Onboard Autonomy System for the Slocum Glider

Seth McCammon*, T. P. Welch†, C. M. Waluk‡, Kelly Benoit-Bird‡, J. A. Barth†, and Geoffrey A. Hollinger*
*Collaborative Robotics and Intelligent Systems Institute, Oregon State University,
{mccammos, geoff.hollinger}@oregonstate.edu
†College of Earth, Ocean and Atmospheric Sciences, Oregon State University {tpw, barth}@ceoas.oregonstate.edu
‡Monterey Bay Aquarium Research Institute {kbb, cwaluk}@mbari.org

***Abstract*—In this paper we present a system for adaptive, long-term underwater exploration and monitoring tasks using buoyancy-driven underwater gliders. To address the limitations of shore-based autonomy for long-term glider deployments, we develop an embedded autonomy system and test it on a TWR Slocum G2 glider. Our system places the autonomy physically within the glider, which greatly diminishes the constraints on the amount of data that can be passed to the autonomy system. Furthermore, our approach eliminates latency in planning that is a result of relying on satellite communications during glider surfacings. In this paper we provide a detailed discussion of the hardware and software components of our embedded navigation (NAV) system. We also demonstrate the capability of the NAV system in the field through results from a pair of sea trials off of the Oregon Coast where the NAV system was used in an adaptive sampling task to identify temperature gradients.**

***Index Terms*—Autonomous Underwater Vehicles, embedded autonomy, adaptive sampling**

Fig. 1: Slocum G2 Glider equipped with the NAV system and active acoustics used in [5]. The NAV system is housed in the black science bay alongside the G2's onboard science computer, just behind the red-tipped active acoustics heads.

I. Introduction

Buoyancy-driven vehicles, such as the Slocum glider [1] shown in Fig. 1, are ideal for conducting long-term monitoring tasks, including mapping upwelling fronts [2] or monitoring harmful algae blooms [3]. Their endurance, typically measured in weeks to months, allows for more cost-effective monitoring of subsurface ocean features, especially when compared to research vessel time, whose costs can be over $30,000 per day [4]. However, gliders are not without their limitations. The Iridium satellite link which forms the communications channel to the glider is a significant barrier to adaptive autonomy. In this paper we propose removing this barrier through embedding a small, efficient computer within the glider to perform autonomous navigation onboard the glider. This lets us avoid using the satellite link to both transmit the data required for autonomy and transmit plans back to the glider.

Typically, these vehicles are controlled from a base station onshore or onboard a research vessel via satellite communications. The bandwidth of the satellite communications is often quite limited, preventing the vehicle from transmitting large quantities of data back to the base station. In practice this is typically addressed by only returning a limited subset of the data to shore. However, for particularly data-intensive sensors, such as active acoustics [5], it becomes impossible to transmit sufficient quantities of data to a shore-based system to enable autonomous behaviors. We would like to draw a distinction between the ability to autonomously execute a series of waypoints, which is a common feature on most gliders, Autonomous Underwater Vehicles (AUVs), and Autonomous Surface Vehicles (ASVs), and the more general adaptive autonomy algorithms developed in the robotics community. These algorithms involve making decisions and adapting robotic behavior in response to observations about the environment [6] such as mapping and tracking biological hotspots [7].

Unlike a research vessel or an ASV where course updates can be processed the moment they are generated or transmitted, updates for AUVs can typically only be received when the vehicle is on the surface. Radio communications cannot penetrate through the water, meaning that unless an AUV carries an underwater communications modality it is rendered incommunicado. Even with an acoustic modem, the AUV could only communicate with other similarly-equipped vehicles within a fairly small range (1 to 10 km), and with an incredibly limited baud rate (up to 15 kbps depending on communications distance). For shore-based autonomy, this means that either the vehicle must be held on the surface while a new plan is generated or the new plan must be generated before the vehicle surfaces. There are major flaws with both

This work was funded in part by support from a W. M. Keck Foundation grant to J. Barth, K. Benoit-Bird, and G. Hollinger and NSF grant IIS-1723924.

