**C-SLAM: Adopting the SLAM Algorithm for Underwater Robotic**

Lucía Martínez Ruiz, Robert D’Antonio, Xinglin He, Zhaowen Gu, and Lydia Jacobs-Skolik

**Abstract**—Since the current methods for precise underwater navigation are either insufficient in terms of accuracy or environmentally unfriendly, C-SLAM aims to address the critical issue of underwater localization for Uncrewed Underwater Vehicles (UUVs) undertaking seabed surveys. To improve the accuracy of navigation and overall operational efficiency, we'll use Simultaneous Localization and Mapping (SLAM) techniques, which can achieve precise localization by looking at matching landmarks and Inertial Navigation System (INS) measurements from sonar data. While processing these sonar data, C-SLAM will adjust position estimation errors through loop corrections and pose-graph optimization methods. As a result, it can generate a detailed 2D/3D visual representation of the vehicle’s trajectory in the seafloor environment. Considering the importance of exploring the seafloor for scientific research, resource identification, and environmental monitoring, it is essential to track the accurate trajectory of UUVs, and our final deliverable, C-SLAM, is dedicated to tackle this issue.

**Index Terms**—Graph Algorithms, Computational Geometry and Object Modeling, Computer vision, Navigation

————————————————

* *Lucía Martínez Ruiz is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail:* [*lmrpueyo@bu.edu*](mailto:lmrpueyo@bu.edu)
* *Robert D’Antonio is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail:* [*robdanto@bu.edu*](mailto:robdanto@bu.edu)
* *Xinglin He is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail:* [*hxl@bu.edu*](mailto:hxl@bu.edu)
* *Zhaowen Gu is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail:* [*petergu@bu.edu*](mailto:petergu@bu.edu)
* *Lydia Jacobs-Skolik is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail:* [*ccjs@bu.edu*](mailto:ccjs@bu.edu)

—————————— ◆ ——————————

# 1 Introduction (Lucía Martínez Ruiz)

The ocean's vast and largely unexplored terrain presents significant challenges for precise underwater navigation, particularly for Uncrewed Underwater Vehicles (UUVs) engaged in seabed surveys. Numerous offshore industries require accurate sonar surveys of the ocean floor. However, existing methods that depend on acoustic beacons, often discarded into the seafloor without retrieval, present constraints on both accuracy and environmental sustainability. Using a GPS is also an inefficient method because it’s unable to transmit signals underneath the ocean’s surface. This challenge of underwater navigation prompted our team to devise a solution that not only addresses the accuracy issue but is more environmentally sustainable.

The purpose of our project is to develop an implementation of the C-SLAM algorithm tailored specifically for subaquatic exploration. The core approach involves leveraging Simultaneous Localization and Mapping (SLAM) techniques to enhance underwater vehicle localization. By utilizing the survey data itself, the algorithm will accurately estimate the UUV's position based on correlated sightings of landmarks and geometric measurements from the sonar data. The proposed approach aims to provide a comprehensive solution for precise underwater navigation and mapping, reducing reliance on physical beacons and thus promoting a more sustainable subaquatic exploration method.

Despite the challenges posed by the underwater environment, the algorithm can identify and classify anomalies in sonar data as landmarks, resist the noise inherent in subsea data, and ensure a reliable and accurate mapping of the UUV's movements. Moreover, because the algorithm adapts to different levels of sonar data clarity, it can be applied to various underwater environments.

By eliminating the necessity for physical acoustic beacons, C-SLAM not only reduces ocean pollution but also stands as a more sustainable alternative compared to other methods. Additionally, the algorithm generates precise updated bathymetric charts and 2D/3D visual graphs of the seafloor environment, which helps us understand the underwater terrain more comprehensively.

Therefore, through this project, our team aims to optimize underwater navigation for UUVs, aiming to make subaquatic exploration more sustainable and precise, so that we can better understand and utilize the vast resources hidden beneath the ocean's surface.

# 2 Conceptual Development (Xinglin He)

## 2.1 Customer’s Problem

The problem of achieving accurate and sustainable navigation for UUVs on the ocean floor focused on the issues in which traditional methods such as physical acoustic beacons are both environmentally harmful and limited in accuracy. To address this problem, an alternative approach is needed to ensure accurate underwater navigation and mapping, addressing the drawbacks of the current methods while prioritizing environmental friendliness. The challenge at hand revolves around the precise localization of the vehicle through processing the collected sonar datasets from UUV and constructing an accurate 2D/3D visual representation of the seafloor environment without the need to place physical acoustic beacons.

