Simulating a Multi-Robot System for Tracking and Monitoring of Coastal Upwellings: Final Project Report

Gilberto Marcon dos Santos Seth McCammon

***Abstract*—For this project, we want to examine the distributed mapping problem in a multi-robot system. One key challenge for this in the underwater domain is asynchronous communications, since ocean gliders are limited in the amount of time that they should spend on the surface. In order to accomplish this, we will build a simulated ocean environment where we can model ocean features, such as upwellings, as well as multiple surface and underwater robots. We also want to begin examining how best to present the maps built by the robots as well as other information to a human operator by means of a decision support system. To accomplish this, we will need to understand what the important characteristics are for controlling a multi-robot system and how to effectively display the status of the system.**

I. Introduction and Related Work

Ocean scientists are interested in exploring and mapping ocean processes, such as coastal upwellings, in order to better understand our oceans and the species that inhabit them. Upwellings occur when longshore winds combine with the rotation of the earth and push warmer surface water offshore, causing cold, nutrient-rich water to be drawn up from the ocean floor. At the interface between the warmer water and the nutrient-rich upwelling, the combination of light, temperature, and nutrients near the surface provides ideal conditions for phytoplankton growth, which in turn serves as the base of a food chain for all manner of marine life including fish, sea birds, and aquatic mammals.

In order to study upwellings and other oceanic processes, researchers typically charter research vessels for week or month-long cruises; however, research vessels have high operation costs, upwards of $30,000 per day [1]. Autonomous Underwater Vehicles (AUVs) and Autonomous Surface Vehicles (ASVs) provide a more cost-effective alternative to research vessels. Buoyancy-driven ocean gliders, such as the Slocum Glider [2], and the Seaglider [3], are common examples of AUVs utilized for scientific exploration. Buoyancy-driven gliders feature low energy consumption, allowing deployments lasting from weeks to months. Single ocean gliders have been successfully deployed for autonomous monitoring tasks, including tracking upwelling fronts [4] and monitoring harmful algae blooms [5].

Small ASVs, such as the Robotic Oceanographic Surface Sampler/Explorer (ROSS/ROSE) [6], have a shorter deployment duration, compared to buoyancy-driven gliders, but are capable of moving more quickly though the environment to rapidly asses points of interest. Additionally, ASVs are more flexible from a communications perspective, since they are not limited to communicating during infrequent surfacing events. This ability allows them to provide a more continuous stream of real-time data to a human operator or to other robots in a Multi Robot System. ASVs' small dimensions and relatively silent operation allows high maneuverability that complements manned research vessels by sampling the ocean surface and air-sea interface in a clean and undisturbed way, allowing access to areas that would otherwise be too dangerous for a manned vessel or would be significantly disturbed by its presence.

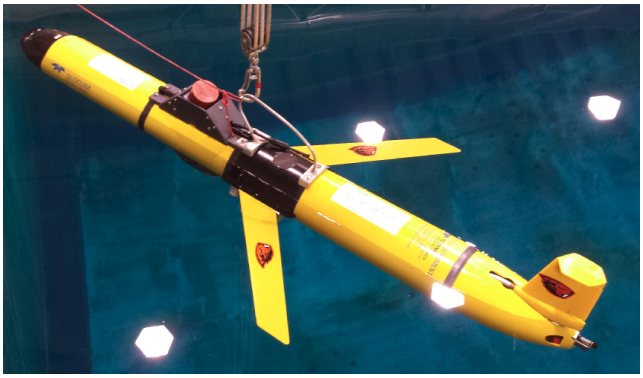

Fig. 1: Slocum Glider [2] used by Oregon State University to monitor coastal upwellings and biological hotspots.

However, use of these tools in real-world applications is limited by many factors. We have identified two of these factors that we seek to address in this project: the ability to visualize the state of a set of marine robots in real time, and the large-scale simulated environment necessary to develop behaviors for one or more aquatic vehicles.

II. Background

A. Oceanographic Data Visualization

Recent work on oceanographic data visualization focuses on fusing satellite and cartographic imagery with data from various sensor sources collected from remotely operated vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs). The visualization tools presented here

Ware, Plumlee, Arsenault, *et al.* [7] present the Geographic Zooming User Interface (GeoZui3D) a software for visualizing 3D data from multiple sources, from satellites to autonomous

underwater vehicles, developed in partnership with the Woods Hole Oceanographic Institution (WHOI). A composite showing the path of an ROV over 1 meter resolution bathymetry visualized with the GeoZui3D graphical user interface is presented by Fig. 2 (a). GeoZui3D shows sub-windows and their location and orientation within the main view, as presented presented by Fig. 2 (b). GeoZui3D generates curtain plots to display water column data from a downward looking high-frequency sonar as presented in Fig. 2 (c). GeoZui3D overlays images captured by an ROV onto ocean floor surface data as presented in Fig. 2 (d).

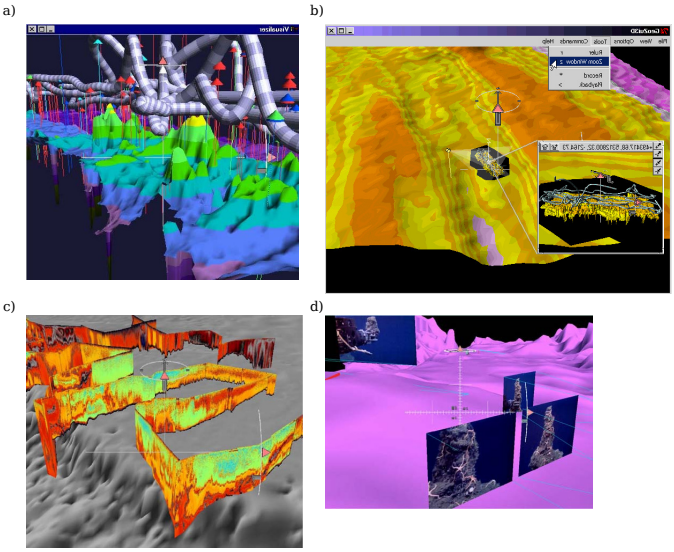

Fig. 2: Data visualization modes provided by GeoZui3D [7].

