming formation was implemented in the simulation environment. Using SimGUI, the formation can be viewed from the top (a) as well as at an angle (b). The vehicles maintain the formation throughout the entire virtual deployment.

based on a small subset of observed readings from a real deployment. To emulate the vertical movement of the water column, the temperature data was shifted to different depths. This however is not representative of what would be observed in real ocean conditions.

With the simulation environment, previous temperature profiles captured from past missions can now be loaded. The algorithm's input is then identical to what is observed in nature. The dataset is also not limited to merely one deployment. The algorithm's depth and temperature threshold parameters can even be tested and made to match what has been witnessed during the same dates of the deployment in previous years. With the integration of HYCOM and ROMS, forecast data could also be used to determine if the parameters are fit for the mission.

The simulator on its own is already a constructive tool in the development of novel algorithms. When combined with SimGUI however, the two components together truly shine. In one example, this platform has been used to develop, debug, and visualize the coordination of multiple gliders.

The number of sensor packages available for the Slocum Glider has steadily increased over years. Unfortunately, the vehicle is limited to a small amount of sensors it can carry aboard during a mission. This has created interest in the dispersal of multiple AUVs during a deployment. The vehicles themselves, and the sensors they carry may be heterogeneous. This approach allows scientists to study an area of interest in a broader context.

Using the infrastructure we have developed a triangle swarming algorithm. The triangle is formed by three gliders, with one lead vehicle and two followers. Each vehicle in the triad can carry different sensor packages and conjointly sample an area at the same time. The lead glider periodically informs the followers of its current location and heading using acoustic communication, such as the Woods Hole Oceanographic Institution's (WHOI) Micro-Modem [20]. The followers calculate their next target waypoint relative to the information received by the lead glider. Their position in the triangle is specified *a priori* although it may be changed if desired.

With this scheme in place only the waypoint for the lead glider must be set. The remaining gliders follow the lead vehicle as long as they can regularly receive updates. During simulations, this update occurred once every ten minutes. This refresh rate was enough for the AUVs to maintain formation. To ensure that the burden of the energy costs does not solely fall on one glider, we envision that the responsibility of the lead glider rotates among the group.

The results of testing the algorithm in the simulator and displaying them in SimGUI are shown in Fig. 5. The gliders are all deployed at the same location and can be seen at first dispersing away from one another towards their position in the triangle. Once the triangle is formed, the follower AUVs turn and proceed the execution of the formation.

Although an engineer effort is still required to port algorithms from the simulation environment to the actual vehicle, the framework makes up for this shortcoming by allowing the algorithm to be extensively tested before deployment. We foresee that the effort and cost expended in lab testing would be less than that of a trial and error approach through sea trials. This development environment, while under continual development, will prove itself as an indispensable tool as we work with the Slocum Glider.

V. Related Works

Google Earth [14] is an excellent and general all purpose utility applicable throughout the deployment process. However, because of its license and our current need for adaptability, we have chosen to implement our own graphical interface. Neptus [21], [22] and AUV workbench [23] most closely resembles our work. They, like in our framework, present a visual interface useful in the planning and playback of missions. It is nonetheless unclear if these systems were adaptable enough for our requirements and for applications like that of the Wii Remote controlled Slocum Glider.

VI. Future Work

Future work is dedicated toward increasing the number of devices that the power measurement infrastructure monitors. This will be accomplished with the next revision of the measurement board currently in development. The new design includes a master with several slave boards; each slave is created to sample only a specific subset of glider components. The master's responsibility is to instruct the other board when and at what rate to sample sensors. The sampled readings are then sent through the data link to the master for logging. To reduce its energy foot print, the master also has the authority to put slave boards into a low power mode when their services are not required.

This modular design, albeit more complex than the previous infrastructure [9] adds flexibility. Once the master board is completed and stabilized, it will likely not require many changes. This separation between the sampling and logging enables slaves to be interchanged in the vehicle while leaving the master in place. Thus, as new scientific sensors are added to the glider, slave boards can be created to monitor their power profile. Depending on the needed configuration, the slaves may also be daisy chained together. The previous design also relied on the glider's science computer to perform the data logging. A major pitfall of this approach was not being able to record sensor data during surfacings because the vehicle disables the science bay. The new infrastructure will surpass the shortcomings we have experienced with our first measurement board and can be customized more easily to match a mission's specifications.

While the second board is still being developed, we have decided to continue to make use of the existing architecture to collect power measurements of the vehicle's devices. During its last deployment, all of the eight initial sensors were employed with the science computer's clock speed increased to handle the data logging. This produced an overhead of 1.28 watts to the stock glider [9]. A modified board using only four sensors and a science computer with a lower clock speed of 7.3 MHz (still higher than the default), decreased the overhead to 0.71 watts, or by 45%. This board, measuring the vehicle's main power, brake, and buoyancy pump, is being readied for a deployment in early August of 2010. With the additional power savings, the glider will fly from the coast of southern New Jersey to the end of the continental shelf to collect readings for the buoyancy engine at depths up to 100 meters. The collected data will be used to extend our energy models for our simulation framework [9].