## 2.2 Conceptual Approach

Compared to the visual datasets or sensor-fusion datasets with Inertial Measurement Unit (IMU) data that most SLAM algorithms commonly use to estimate the vehicle’s location above the seafloor, the side-scan sonar datasets collected from UUV present distinctive characteristics. These datasets include unique side-scan images and Inertial Navigation System (INS) data, comprising latitude, longitude, depth, heading, pitch, and roll values. During the pre-processing stage, the dead-reckoning method could be a potential mathematical approach to compute the vehicle’s position relative to the seafloor at each time step, which involves the integration of the INS data and previously predicted positions. Also, C-SLAM should be capable of identifying and extracting correct landmarks from the side-scan images, while matching the key points between paired side-scan images. Following the image processing stage, the identified landmarks can be used in optimizing the pose-graph of the vehicle through loop correction based on landmark correspondence at different time steps. Then, the vehicle’s trajectory will be refined by reconciling the observed landmarks with those detected at various points in time. After updating every position of the vehicle on the pose graph through the loop correction process, a comprehensive 2D/3D visual map should form. This map can serve as a visual representation of the refined trajectory of the UUV during its exploration of the underwater environment. It stands as the primary final deliverable we aim to achieve by the end of this project.

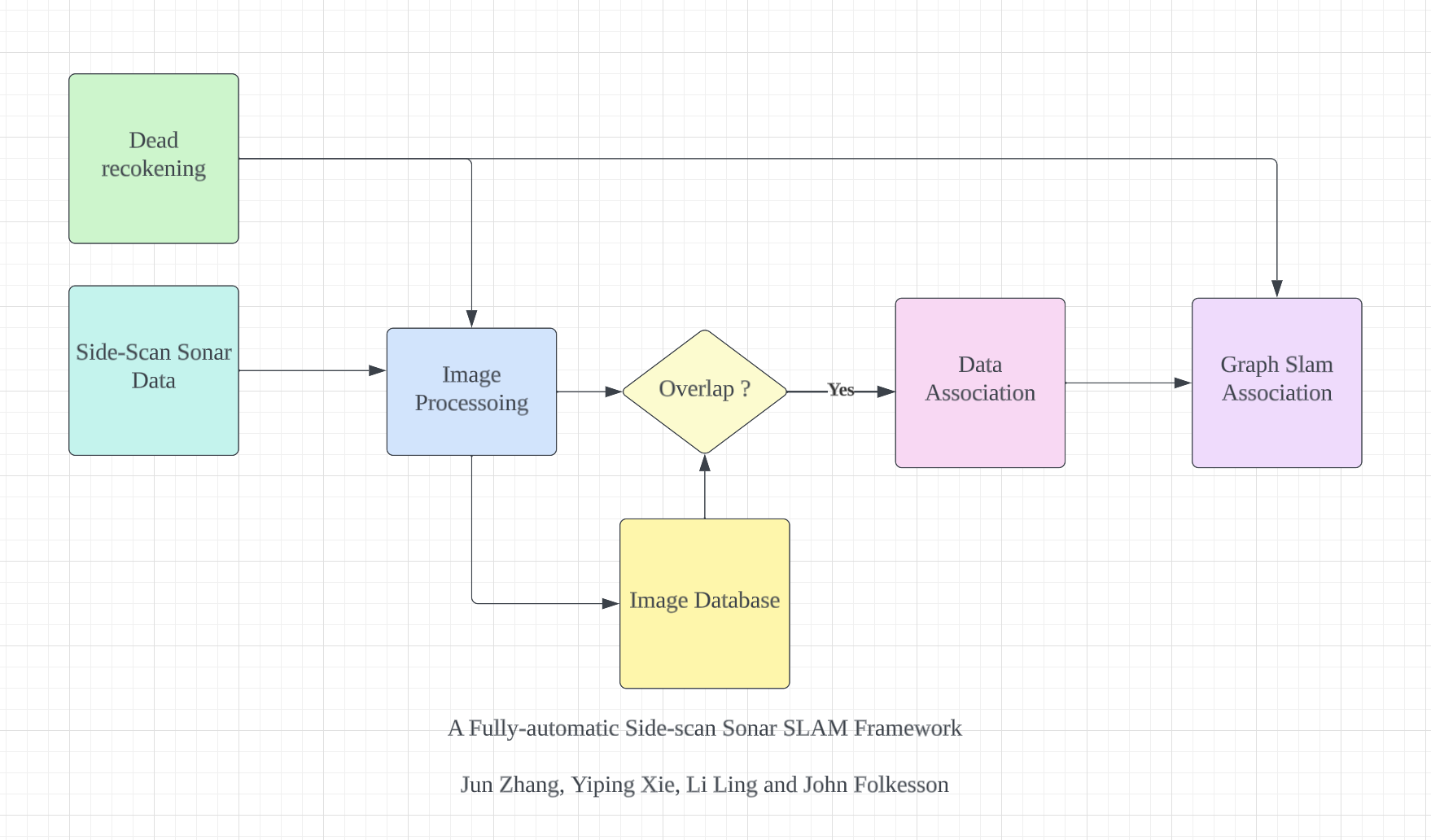
## 2.3 Alternative Solutions

The two alternative solutions we considered are ORB-SLAM3 and LSD-SLAM. ORB-SLAM3 [1] is an advanced system that can perform either visual, visual-inertial, or multi-map SLAM with various types of cameras. Its initialization process relies heavily on Maximum-a-Posteriori (MAP) estimation, which can effectively estimate the true scale of the scene being reconstructed based on the visual and inertial sensor uncertainties. When an active map is initialized, the tracking thread focuses on localizing incoming frames, while the local mapping thread continually enhances and extends the map by incorporating new keyframes. Subsequently, the loop correction from the loop and map merging thread were used to minimize the errors in these keyframes to create a final map. In addition, by incorporating a multi-map system, ORB-SLAM3 can relocalize itself when it experiences tracking losses. Although the sonar dataset collected from UUV cannot be directly input into ORB-SLAM3 to generate an active map due to the unique characteristics of its INS data, it is still an accurate, effective SLAM algorithm that can serve as a strong reference to guide us in the development of C-SLAM.