Arsenault, Ware, Plumlee, *et al.* [8] presents an extension to GeoZui3D for time-varying datasets. A time-line allows the user to dynamically specify time ranges for visualization and playback the data as it was recorded over time, for both littoral and deep waters. Fig. 3 shows water surface levels as it varies with time due to tide changes. The GeoZui3D project is discontinued and there is no source code openly available.

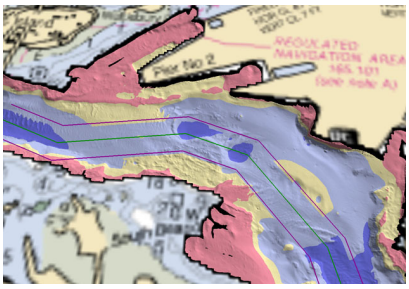

Fig. 3: Data visualization modes provided by GeoZui3D [8].

Schlitzer [9], from the Alfred Wegener Institute for Polar and Marine Research (AWI), presents the Ocean Data View (ODV), a proprietary freely available software package for visualizing oceanographic data in both map-based and time-based plots. ODV is a currently active project that offers a C++ Application Programming Interface and solid documentation. ODV's general framework allows generating scatter plots and shading and contouring over cartographic maps. The software package includes the entire World Ocean Circulation Experiment (WOCE) dataset as example data, with over 200 property distributions, such as oxygen, phosphate, nitrate, silicate, temperature, salinity, and CFC [9]. The ODV graphical user interface is presented by Fig. 4. ODV is made for off-line data visualization, does support live data, and does not have open-sourced code.

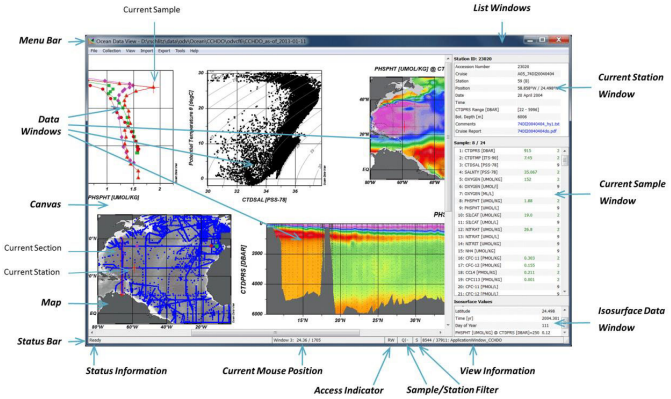

Fig. 4: The ODV graphical user interface [9].

McCann [10], from the Monterey Bay Aquarium Research Institute (MBARI), presents the Geographical Virtual Reality Markup Language (GeoVRML) for visualizing 3D oceanographic data from multiple sources, such as satellite and underwater vehicles. Sensor sources include ship and ROV navigation information, bathymetric maps, collections of geologic and biologic specimens, video frame grabs, annotations of the recorded video, and acoustic backscatter intensity. GeoVRML focuses on playing back data collected from ROV dives and displaying them in the context of historical data. GeoVRML is an open standard that allows representing ROV dives and related data in a human-readable form. Fig. 5 presents data overlay of multiple ROV tracks and satellite depth maps in GeoVRML.

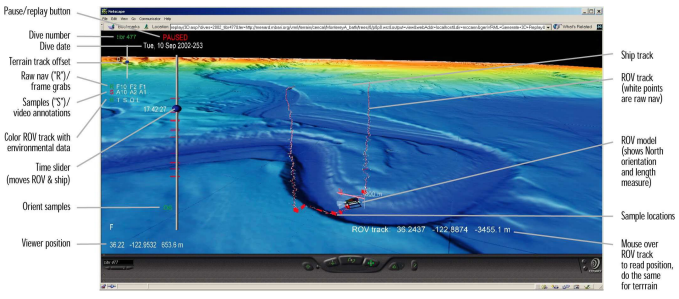

Fig. 5: The GeoVRML graphical user interface [10].

Gomes, Cline, Edgington, *et al.* [11] presents the Oceanographic Decision Support System (ODSS), a web-based decision support system developed by the Monterey Bay Aquarium Research Institute (MBARI) for coordinating and planning multi-vehicle surveys across several different fields of

oceanography. Fig. 6 presents the main user interface of the ODSS.

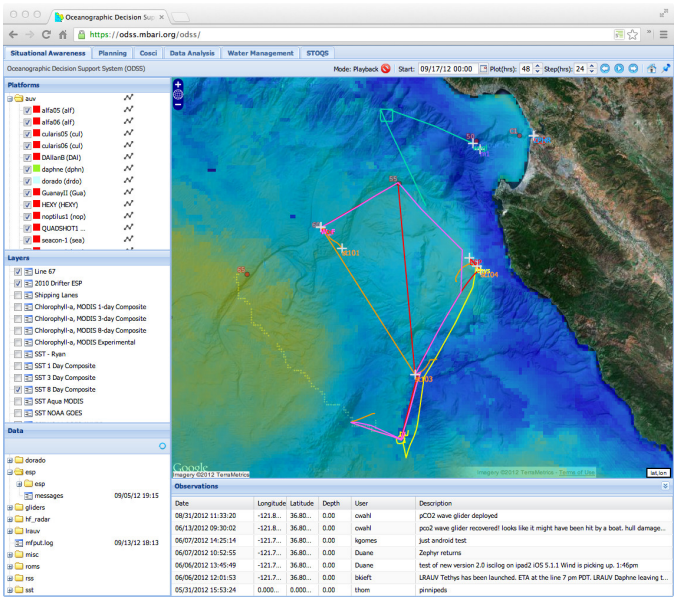

Fig. 6: The main ODSS user interface [11].

The Spatial Temporal Oceanographic Query System (STOQS), also developed by the Monterey Bay Aquarium Research Institute (MBARI), provides efficient access, visualization, and analysis to measurements generated by ships, moorings, drifters, gliders, and autonomous underwater vehicles (AUVs) [12]. Fig. 7 presents the main user interface of the STOQS. ODSS and STOQS do not integrate with the robot operating system.