The authors would also like to acknowledge the indispensable contributions of Anatoli Erofeev and Steve Pierce along with the captains and crews of the R/V Paragon and R/V Elakha.

of these approaches. If the glider is held at the surface while a new plan is prepared it exposes itself to being struck by passing surface traffic or being blown off-course by the wind and surface currents. Due to the limited satellite bandwidth, it can take 10 or more minutes to transmit the glider's data back to the base station. Adding to that the time it takes for the shore-based autonomy to determine a new path and time to transmit the new path back to the glider, the glider can be on the surface exposed to these dangers for 30 to 40 minutes at a time. With our embedded autonomy system, there is a minimal increase in the amount of data transmitted to shore, avoiding the need for overly long surfacings.

Pre-generating plans introduces a latency of a minimum of one surfacing interval (commonly 2 to 6 hours) between when the plan is created and when the plan is adopted by the glider [8]. This means the plans are created using observations that may be hours old and that there can be a significant amount of uncertainty over the true vehicle location during the path planning. Depending on how dynamic the environmental process is that the glider is attempting to sample, the age of the observations can play a significant role in the performance of the vehicle in the sampling task. Perhaps more problematic is the issue of uncertainty in the vehicle's location. While the vehicle's position may be estimated via interpolating along its previous path, the presence of space and time-varying currents and other disturbances can easily diminish the accuracy of such predictive methods. Significant errors in the position estimate can greatly diminish the effectiveness of many autonomous sampling algorithms, since typically the rewards considered by those planners are path-dependant.

To address these limitations, we embedded a small navigation (NAV) computer within the science bay of the glider. By locating the autonomy system onboard the glider, the NAV computer gains access to the data collected by the glider in real-time, enabling the glider to make autonomous navigation decisions without requiring shore communications. We provide an extended discussion of our design requirements and constraints, and the process of constructing the NAV system to meet these. Furthermore, we show the operational utility of the NAV system in a pair of sea trials. The remainder of the paper is structured as follows. Section II provides a discussion of related work in glider and other AUV field deployments, as well as embedded autonomy in other areas of field robotics. Section III describes the main design criteria we considered while designing our system. The hardware and software components of the NAV system are described in Section IV and V, respectively. Finally, in Section VI, we demonstrate the utility of our system in a set of sea trials off of the Oregon Coast using a Slocum G2 glider testbed.

II. Related Work

Recently, there has been a significant amount of interest in the use of autonomous marine vehicles for various sampling and target-seeking tasks. A wide array of informative path planning algorithms have been proposed as potential solutions for providing autonomous control for one or more marine vehicles during a sampling mission. These planners include sampling based planners [9], anytime planners which use exhaustive search techniques [10] [11], and information-theoretic planners [12] [13]. These algorithms have been tested extensively in simulation and small-scale field tests, using lakes and reservoirs as stand-ins for the open ocean. As we seek to implement these state-of-the-art autonomy algorithms on seagoing AUVs for ocean sampling tasks, we must first address some of the hardware limitations of widely used AUV platforms, such as the Slocum G2.

The G2 glider and other AUVs have been widely utilized in large-scale marine sampling tasks, either alone [2] [5] [14], in groups [8], or part of larger heterogeneous multi-asset deployments [3] [15] [16]. These sampling tasks involve solving NP-Hard problems, requiring significant CPU processing power and memory. Some AUVs such as the Tethys AUV used in [2] or [14] carry sufficient on-board computation to run these algorithms however the Slocum G2 does not. Its main science computer was designed in the mid-1990s, and has a mere 1MB of RAM. In order to conduct autonomous operations, Slocum Gliders therefore rely on shore-based computers to generate and transmit plans via Iridium to the glider. This methodology has been used to some success in controlling teams of gliders in Monterey Bay [8] [15]; however the authors acknowledge the limitations imposed by the latency that results from needing to plan before the gliders are on the surface. In our work we seek to address this limitation of the G2 glider platform by moving the autonomy computer on board the glider.

This work follows other work in field robotics, where the capabilities of existing robotic platforms are augmented by power-efficient, small form factor computers that are placed onboard field vehicles to enable autonomous behavior. By mounting a smartphone aboard a quadcopter the authors of [17] have been able to demonstrate improved navigation in GPS-denied environments using only local computational resources. Other UAV systems take advantage of the computation provided by NVIDIA's Jetson boards [18] for performing task allocation or an Intel NUC [19] enabling fast perception and control.