LSD-SLAM [2] is a direct monocular SLAM algorithm that can reconstruct the 3D environment in real-time. In contrast to ORB-SLAM3, it uses filtering-based depth map estimation to update the current keyframe in the pose graph. The advantage of LSD-SLAM is its flexibility in accurately estimating both fine details and large-scale geometry through scale-aware formulation, which can detect scale drift to minimize alignment error. However, LSD-SLAM demonstrated lower accuracy and generalizability compared to ORB-SLAM3 [1], so we have shifted our focus away from LSD-SLAM, no longer considering it as our primary reference.

Please find the detailed engineering requirements listed in Appendix 1. These requirements specify the parameters that our C-SLAM algorithm needs to meet to successfully address the problem.

# 3 System Descriptions (Zhaowen Gu)



*Figure 1. Block diagram of SLAM algorithm.*

In a typical SLAM algorithm, six distinct sessions are involved, as outlined above. Our process involves processing images utilizing side-scan sonar data and comparing it with the global map. Subsequently, we examine whether the local image's landmarks align with any global landmarks. If there's a match, we link the global and local landmarks; otherwise, the algorithm progresses.

## 3.1 Python Simulation

We've developed a simplified Python simulator to showcase fundamental SLAM concepts, comprising the same six sessions but in a more streamlined manner. This simulator will serve as a metric to gauge errors once the final SLAM algorithm is complete.

At its core, the algorithm operates on the premise that each movement of the vehicle triggers the SLAM algorithm to estimate landmark poses relative to itself. To mimic a realistic sea environment, we introduce adjustable Gaussian Error.

## 3.1.1 Robot Class

First, we'll be localizing a robot in a 2D grid world. The basis for simultaneous localization and mapping (SLAM) is to gather information from a robot's sensors and motions over time, and then use information about measurements and motion to reconstruct a map of the world. The robot lives in 2D, x-y space, and its motion is initially pointed in a random direction. It moves in a straight line until it comes close to a wall at which point it stops. For measurements, it senses the x- and y-distance to landmarks. This is different from range and bearing as commonly studied in the literature, but this makes it much easier to implement the essentials of SLAM without cluttered math. Each movement is subjected to Gaussian noise for realism.