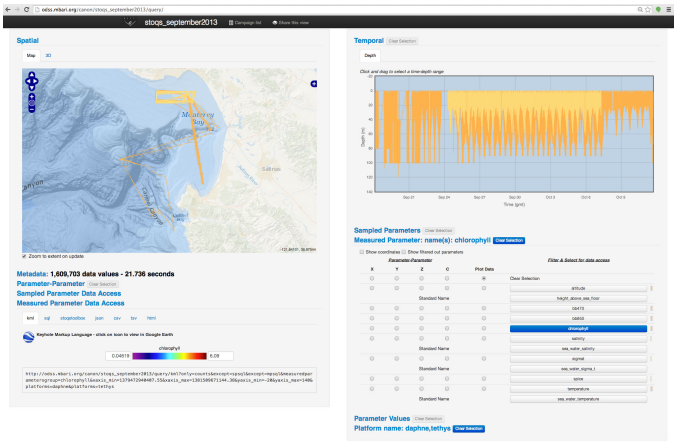

Fig. 7: The main STOQS user interface [12].

Nobre [13] presents OceanPaths, a web-based oceanographic data visualization tool that uses the JavaScript document visualization library D3 [14] and the JavaScript dynamic map visualization framework OpenLayers [15]. OceanPaths utilizes modern frameworks for interactive data visualization and allows the user to generate and visualize custom ocean profiles, including temperature, salinity, oxygen content, independent of the actual profiles produced by the data-collecting vehicles. Fig. 8 presents a custom profile (in red) generated from the actual vehicle profiles, (in blue). OceanPaths is made for offline data visualization and its source code is not openly available.

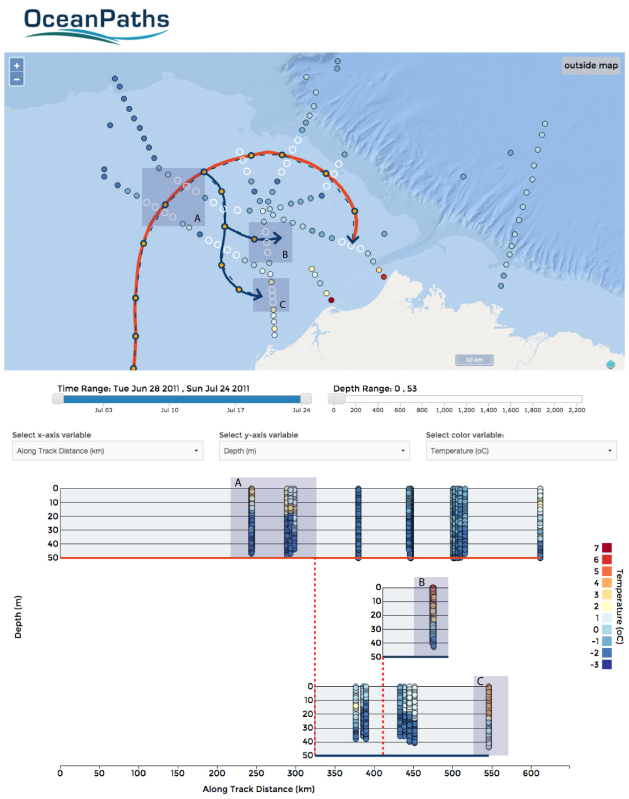

Fig. 8: The OceanPaths graphical user interface [13].

Hanson, Woo, Thomson, *et al.* [16], from the University of Western Australia Oceanographic Institute, present Gliderscope, a tool for visualizing oceanographic data collected by ocean gliders made in Matlab. Data sources include temperature, conductivity, dissolved oxygen, irradiance, optical backscatter of particulates at various wavelengths, fluorescence of phytoplankton pigments (e.g. chlorophyll a, phycocyanin) and coloured dissolved organic matter (CDOM), in addition to acoustic backscatter, acoustic Doppler current profilers (ADCPs), and small-scale turbulence and dissolved nutrient sensors. Gliderscope is an active project that is well documented, but requires the Windows operating system, in addition to Google Earth, which demands an active Internet connection. The main dashboard for loading and selecting glider-collected data is presented by Fig. 9 (a). The plotting interface, for producing different types of plots from glider-collected data, is presented by Fig. 9 (b).

B. Underwater Simulators

Underwater simulators exist for a range of applications, from training UUV operators to developing UUV control and servoing systems. Cook, Vardy, and Lewis [17] presents a

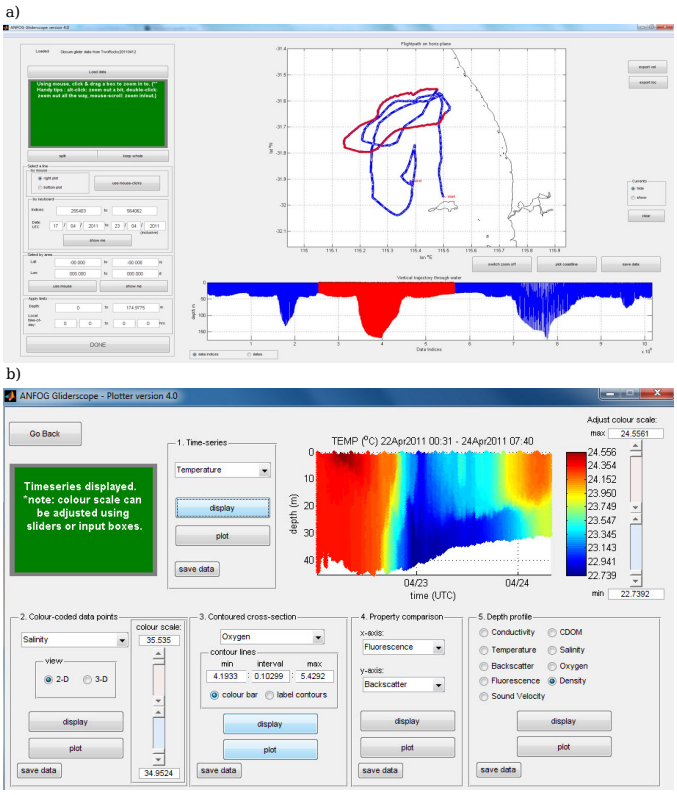

Fig. 9: Data visualization modes provided by Gliderscope [16].

survey of multi-robot simulators focused on AUVs. The survey emphasizes that many AUV simulators referred to in literature are proprietary and focus on operator training, while others are internal academic projects, unavailable for public use. The few publicly available simulators surveyed have a variety of limitations. MORSE [18], although well documented, does not focus on the multi-robot aspects of AUVs, which are regarded as advanced uses and require extra scripting. MORSE is built within Blender; thus, imposes a steep learning curve into a software whose primary use is not robot simulation. V-REP [19], another well documented robot simulator, focuses on modeling inverse kinematics of robot actuators and also does not focus on the multi-robot aspects of AUVs. SubSim Cook, Vardy, and Lewis [17], a simulator focused on AUVs, no longer has active documentation, and supports only one robot.