III. Design Requirements

Our NAV system consists of an small computer mounted within the glider along with custom software packages for both the NAV computer and the G2's science computer. In the initial stages of developing the NAV system, we identified four main requirements that must be met:

- **Physical Space Constraints:** The free space on the interior of an underwater glider is quite limited. Batteries, ballast pumps, scientific instrumentation, and the glider's own flight and science computers take up most of the available space within the hull. The NAV system must physically fit alongside all of these items within the confines of the Glider's 22cm diameter hull.
- **Maximize Computation:** Adaptive sampling and informative path planning are both problems known to be NP-

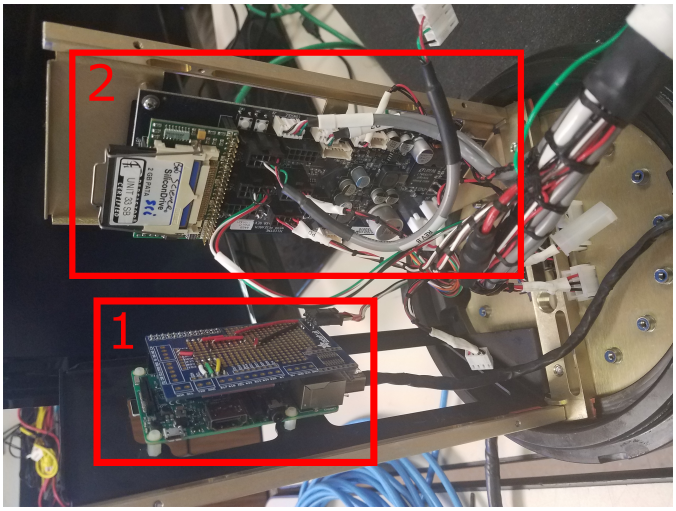

Fig. 2: Raspberry Pi 3 NAV Computer (1) mounted in exposed Slocum G2 Glider science bay opposite the science computer (2). The shield containing the TTL/RS232 Converter, Step-Down voltage regulator and denoising inductors can be seen mounted on top of the Pi.

Hard [9], meaning that computing optimal solutions can be very computationally intensive. In order to support this computational load, we would like to maximize the amount of compute power in the NAV system.

- **Interface with Existing Systems:** Due to the proprietary nature of many of the Slocum glider's systems, there is little flexibility on the glider end in terms of modifications to software protocols and hardware interfaces. Since the NAV system must be be capable of interfacing with the glider to perform its function, we require that the NAV system is able to connect with the glider reliably and robustly. In addition, the NAV system should be able to communicate with other sensors (such as active acoustics) that may be added to the glider's payload either independently or through the glider's science computer.
- **Maximize Mission Duration:** One of the main features of the Slocum glider and other glider-type AUVs is their power-efficiency and endurance. In order to preserve the glider's endurance as much as possible, the NAV system should draw minimal power from the glider's batteries. It is worth noting that this design requirement is directly at odds with maximizing computation, since generally more powerful computers also require more power.

In the following sections we provide an extended discussion of the design choices and compromises we had to make in order to meet these requirements.

IV. Hardware Description

There are a wide variety of small form factor computers that could have served as the main NAV computer in our system. We quickly narrowed our search to considering the Raspberry Pi 3 and the Gigabyte Brix, which is an Intel NUC. These two computers sit at opposite ends of the power spectrum. The Gigabyte Brix supports a powerful Intel i7 processor and several gigabytes of RAM. However this comes at a cost; the Brix has a relatively large form factor (5cm $\times$ 11.25cm $\times$ 12cm) along with relatively hefty power requirements. On the other end of that spectrum lies the Raspberry Pi 3B. The Pi has an ARM processor and 1GB of RAM, however it is significantly smaller than the Brix, occupying only 1.7cm $\times$ 8.5cm $\times$ 5.5cm. This smaller form factor meant that it, unlike the Brix, could fit within the glider's science bay opposite the science computer with minimal modifications. In order to fit the Brix in our G2 glider, we would have had to have mounted it within our custom-built acoustics bay, alongside a DC voltage transformer and other equipment for operating the active acoustics.