Calculating movements:

| x = self.x + dx + np.random.normal() \* self.motion\_noise  y = self.y + dy + np.random.normal() \* self.motion\_noise |
| --- |

The sensing function returns x- and y- distances to landmarks within the visibility range because not all landmarks may be in this range, the list of measurements is of variable length. For each landmark, the SLAM algorithm:

1. Computes dx and dy, the Euclidean distances between the robot and landmark.
2. Account for measurement noise by \*adding\* a noise component to dx and dy. The noise component should be a random value between [-1.0, 1.0)\*measurement noise.
3. If the landmark falls outside of the measurement range, then not recorded. If the landmark falls inside of the measurement range, then record.Last but not least, I subtract the estimated landmark poses and compare them with the ground truth(my input) using Euclidean distance and calculate the mean of error.

Measuring landmarks:

| # compute dx and dy  dx = landmark[0] - self.x  dy = landmark[1] - self.y  # adding noise  noise=self.rand()\*self.measurement\_noise  dx += noise  dy += noise  # check if out of range  if(abs(dx) < self.measurement\_range and abs(dy) < self.measurement\_range):  measurements.append([index, dx, dy]) |
| --- |

## 3.1.2 Data Processing

For the SLAM process, our procedure involves gathering a sequential set of robot sensor measurements followed by its motions over a specified timeframe. Solely relying on this gathered data, we aim to reconstruct the map of the world while pinpointing the robot and landmark locations. The input data takes the form of a structured list denoted as 'data'. This array is designed to hold sensor measurements and movements in a precise order: [measurements, [dx, dy], [time steps]]. This structured format proves instrumental in extracting the necessary data and constructing constraint matrices and vectors.

An instance of data:

[[[[0, 4.058747232257613, -2.0202536331868917],

[1, -2.110767012495474, 0.8102321220600216],

[2, 3.175840190016389, -0.9031606754281155]],

[1, 2]]]

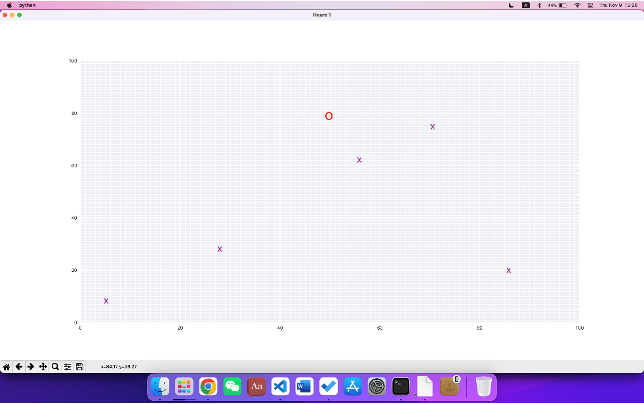
Whereas the first vector indicates the landmarks within the detection range: robot senses landmark 0 at (4.0587,-2.0202) distance away relative to itself; it senses landmark 1 at (-2.1107,0.81023) distance away relative itself; it also senses landmark 2 at (3.1758,-0.90316) distance away relative itself; the last [1,2] vector indicates at time 0, it moves 1 to the right and 2 to the top. Time is implied to be 0 in this case.

## 3.1.2 Creating a world

Then we create a world and specify the hyperparameters:

| # world parameters  num\_landmarks=5 #number of landmarks  N = 20 #time steps  world\_size=100.0 #size of world (square)  # robot parameters  measurement\_range=50.0 #range at which  we can sense landmarks  motion\_noise= 2.0 #noise in robot motion  measurement\_noise= 2.0 #noise in the  measurements  distance= 20.0 #distance by which  robot (intends to) move each iteration |
| --- |

The world can be printed for inference with NumPy.

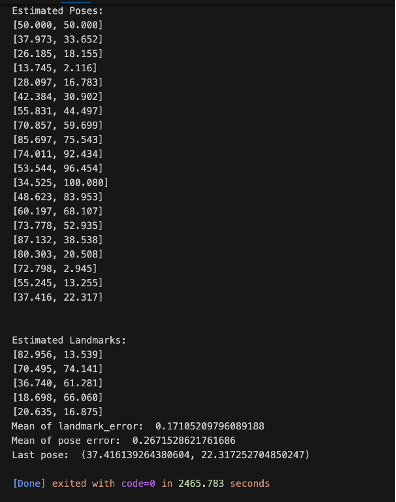


*Figure 2. Python simulator. A virtual seabed grid with 100\*100 size. The red circle is the location of the robot and Xs are the landmarks.*

## 

## 3.1.2 Testing and Evaluation

To ensure the versatility of our SLAM across different scenarios and document any potential errors, we plan to input test datasets and compute the mean square error as a metric. Ground truth data, manually fed into the algorithm, serve as our reference against which we compare the detected, noisy data version.