The UWSim [20] is an open source simulator for AUVs focused on physically and visually accurate underwater simulations for developing visual servoing control. UWSim focuses on single AUVs, and requires inspecting source code due to its lacking and sparse documentation. UWSim also demands laboriously defining simulation parameters and has no up-to-date binary package distributions or building instructions.

Manhães, Scherer, Voss, *et al.* [21] presents a more recent survey focused on the multi-robot aspect of AUV simulation. The survey highlights the deficiencies of existing simulators and introduces an extension to the Gazebo simulator for underwater inspection of offshore wind parks. The UUV Simulator is the most recent, well-documented, and publicly available AUV simulator to support multi-robot systems. The UUV Simulator follows the ROS convention of distributed modular nodes that communicate over a publish-subscribe architecture. The UUV Simulator focuses in small scale underwater dynamics and accurately models underwater sensors, thrusters, fins, and other actuation devices.

The accurate micro-scale simulation accuracy of the UUV Simulator, however, comes as a downside for projects focused on multiple heterogeneous vehicles, especially for monitoring tasks that have week-long deployments and do not require manipulation. The use of Gazebo limits scalability to multi-robot systems: Gazebo is computationally demanding for multi-robot real-time simulations and has an upper limit of ten robots. None of the existing AUV simulators offer the adequate combination of thorough documentation, up-to-date support, and focus on the multi-robot aspects of AUVs, while scaling to simulate time-extended kilometer-scale ocean features, such as upwellings.

To provide a platform that will serve as a testbed for our future work on coordinating multiple marine robots tracking and monitoring coastal oceanographic features, we plan on developing a simulator that will allow us to use large-scale oceanographic models, such as those provided by the Regional Ocean modeling System (ROMS) [22] to build simulated environments for a multi-robot system.

III. System Requirements and Architecture

We developed our system to meet several key requirements given to us by a team of oceanographers and support staff:

1) The interface should present the data collected by all robots, ship-based sensors, and external sources, such as satellites and buoys, in a unified graphical user interface which includes cartographic maps on geographical scales.
2) The system should provide multiple users parallel access to the GUI through a variety of consumer devices, such as smart phones, tablets, and laptops, however not be reliant upon an active Internet connection.
3) The interface should be familiar, easy to use, and similar to existing tools utilized by oceanographic research .

Additionally the simulator should be able to model real-world constraints and conditions encountered by robots in the multi robot system. These include:

1) The simulator should be able to model a time-varying ocean environment.
2) The simulator should be able to model a variety of heterogeneous robots, including autonomous surface vessels and underwater gliders, which have widely differing capabilities, motion models, and limitations.
3) The simulator should reflect the robot's differing modes of communication (freewave radio, iridium satellite, etc), and the constraints that corresponds with each.

Taken together, these requirements demand a modular system that allows for flexible development of a variety modules

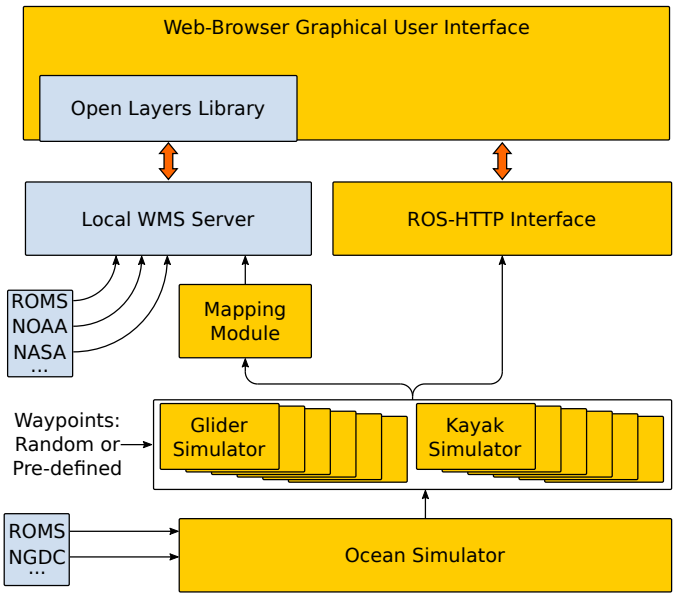

Fig. 10: The system diagram. The modules in blue represent third-party software and databases, while the modules in yellow represent code developed for this project. The thin black arrows represent communication flow through the ROS IPC. The thicker orange arrows indicate communication through HTTP.

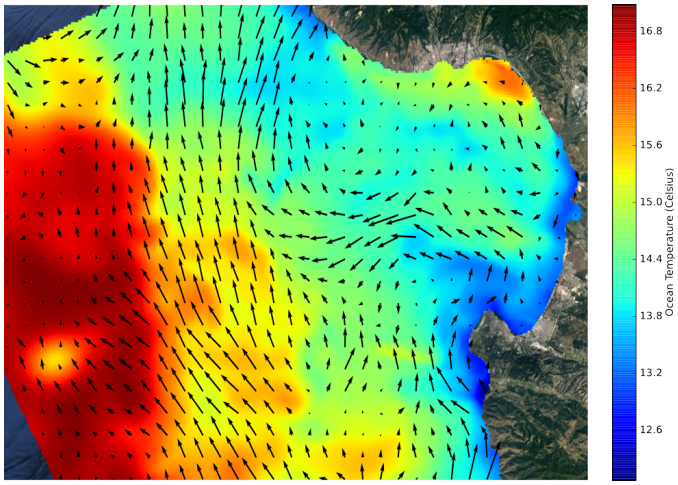

Fig. 11: ROMS data modeling ocean surface temperature and currents in Monterey Bay, CA overlaid on satellite imagery. Visualized using MATLAB. Satellite Imagery from Google Earth.

that can simulate communication with each robot, while providing the core modules with a uniform interface, allowing for easy visualization.