In addition to being a smaller form factor, the Raspberry Pi has an advantage in terms of power draw. In bench testing of the two devices we found that the Brix drew 7.5 Watts of power while the Raspberry Pi used 1.9 Watts. Both computers were measured under a typical load, without adding any of the peripherals that they each would need (such as serial converters or voltage transformers). The Slocum G2 can be equipped with lithium or alkaline primary batteries. The lithium batteries have a significantly larger battery capacity, however they operate at a slightly lower average voltage. The effects of this on the total power requirements and mission duration is shown in Table I. Based on these measurements we predict a 41% to 46% reduction in mission time using the Raspberry Pi as the NAV computer versus a 73% to 77% reduction using the Brix. These numbers are computed ignoring the power requirements of the active acoustics which would add about 22 Amp-hours per day to the power requirements using alkaline batteries. Given that one of the key advantages of the Slocum glider is its ability to stay on-station conducting monitoring tasks for long periods of time, we determined that the Brix demanded too much of a sacrifice in terms of battery life to be worth the added computational resources. When combined with the additional difficulties of mounting the Brix within the glider, we selected the Raspberry Pi as the hardware for our NAV computer, as it better met our fourth design requirement while still providing sufficient computational resources for autonomy.

In order to interface with the glider's onboard systems, the NAV computer was mounted in the glider's science bay using a custom fabricated aluminum mounting bracket (as shown in Fig. 2). The Pi computer was connected to the glider's science computer by a single cable that provided power to the Raspberry Pi as well as carried the bidirectional serial communications. The NAV computer was connected to the acoustics computer via an Ethernet cable that ran along the glider's length. To fully integrate the NAV computer into the glider, we needed to step-down 15 Volt supply battery voltage to the 5 Volts required by the Pi. Additionally, the Raspberry Pi's GPIO pins use TTL levels for serial communications as opposed to the RS232 levels used by the glider. In order to allow communication between the two computers, a converter chip that translated between the two serial levels was required. Finally, since the active acoustic transducer heads are quite sensitive to noise (both mechanical and electrical) in the glider,

TABLE I: Power requirements for Slocum Glider and NAV system candidates

			Slocum G2 Glider		Gigabyte Brix (7.5 W)			Raspberry Pi 3B (1.9 W)		
	Battery Voltage	Usable Battery Capacity	Power Consumption	Operational Duration	Total Power Consumption (Brix + Glider)	Estimated Operational Duration	Estimated Duration Reduction	Total Power Consumption (Pi + Glider)	Estimated Operational Duration	Estimated Duration Reduction
Alkaline	13 V	140 Ah	5 A-h/day	28 days	18.9 A-h/day	7.4 days	73%	8.5 A-h/day	16.5 days	41%
Lithium	10.5 V	630 Ah	5 A-h/day	126 days	22.1 A-h/day	28.5 days	77%	9.3 A-h/day	67.5 days	46%

mitigating any electrical noise caused by the NAV system was important to maintaining high performance in the acoustics. To accomplish this we used a pair of 47 μH inductors to dampen any electrical noise into or out of the NAV system. All of these components: inductors, TTL/RS232 converter, and voltage regulator were attached to a shield which itself attached to the GPIO pins on the Pi. When fully assembled, the shield added 1 cm to the height of the NAV computer assembly, still comfortably fitting in the glider's science bay as can be seen in Fig 2.

V. Software Description

In order to support our autonomy algorithms, we developed a custom software package that would allow the NAV computer to send commands such as new waypoints, requests for surfacings, and modification of the glider's yo-yo depth envelope via serial strings. Additionally through this interface the glider supplied the NAV computer with the data collected by its sensors in near real time. Communications back and forth between the NAV computer and the glider computer over a serial connection used a NMEA sentence syntax, which includes a checksum to help reject corrupted or incomplete messages. These communications allowed the glider to modify its mission on the fly and perform adaptive autonomy.

Although there is a serial connection between the NAV computer and the glider, since the NAV computer is an entirely separate computer system to the glider's flight and science computers, they do not directly share files and data. This means that when a glider surfaces and connects to the dockserver via its Iridium satellite link to allow the download of its locally stored data, the glider's operator has no access to the data stored on the Pi. To address this issue, we developed a set of heartbeat messages which regularly report status information to the glider, so that it can be relayed to a shoreside user. These heartbeat logs, along with the commands issued by the NAV system and observations of the behavior of the glider as it moves through the ocean are the only ways that a human operator could glean information about the internal state (system status, planning logic, etc.) of the NAV system while underway. The heartbeat messages contained information about the 'health' of the various software components that together enable autonomous behavior.