To further improve the simulation framework we will continue our work on the integration of ROMS and HYCOM data. This is useful in the replay of past, ongoing, and future missions. As previously stated, the glider is extremely sensitive to ocean currents. We hope that by applying current estimations to a virtual vehicle we can support flight engineers in the selection of new target waypoints. Also, during the training of new engineers the infrastructure can give an inclination of the effects of currents or tides without endangering a real glider. Finally, the infrastructure is planned to be used in a classroom setting. We hope that this framework will be useful for students learning about the glider and prove to be an excellent instrument for teaching.

VII. Conclusion

We have described our continuing work on the simulation infrastructure and introduced its companion graphical user interface, SimGUI. When used in conjunction, an excellent and useful tool in education, replay and analysis, and algorithm development and testing is produced. Its use as an educational medium is evident when the Wii Remote is picked up by a newcomer who quickly learns to fly the vehicle. It is pertinent to the replay and analysis of missions to aid in the diagnosis and visualization of previously flown missions. Finally, it is suitable in the development and testing of novel algorithms as shown by our implementation of a Slocum Glider swarming algorithm.

VIII. Acknowledgments

We would like to thank the department of Marine and Coastal Science and WINLAB at Rutgers University for their support and help. This research has been partially funded by NSF awards CSR-CSI #0720836 and MRI #0821607.

REFERENCES

[1] Teledyne Webb Research, "Slocum glider," (Falmouth, MA). http//www.webbresearch.com/slocum.htm.

[2] J. Sherman, R. E. Davis, W. Owens, and J. Valdes, "The autonomous underwater glider s̈pray," in *IEEE Journal of Oceanic Engineering*, vol. 26, October 2001.

[3] C. C. Eriksen, T. J. Osse, R. D. Light, T. Wen, T. W. Lehman, P. L. Sabin, J. W. Ballard, and A. M. Chiodi, "Seaglider: A long-range autonomous underwater vehicle for oceanographic research," in *IEEE Journal of Oceanic Engineering*, vol. 26, October 2001.

[4] Hydroid, LLC., "Remus AUV," http//www.hydroidnc.com/.

[5] C. Kunz, C. Murphy, R. Camilli, H. Singh, J. Bailey, R. Eustice, M. Jakuba, K. ichi Nakamura, C. Roman, T. Sato, R. A. Sohn, and C. Willis, "Deep sea underwater robotic exploration in the ice-covered artic ocean with auvs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2008.

[6] B. Schulz, R. Hughes, E. Matson, R. Moody, and B. Hobson, "The development of a free-swimming uuv for mine neutralization," in *Proceedings of MTS/IEEE OCEANS 2005*, vol. 2, September 2005.

[7] B. Schulz, R. Hughes, E. Matson, R. Moody, and B. Hobson, "The theseus autonomous underwater vehicle. a canadian success story," in *Proceedings of MTS/IEEE OCEANS Ý7*, vol. 2, October 1997.

[8] J. G. Graver, R. Bachmayer, N. E. Leonard, and D. M. Fratantoni, "Underwater glider model parameter identification," in *Proceedings 13th International Symposium on Unmanned Untethered Submersible Technology*, 2003.

[9] H. Woithe and U. Kremer, "Slocum glider energy measurement and simulation infrastructure," in *IEEE Oceans 2010 Conference*, (Syndey, Australia), May 2010.

[10] National Oceanic and Atmospheric Administration, "National geophysical data center," http://www.ngdc.noaa.gov/.

[11] Rutgers University, Coastal Ocean Observation Lab, "CODAR," http://marine.rutgers.edu/cool/codar.html.

[12] A. Shchepetkin and J. McWilliams, "The regional oceanic modeling system (roms): a split-explicit, free-surface, topography-following-coordinate oceanic model," in *Ocean Modelling*, vol. 9, pp. 347–404, 2005.

[13] H. Woithe and U. Kremer, "A programming architecture for smart autonomous underwater vehicles," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 4433–4438, Oct. 2009.

[14] Google Inc., "Google Earth," http://earth.google.com/.

[15] Rutgers University, "The scarlet knight's trans-atlantic challenge," http://rucool.marine.rutgers.edu/atlantic/.

[16] "OGRE," http://www.ogre3d.org/.

[17] "CWiid," http://abstrakraft.org/cwiid/.

[18] N. Cruz and A. Matos, "Reactive auv motion for thermocline tracking," in *IEEE Oceans 2010 Conference*, (Syndey, Australia), May 2010.

[19] S. Petillo, A. Balasuriya, and H. Schmidt, "Autonomous adaptive environmental assessment and feature tracking via autonomous underwater vehicles," in *IEEE Oceans 2010 Conference*, (Syndey, Australia), May 2010.

[20] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The WHOI micro-modem: An acoustic communications and navigation system for multiple platforms," in *IEEE Oceans Conference*, (Washington, D.C.), 2005.

[21] P. Dias, S. Fraga, R. Gomes, G. Goncalves, F. Pereira, J. Pinto, and J. Sousa, "Neptus - a framework to support multiple vehicle operation," in *Oceans 2005 - Europe*, vol. 2, pp. 963 – 968 Vol. 2, 20-23 2005.

[22] P. Dias, G. Goncalves, R. Gomes, J. Sousa, J. Pinto, and F. Pereira, "Mission planning and specification in the neptus framework," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 3220 –3225, 15-19 2006.

[23] D. Davis and D. Brutzman, "The autonomous unmanned vehicle workbench: Mission planning, mission rehearsal, and mission replay tool for physics-based x3d visualization," in *14th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, (Durham, New Hampshire), August 2005.