We developed the system architecture shown in Fig. 10 to provide a framework that allows us to meet these requirements and design constraints. We leveraged the Robot Operating System (ROS) [23] as the main inter-process communication (IPC) framework, which allows us to develop each part of the system to as an independent program, providing the uniform interface and modularity that our system requires. The visualization component is handled via a web-based GUI which interfaces with the ROS simulator to provide real-time updates to the visualization of the state of the multi-robot system.

IV. A System-Level Simulator for Multi-Robot Systems in Marine Environments

The first component of this project is a system level-simulator developed within the Robot Operating System (ROS) framework. This simulator uses 3-D models of ocean processes, such as those from the Regional Ocean modeling System (ROMS) [22] to provide realistic ocean conditions in which to simulate the behaviors of a multi-robot system. For the purposes of this simulation, we decided to use the high-resolution (300m) model of Monterey Bay, California [1]. A snapshot of the ROMS data is shown in Fig. 11, where the ocean currents are overlaid on a map of the ocean's surface temperature. Additionally, we incorporate bathymetry data from the National Oceanographic and Atmospheric Administration (NOAA) National Center for Environmental Information (NCEI) to develop a model of the ocean floor and other features.

The simulator component of our project consists of the components labelled 'Glider Simulator' 'Kayak Simulator' and 'Ocean Simulator' shown in Fig. 10. The Ocean Simulator acts as the foundation for our system, importing ROMS and bathymetry data to create a time-varying 3-dimensional model of the ocean. It is implemented as a single ROS node that also acts as a time server for the remaining components, allowing us to simulate at a significantly faster rate than real time. In testing, we found that we were easily able to simulate at 3,600 times faster than real time. Since the ROMS data is discretized into snapshots set 6 hours apart, we use linear interpolation to create a continuous time-varying environment.

The Glider Simulator and Kayak Simulator components each model a single ROSS/ROSE kayak or a Slocum glider. They each consist of a pair of ROS nodes. The first of these handles the ground truth vehicle dynamics and sensing, while the second models the communications and sensing noise that may occur. Each individual vehicle is contained within a single ROS namespace (e.g. kayak_0 or glider_3). The interface with the world simulator is handled through a ROS service call, which provides the sensor information that a vehicle is observing at its current location.

We utilize a relatively simplistic vehicle motion model, whereby upon receiving a waypoint, the vehicle changes heading to point towards the waypoint, and then moves toward it at a constant velocity. The underlying assumption behind this model is that the ocean environment is very large compared to any turning radius that the vehicles may have, and so we can safely ignore how the turning radius impacts the vehicle's path. The ROSS/ROSE kayak, since it is an ASV, is restricted to travelling on the ocean's surface. However, we do model the yo-yo behavior of the Slocum glider. The glider travels

[1]The particular ROMS dataset we utilized is available from http://west.rssoffice.com/ca_roms_nowcast_300m

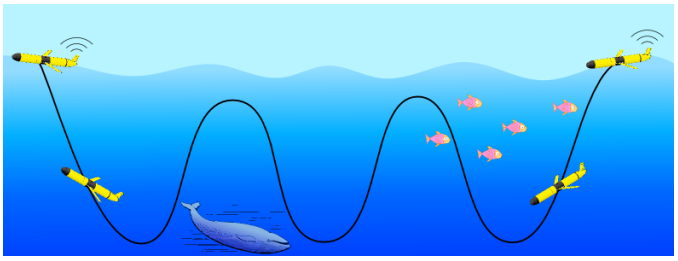

Fig. 12: Yo-yo behavior of the Slocum Glider. The glider must surface in order to utilize its Iridium satellite or freewave radio transmitters.

within a user-specified depth envelope (upper and lower depth bounds). Additionally, the glider has a surfacing behavior, which brings it outside of its depth envelope, and brings it to the surface, allowing it to communicate.

Real robots do not always have perfect estimates of their location in the world, nor do they generate noise-free sensor readings. To model this phenomena, we built a second ROS node that intercepts the ground truth position and sensor information that is generated by the simulator node and injects Gaussian noise to model the noise present in the environment. Additionally, we can apply constraints such as the inability to send communications while the robot is submerged at this point. The result is that from the outside, the robot appears to send sensor information in a similar way as it would in the real world.

V. Graphical User Interface

The second component of our project is to develop a user-friendly interface that will allow a human operator to easily visualize the state of a multi-robot system. Toward this end, we developed a web-based graphical user interface (GUI) that uses a central webserver to display information taken from ROS about the state of the multi-robot system. This is combined with a database of maps to provide a user with a powerful tool for monitoring the multi-robot system.

A. Web-Browser Interface

The web-browser graphical user interface was implemented utilizing the JavaScript language [24], the jQuery [25] JavaScript framework for concise Hypertext Markup Language (HTML) manipulation, and the OpenLayers [26] JavaScript library for interactive visualization of cartographic data in web-browsers. JavaScript was adopted because it is the standard tool for developing front-end web-browser applications, is thoroughly documented, is supported on all major web-browsers, and presents the most extensive set of frameworks and libraries for web development. OpenLayers was adopted because it is open-source, is thoroughly documented, and is the most stable and widely adopted open-source tool for visualizing maps and cartographic data on the web.

The main navigation and visualization command buttons are located at the right-hand side of the main view, presented in Fig. 13. The minimap offers and overview of the surrounding area when expanded, presented in Fig. 14. Collapsed by default, the minimap is presented by pressing the minimap button at the top-right corner of the screen. The layer selector allows the user to interactively select and customize data visualization, presented in Fig. 15. Collapsed by default, the layer selector is presented by pressing the layer button at the top-left corner of the screen. After selecting a layer, the user is presented with visualization options specific to that layer.