A. NAV System Software

Onboard the NAV computer, we used the Robot Operating System (ROS) [20], an open-source software platform developed specifically for robotics applications. One prevalent issue in robotics, and particularly marine robotics, is the plethora of custom software interfaces for different vehicles. While there have been some attempts to build similar standardized interfaces specifically for marine vehicles, such as MOOS and MOOS-IVP [21], ROS is a more widely-used system for autonomous ground and air vehicles, with a variety of freely available implementations of autonomy algorithms as well as sensor and vehicle drivers. By using ROS as the foundation of our system, we are able to streamline future development, by having written a set of software drivers for the Slocum glider. We use these drivers to translate the raw data coming from the glider into standard ROS data types which are easily interpreted by any algorithm implemented within a ROS-based framework.

ROS divides up the robotic control software into a set of nodes, each of which can be specialized and dedicated to a single functionality. ROS provides the middleware infrastructure to allow messages to be passed between these nodes via TCP, allowing them to share information. In our system we wrote five such nodes:

- **Glider Communications Node:** This node served as the interface between the ROS system and the Glider. its role was to ensure that the proper messages were being sent back and forth across the serial connection. In addition to providing syntax checking on incoming and outgoing messages, this node maintained a queue of messages to send to the glider. To ensure that the glider received each message, the top message in the queue would be re-sent to the glider every second until an acknowledgement of the message was received, at which point the message would be popped from the message queue and the next message would be sent.
- **Bio-Acoustic Transducer Communications Node:** Similar to the glider communications node, this node translated the communications from the active acoustics system into messages within the ROS framework. Unlike the glider communications where messages were passed in both directions over a serial connection, communications with the active acoustics consisted of UDP packets which only went one-way, from the acoustics to the NAV. This node parsed the incoming UDP packets which contained acoustic data encoded as NMEA sentences and re-broadcast their contents within the ROS architecture. We conducted deployments both with and without the active acoustics. When the active acoustics were not present, this node was disabled to reduce the battery consumption of the NAV computer.
- **World Estimation Node:** In order to autonomously

make decisions on how to move about the world, the NAV system must first use the data that the glider collects to build an internal estimate of the state of the world. To accomplish this, we used a Gaussian Process (GP) [22], which is a powerful tool commonly used in field robotics to estimate the value of an environmental variable across an N-dimensional spatiotemporal domain $\mu(x) = \{\mathbb{R}^N \mapsto \mathbb{R}\}$ [9] [13] along with generating a measure of uncertainty in the value of that variable, σ. In our case we used a GP to estimate the magnitude of the acoustic returns measured via the active acoustics. On deployments without acoustics, the environmental variable of choice was typically salinity or temperature.

- **Glider Navigation Node:** In order to reduce the amount of computation, we did not want the planning algorithm to constantly be planning new paths for the glider. The Glider Navigation Node is responsible for determining when a new plan is required, *i.e.* when 1) the glider has reached the end of its previous path or 2) a pre-defined amount of time has passed since the glider last received a new plan. At this point, the glider navigation node will issue a request to the world estimation node for an updated world model. Once the request is received this node will pass the updated world model, along with the latest information on the glider's position to the planning node.
- **Planning Node:** The final node in the NAV system is the node which performs the planning for the vehicle. During the various deployments of our NAV system we used two different planners for this task. The simpler of these planners simply directed the glider to head toward the point in the environment where the objective function is maximized. Typically this objective function was either the value of the environmental variable modelled by the GP, μ, or the value of the α-upper confidence bound $\mu+\alpha\times\sigma$. The parameter α is a constant scaling factor that trades off between exploitation of previously observed information and exploration of unobserved areas in the environment. The other planner used in conjunction with the NAV system is our topological hotspot planning algorithm. This planner divides the environment into a set of hotspots and then determines how much time to spend sampling within each one. The hotspot algorithm is described in more detail in [7].