The palette and the style selector drop-down menus allow the user to select different color map palettes or contours, presented in Fig. 16. Opacity levels can be adjusted using a slider menu and allows the user to visualize multiple layers of

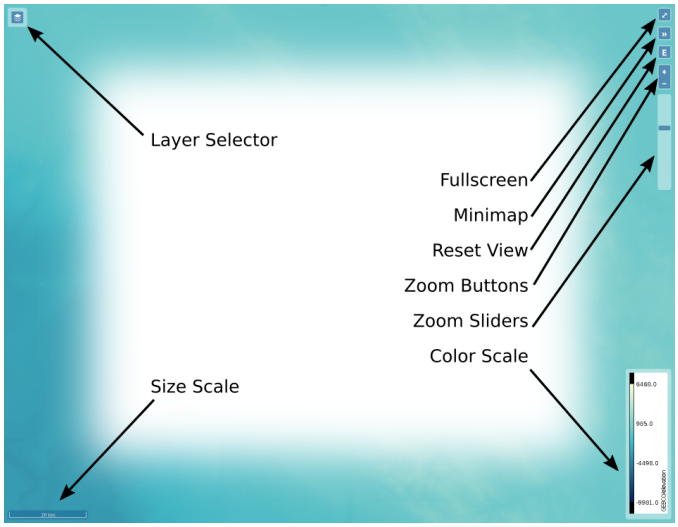

Fig. 13: The main navigation and visualization command buttons.

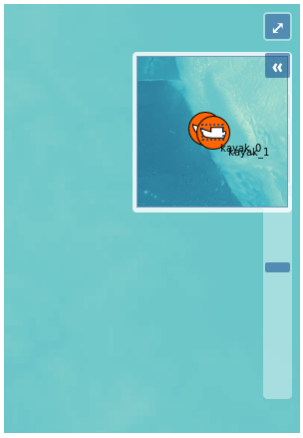

Fig. 14: The minimap offers an inset overview of the surrounding area.

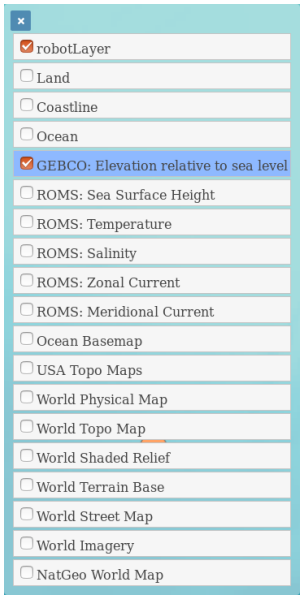

Fig. 15: The layer selector allows the user to interactively select and customize data visualization by enabling and disabling layers to visualize and also by clicking and dragging to change the order of the layers.

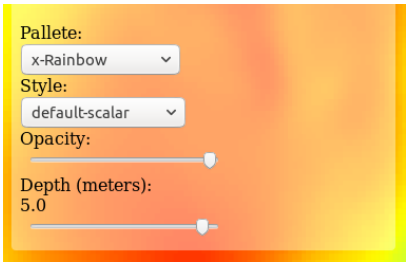

Fig. 16: The layer style selector.

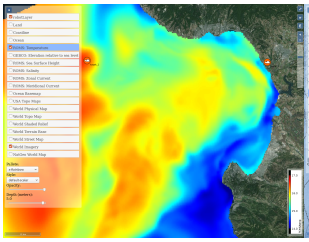

Fig. 17: The color-coded visualization provides a gradient of the desired feature

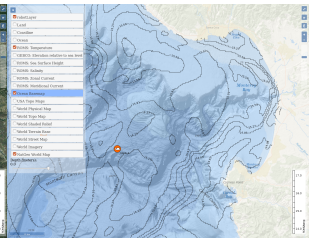

Fig. 18: The contour visualization displays isobars of the desired feature

data over the same geographical region, as presented in Fig. 16. Data sources such as ROMS provide variable readings at various depths, such as temperature, salinity, and currents, and can be visualized by moving the depth slider, presented in Fig. 16.

Temperature data from ROMS is displayed as a color-coded layer over a satellite imagery background layer in Fig. 17, and presented as a contour map over a bathymetry map background in Fig. 18. The use of the layer selector, along with the opacity slider and the contour visualization, allows the user to visualizer multiple sources of data superimposed on the same view, as presented in Fig. 18. The visualization of multiple sources of data is important for decision making on oceanographic deployments, where ocean features are identified by analyzing a combination of variables over a geographical location, instead of each variable in isolation.

The mouse navigation and the color and size scale features are standard from the OpenLayers and the ncWMS tools, respectively, and were activated and parameterized for this project without modifying the framework's core code. The button navigation, the minimap, the robots positions display, and the coordinates pop-up displays, were developed utilizing base functionality offered by the OpenLayers framework, but required modifying the framework's core code to allow it to integrate with our custom HTTP-ROS interface. The layer selector and the visualization controls (opacity, colormap selection, etc) were developed using our own source code.

B. Alternative Visualization Module

An alternative module for data visualization was prototyped during the development of this project that does not rely on a browser. This alternative approach offers different advantages and disadvantages over the web-browser GUI. The advantages include (a) not relying on third party libraries other than a standard low level graphics application programming interface (API); (b) providing the user with lower latency visualization of the state of the system; (c) providing a more fluid view of system dynamics, such as robot's movement; (d) providing a more freely customizable set of visualization features; (e) the ability to natively support 3D visualization modes; (f) the ability to run in lower-end hardware; (g) natively built inside the ROS framework. The disadvantages include (a) restricted compatibility with consumer devices, such as phones, tables, and laptops that do not run a Linux operating system; (b) a steeper learning curve for developing and modifying the user interface; (c) requires compiling code into binaries prior to execution; (d) requires low-level definition of all visualization features; (e) does not leverage the large set of open-source web-based visualization libraries.