B. Adaptations to Glider Software

The standard operating mode for a glider is to follow a predefined set of waypoints recorded in a file called a 'goto file'. If the human operator or a shore-based autonomy algorithm wants to alter the glider's behavior, a new goto file must be generated and uploaded to the glider via Iridium. In order to allow the NAV system to make modifications to the glider's path while underway, we developed a module which ran on the glider's science computer which enabled it to change the active waypoint in the glider's mission while underway based on the waypoint commands it received from

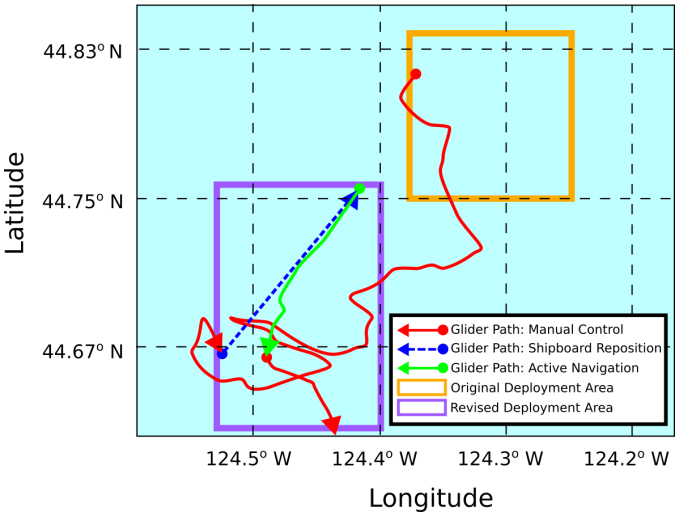

Fig. 3: Path taken by glider in June 2018 deployment off the Oregon Coast. Due to strong currents in the initial testing area, midway through the deployment we recovered and redeployed the glider in a new zone further offshore to reduce the effects of these currents on the active navigation.

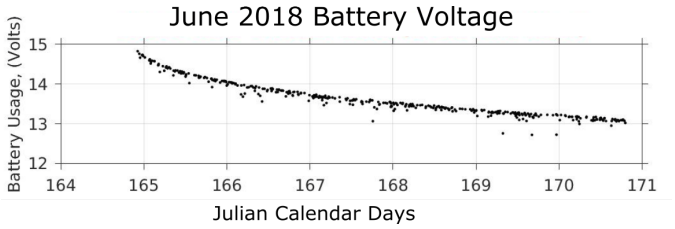

Fig. 4: Primary alkaline battery voltage for Slocum Glider for June 2018 deployment.

the NAV computer. Due to the asynchronous operations of the glider's flight and science computers a delay of 8 s or about twice the wake up cycle of the glider's flight computer was introduced on the NAV computer between messages sent to the glider's science computer.

VI. Sea Trials

After constructing the NAV system, we wanted to test it using a glider in real-world conditions. The three main goals of these tests were 1) to evaluate whether the Raspberry Pi carried enough computational power to function as an autonomy computer, 2) to confirm our estimations on how the NAV system would impact the maximum mission length of the glider, and 3) to test our hotspot-based autonomy algorithm in a real-world environment.

To perform these tests, we conducted initial sea trials in 2017 in Monterey Bay and near Newport, off the Oregon Coast using a Slocum G2 Glider equipped with alkaline batteries. During these initial trials, the NAV system was in a ride-along mode to test its draw on the glider's batteries, particularly when mounted alongside the active acoustics. In this mode, the NAV system was fully operational in the glider, with the sole caveat that the planning node was disabled and therefore not generating plans to pass back to the glider. During some of the trials in late 2017, we used the NAV system to generate

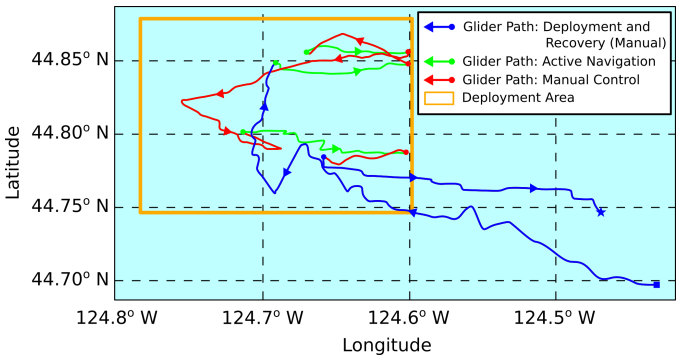

Fig. 5: Path taken by glider in August 2018 deployment off the Oregon coast. As can be seen in Fig. 7, the strongest temperature gradient lies along the eastern side of the deployment zone. As a result, when the glider was operating under active navigation, it targets this region to sample in high-gradient regions.

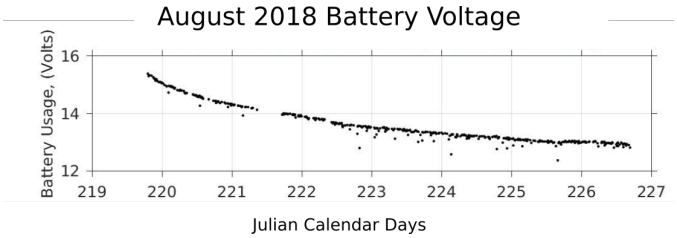

Fig. 6: Primary alkaline battery voltage for Slocum Glider for August 2018 deployment

single waypoints for the glider (as opposed to complete paths) to verify that our software on the glider's computer would properly adjust the glider's waypoint as a stepping stone toward testing fully autonomous behaviors. Unfortunately, this testing was cut short as the acoustics-equipped glider was lost at sea. We continued testing of the NAV system without acoustics (which could not be replaced) in 2018, in a series of trials off the Oregon coast. In these trials instead of using the autonomous capabilities to find areas with high acoustic returns, we instead used temperature and salinity as a proxy metric. The sea trial results presented here will focus on two trials conducted in June and August of 2018.

During these trials, we used the NAV system to perform active navigation for the glider, leveraging data collected in situ to inform decisions on where to send the glider. For each of these deployments, the glider would initially fly a pre-determined pattern to collect a sparse initial dataset to allow the NAV system to construct a model of the deployment area. Once the NAV system has an environmental model, it would then use the model to plan a path to maximize the utility of the glider's sampling using the algorithm described in [7]. The objective for the glider in both of these experiments was to identify regions of the environment which contained strong temperature gradients as might be found in a coastal upwelling.

Fig. 3 shows the path that the glider took during the deployment in June of 2018. During the June 2018 deployment, we found there to be strong southerly currents in the deployment area, and so a new, larger area (shown in purple) was chosen farther South-West, and the glider was recovered and re-positioned using a ship. Once re-deployed, the glider operated under active navigation for nearly 24 hours, before returning to manual control to rendezvous with the ship for recovery. The glider's battery voltage during the June deployment is shown in Fig. 4. While we did not deploy the glider for a sufficient duration to completely drain the batteries, the observed voltage decay is in line with our previously calculated mission duration of about 19 days.

In a second trial in August 2018, we operated the glider within the deployment zone under a combination of active navigation and human guidance for approximately 118 hours (4.9 days). The path the glider took during that deployment is shown in Fig. 5. The blue segment of the trajectory shows the path that the glider took from its original deployment point (the square) to the operational zone, including the initial sparse survey as well as the path back to the recovery point (the star) after the conclusion of the experiment. After arriving in the zone and completing the initial survey, the glider was either operating autonomously under active navigation (green) or being guided by manually generated waypoints (red). The glider's battery voltage is shown in Fig. 5. In this deployment, the glider's recovery was motivated by ship availability, and therefore it was recovered before the battery was fully depleted. However, as in June, the August deployment's battery consumption was in-line with our estimates, indicating an approximate 12 day battery lifespan using the NAV system.

In addition to evaluating the NAV system as a component of a G2 glider, the August deployment was a chance to test the performance of our active navigation algorithm. Over the course of the deployment, a front was identified, and through a combination of autonomous and manual planning, it was repeatedly sampled. Fig. 7 shows the samples taken by the glider and the reconstructed temperature field. During the active navigation segments of the August trial, the NAV system autonomously generated 10 waypoints while underway, demonstrating that the NAV system can be effectively used in the field.

VII. Conclusion

In this paper we presented our NAV system to enable onboard autonomy for an underwater glider. By embedding a small, power-efficient computer in the glider, we are able to avoid issues relating to latencies in planning caused by having to pre-plan for a glider before it surfaces as well as risks presented by holding the glider at the surface while a plan is generated. We demonstrated the capability of the NAV system in a pair of sea trials off the Oregon Coast, where we examined both its ability to perform adaptive sampling via autonomous navigation, as well as characterized the cost of the onboard NAV system in terms of operational duration of the glider.