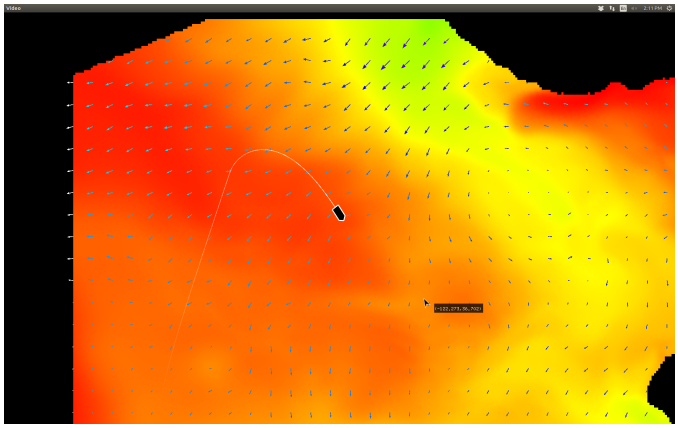

Fig. 19: User Interface for simulated environment presenting the current position of a AUV and its path over the Monterey Bay. The color map represents salinity at the surface. The overlaid quiver plot indicates ocean current magnitude and direction.

The alternative GUI module was developed using C++ and the OpenGL API, the industry standard for high performance graphics, and monitoring the status of the system in real time. The graphical user interface presents the most recent position of each marine vehicle, its latest position history, color-coded layers of water temperature and salinity at various depths, and the water velocity vector field at various depths. User Interface for simulated environment presenting the current position of a AUV and its path over the Monterey Bay, overplayed with salinity at the surface, the vector field of water currents at the surface level is presented in Fig. 19.

VI. Case Study: Autonomous Multi Robot Mapping

In order to demonstrate the capabilities of our simulator and interface, we implemented a simple multi-robot mapping behavior. This behavior combines many of the aspects of our system, as it requires an environment to map, multiple robots to map it, the robots to be able to successfully communicate their observations, and a way to relay the results of the mapping task back to the human. To handle the multi-robot map inference, we leveraged a Sparse Gaussian Process [27], which improves on a standard Gaussian Process (GP) by

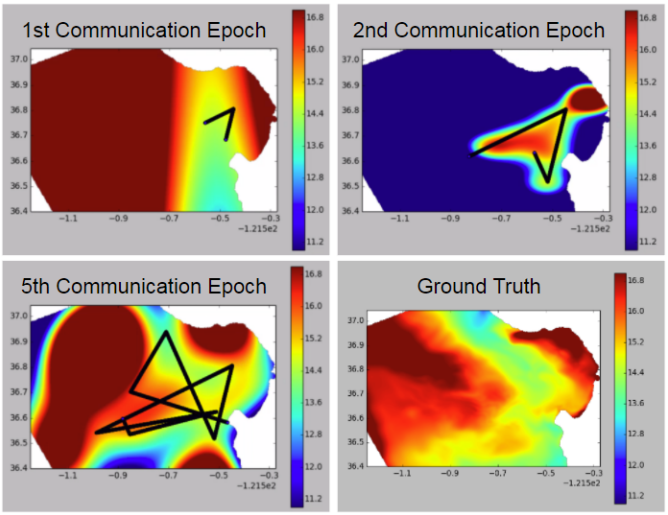

Fig. 20: Simulated mapping task over 5 communications epochs. The black lines represent the paths taken by the two simulated kayaks.

maintaining a set of basis vectors which are then used to compute a GP. The advantage is that the key computational requirement of the GP, a matrix inversion, is significantly reduced in size, since in the Sparse GP, the inversion is computed over a relatively small set of basis vectors, rather than over the full set of observations. In an extended mapping task, like the one we simulate, this set of observations can quickly become intractable for a standard GP.

In the simulated mapping task, we used 2 ROSS/ROSE Kayaks to generate an estimate of the ocean surface temperature. Waypoints for the kayaks were generated randomly (though an area of future work is developing methods for doing this more intelligently). As the two robots explored the environment, they reported their sensor observations to the mapping module over a set of communications epochs. At each of these epochs, the mapping module used the full set of observations to train the Sparse GP. The output of this was then written to a NetCDF file, which was passed to the Web GUI to be displayed for the human user. The results of the case study are shown in Fig. 20. While we did not use any objective metrics to evaluate the accuracy of the estimated map, a visual comparison with the ground truth shows that the estimated map is a good approximation in areas that the robot has visited, and makes reasonable guesses in areas that have yet to be visited.

VII. Conclusion and Future Work

This project is merely the first step towards developing a multi-robot system that can be used to explore our oceans. A robust simulator will allow us to develop more effective mapping behaviors, while a flexible GUI will allow a human user to easily observe and interact with the robot system. We plan on continuing the development of the simulator to improve its reliability and accuracy as well as continuing to work on iterating the interface to provide better, more intuitive interactions for a human user.

There are several areas of the simulator that require improvement. The most glaring one is the lack of any physical interaction that the robots have with their environment. This includes both collision detection with the ocean floor and disturbances from ocean currents. Currently, the simulator will allow a robot to move overland or beneath the ocean's floor. While collision events are rare, we believe that modelling them will allow us to develop some error recovery behaviors, while also improving the realism in our simulated environment. Another area for improvement is in the robot's sensor models. Currently, we assume that the only sensor aboard each robot is a flow-through CTD, which is capable of measuring salinity, temperature, and density at the robot's location. While this closely reflects the Slocum Glider's sensor payload, the ROSS/ROSE kayak also carries a profiling CTD, which enables it to collect CTD data at depths of up to 10 meters. Currently, the ROSS/ROSE is limited to only sampling at the ocean's surface. The final area that we would like to improve the simulator is in the communications model. Currently, we assume that all communications is done via Iridium satellite, which has a global range, though is restricted on bandwidth. We would like to augment this with a model for a range-limited communications, such as freewave radio, since both the glider and the kayak carries a freewave transceiver.

The future work on the GUI includes (a) optimizing the data flow and data presentation mechanisms for a more response user experience and more robust integration of other sources of data; (b) incorporating an advanced plotting library for interactive plots generation, such as scatter and profile plots in arbitrary variable axis; (c) generalizing the ROS-HTTP server for exposing ROS topics and ROS services to the user interface, encapsulated as an API; (d) implementing vector field plots of multi-variable data, such as ocean currents; (e) creasing visualization parameter bundles, or themes, that allow the users to quickly load a visualization mode tailored to specific activities, rather than exposing all individual visualization options to the user; and (f) optimizing the GUI and creating custom visualization modes for portable touch-screen devices, such as tables and smart phones; (g) incorporating time-varying historical data, allowing the user to navigate the data over a time-line; (h) allow the user to display vehicle-specific data, such as battery levels and deployment time; (i) allow the user to define regions of interest and dispatch tasks to the vehicles or groups of vehicles.

Once completed, we see this becoming a powerful tool for marine roboticists, allowing them to simulate their systems in a real-world environment. By designing the simulator to be robot agnostic, a researcher can define the parameters of his or her own robotic platform(s), and use them to explore a simulated world. Furthermore the interface that we develop will allow for a more clear and concise way of interacting with either a simulated or real multi robot system. Hopefully this, when finished, will allow for a smoother development process for multi robot marine systems, enabling them to more smoothly transition from the lab to the real world.

References

[1] National Research Council *et al.*, *Science at sea: Meeting future oceanographic goals with a robust academic research fleet*. National Academy Press, Washington, DC, 2009.

[2] C. Jones, E. Creed, S. Glenn, J. Kerfoot, J. Kohut, C. Mudgal, and O. Schofield, "Slocum gliders - a component of operational oceanography," in *Proc. 14th Int. Symp. on Unmanned Untethered Submersible Technology*, 2005.

[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," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 4, pp. 424–436, 2001.

[4] Y. Zhang, J. G. Bellingham, J. P. Ryan, B. Kieft, and M. J. Stanway, "Autonomous four-dimensional mapping and tracking of a coastal upwelling front by an autonomous underwater vehicle," *Journal of Field Robotics*, vol. 33, no. 1, pp. 67–81, 2016.

[5] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *Journal of Field Robotics*, vol. 28, no. 5, pp. 714–741, 2011.

[6] J. A. MacKinnon, J. D. Nash, M. H. Alford, A. J. Lucas, J. B. Mickett, E. L. Shroyer, A. F. Waterhouse, A. Tandon, D. Sengupta, A. Mahadevan, M. Ravichandran, R. Pinkel, D. L. Rudnick, C. B. Whalen, M. S. Alberty, J. S. Lekha, E. C. Fine, D. Chaudhuri, and G. L. Wagner, "A tale of two spicy seas," *Oceanography*, vol. 29, no. 2, pp. 50–61, 2016.

[7] C. Ware, M. Plumlee, R. Arsenault, L. A. Mayer, and S. Smith, "Geozui3d: Data fusion for interpreting oceanographic data," in *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295)*, vol. 3, 2001, pp. 1960–1964.

[8] R. Arsenault, C. Ware, M. Plumlee, S. Martin, L. L. Whitcomb, D. Wiley, T. Gross, and A. Bilgili, "A system for visualizing time varying oceanographic 3d data," in *OCEANS 04. MTTS/IEEE TECHNO-OCEAN 04*, vol. 2, 2004, pp. 743–747.

[9] R. Schlitzer, "Interactive analysis and visualization of geoscience data with ocean data view," *Computers & geosciences*, vol. 28, no. 10, pp. 1211–1218, 2002.

[10] M. P. McCann, "Using geoVRML for 3d oceanographic data visualizations," in *Proceedings of the ninth international conference on 3D Web technology*, ACM, 2004, pp. 15–21.

[11] K. Gomes, D. Cline, D. Edgington, M. Godin, T. Maughan, M. McCann, T. O'Reilly, F. Bahr, F. Chavez, M. Messi, J. Das, and K. Rajan, "ODSS: A decision support system for ocean exploration," in *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*, 2013, pp. 200–211.

[12] M. McCann, R. Schramm, D. Cline, R. Michisaki, J. Harvey, and J. Ryan, "Using STOQS (the spatial temporal oceanographic query system) to manage, visualize, and understand AUV, glider, and mooring data," in *2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*, 2014, pp. 1–10.

[13] C. Nobre, "Oceanpaths: Visualizing multivariate oceanography data," PhD thesis, 2016.

[14] M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-driven documents," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[15] T. Gratier, P. Spencer, and E. Hazzard, *Openlayers 3: Beginner's guide*. Packt Publishing Ltd, 2015.

[16] C. E. Hanson, L. M. Woo, P. G. Thomson, and C. B. Pattiaratchi, *Observing the ocean with gliders: Techniques for data visualization and analysis*, 2017.

[17] D. Cook, A. Vardy, and R. Lewis, "A survey of auv and robot simulators for multi-vehicle operations," in *Autonomous Underwater Vehicles (AUV), 2014 IEEE/OES*, IEEE, 2014, pp. 1–8.

[18] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular open robots simulation engine: Morse," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 46–51.

[19] E. Rohmer, S. P. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2013, pp. 1321–1326.

[20] M. Prats, J. Pérez, J. J. Fernández, and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 2577–2582.

[21] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation," in *OCEANS 2016 MTS/IEEE Monterey*, IEEE, 2016, pp. 1–8.

[22] A. F. Shchepetkin and J. C. McWilliams, "The regional oceanic modeling system (roms): A split-explicit, free-surface, topography-following-coordinate oceanic model," *Ocean Modelling*, vol. 9, no. 4, pp. 347–404, 2005.

[23] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. of the IEEE International Conf. on Robotics and Automation: Workshop on Open Source Software*, 2009.

[24] D. Flanagan, *JavaScript: The definitive guide*. " O'Reilly Media, Inc.", 2006.

[25] K. De Volder, "JQuery: A generic code browser with a declarative configuration language," in *PADL*, Springer, vol. 6, 2006, pp. 88–102.

[26] T. Gratier, P. Spencer, and E. Hazzard, *OpenLayers 3: Beginner's guide*. Packt Publishing Ltd, 2015.

[27] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural computation*, vol. 14, no. 3, pp. 641–668, 2002.