Embedding autonomy within oceangoing gliders is the first step towards building a fleet of autonomous sampling vehicles. As future work, we are interested in leveraging the benefits of

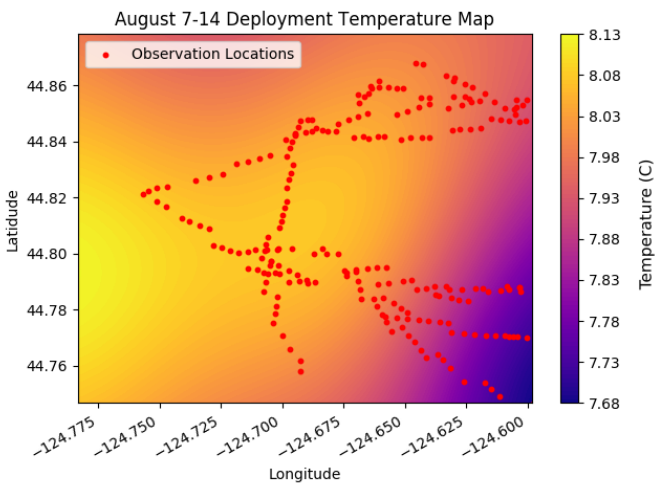

Fig. 7: Final temperature field estimate from August 2018 deployment. Temperature samples used to train the GP are shown by red dots.

embedded autonomy to build a multi-robot team capable of using distributed autonomy algorithms to improve the speed, accuracy, and efficiency of autonomous sampling. This would allow us to move away from glider fleets being controlled at a single point, and toward a more flexible and robust form of autonomy.

References

[1] O. Schofield *et al.*, "Slocum gliders: robust and ready," *Journal of Field Robotics*, vol. 24, no. 6, pp. 473–485, 2007.

[2] Y. Zhang, *et al.*, "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.

[3] B. N. Seegers *et al.*, "Subsurface seeding of surface harmful algal blooms observed through the integration of autonomous gliders, moored environmental sample processors, and satellite remote sensing in Southern California," *Limnology and Oceanography*, vol. 60, no. 3, pp. 754–764, 2015.

[4] National Research Council, *Science at Sea: Meeting Future Oceanographic Goals with a Robust Academic Research Fleet*. National Academy Press, Washington, DC, 2009.

[5] K. Benoit-Bird *et al.*, "Equipping an underwater glider with a new echosounder to explore ocean ecosystems," *Limnology and Oceanography: Methods*, vol. 16, no. 11, pp. 734–749, 2018.

[6] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[7] S. McCammon and G. A. Hollinger, "Topological hotspot identification for informative path planning with a marine robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018.

[8] E. Fiorelli *et al.*, "Multi-AUV control and adaptive sampling in Monterey Bay," *IEEE journal of oceanic engineering*, vol. 31, no. 4, pp. 935–948, 2006.

[9] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[10] N. K. Yilmaz *et al.*, "Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming," *Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 522–537, 2008.

[11] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, Minnesota, 2012.

[12] J. Binney, A. Krause, and G. S. Sukhatme, "Informative path planning for an autonomous underwater vehicle," in *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, 2010.

[13] K.-C. Ma *et al.*, "Data-driven learning and planning for environmental sampling," *Journal of Field Robotics*, vol. 35, no. 5, pp. 643–661, 2018.

[14] J. Das *et al.*, "Simultaneous tracking and sampling of dynamic oceanographic features with autonomous underwater vehicles and Lagrangian drifters," in *Experimental Robotics*. Springer, 2014, pp. 541–555.

[15] N. E. Leonard *et al.*, "Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in Monterey Bay," *Journal of Field Robotics*, vol. 27, no. 6, pp. 718–740, 2010.

[16] A. S. Ferreira *et al.*, "Advancing multi-vehicle deployments in oceanographic field experiments," *Autonomous Robots*, vol. 43, no. 6, pp. 1555–1574, Aug 2019.

[17] G. Loianno *et al.*, "Autonomous flight and cooperative control for reconstruction using aerial robots powered by smartphones," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1341–1358, 2018.

[18] A. J. Smith *et al.*, "Real-time distributed non-myopic task selection for heterogeneous robotic teams," *Autonomous Robots*, vol. 43, no. 3, pp. 789–811, 2019.

[19] K. Mohta *et al.*, "Fast, autonomous flight in GPS-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[20] M. Quigley *et al.*, "ROS: an open-source robot operating system," in *IEEE International Conference on Robotics and Automation (ICRA), Workshop on Open Source Software*, Kobe, Japan, 2009.

[21] M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard, "Nested autonomy for unmanned marine vehicles with MOOS-IvP," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, 2010.

[22] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.