*Figure 3. Estimated poses after each movement and the estimated landmark location relative to the robot at the end of all movements. The mean of landmark error, mean of pose error, and last pose are shown.*

This straightforward yet highly effective approach greatly aids in comprehending the SLAM algorithm within an ideal, controllable environment. Customizing parameters like measurement range, measurement error, motion noise, and measurement noise allows for tailored adjustments. We intend to use this method for comparative analysis with our eventual true SLAM algorithm. By replicating the environment with zero measurements and motion noise, we aim to contrast our SLAM algorithm with this simplified version. This streamlined setup will serve as a valuable unit test or benchmark to evaluate our performance.

# 4 First Semester Progress (Lucía Martínez Ruiz & Robert D’Antonio)

Over this semester, our team has made significant progress in advancing our project goals. First, we identified and utilized the HoloOcean simulator, which allows us to gather sonar data from simulated underwater environments of our design. Having a tool like HoloOcean is crucial for developing, testing, and refining our Simultaneous Localization and Mapping (SLAM) algorithm.

In addition to HoloOcean, our team successfully located a real-world dataset containing side-scan sonar from an underwater survey in the Bassurelle Sandbanks of the English Channel. This dataset, made publicly available by the European Union, has proven instrumental in our development of data preprocessing functionality. Furthermore, we utilized this dataset to first help visualize what real-world side scan sonar looks like, which has been tremendously helpful as we work on methods for landmark classification.

To further enhance our project, we conducted and demonstrated three tests to validate three distinct portions of our project:

1. The first test involved testing the HoloOcean environment. Here, a direct comparison between real-world side-scan sonar data and the data generated by HoloOcean was performed with the oversight of our client. It was determined that the data generated by HoloOcean is realistic and reliable enough to be suitable for our purposes. Thus, the results strongly support using HoloOcean as one of our primary data sources for this project..
2. The second test was focused on testing the Basic SLAM Algorithm that we developed. This algorithm is of great significance also because it does not involve any sophisticated image processing or math. It works for all circumstances given the observation and parameters. It applies to our future SLAM algorithm as well and we believe it is the foundation of the algorithm.
   1. Test 1 exhibited excellent performance, with a mean pose error of 0.23% and a mean landmark error of 0.15%, categorizing it as perfect and consistently good, respectively.
   2. Test 2 results showed a 0.24% mean pose error and a 0.14% mean landmark error, falling within the range of consistently good as well.
3. The last test involved ORB3-SLAM, known for its proficiency in creating an accurate map view through processing images and IMU data from micro aerial vehicles (MAVs). Despite the uniqueness of our sonar dataset, which only contains INS data instead of IMU data, we see potential in ORB-SLAM3 as a guiding benchmark for our own SLAM algorithm development. While there may be challenges in directly feeding our sonar data into ORB-SLAM3, we remain committed to exploring its content and logic.

We acknowledge the challenge of adapting ORB-SLAM3 to our unique dataset and are mindful of potential accuracy issues due to the absence of IMU data. Nevertheless, we are dedicated to addressing this challenge by testing only the images created by sonar data with ORB-SLAM3 and exploring alternative methods to replace IMU data with INS data. Our goal is to tailor the algorithm to meet our specific requirements, achieving a successful SLAM algorithm for our sonar dataset that mirrors or surpasses the efficiency and accuracy demonstrated by ORB-SLAM3. This strategic move reflects our commitment to a comprehensive evaluation of the algorithm's efficacy in a real-world context.

# 5 Technical Plan (Robert D’Antonio, Zhaowen Gu & Xinglin He)

**Task 1. HoloOcean Simulations**

We are working on the development of a set of Python scripts dedicated to creating data using the HoloOcean simulator. These scripts will facilitate the simulation of UUVs mapping diverse ocean environments, each with varying conditions. This data is crucial for both training and validating our SLAM implementation. Specifically, we aim to create a minimum of three distinct simulations, each representing different underwater environments. The first should represent a substantially cluttered seabed, from which many landmarks could be identified. The second should be a sparsely populated seabed, where we expect the same landmarks to be classified regardless of the UUV’s route. And the third should be a sparse seabed, in which our algorithm should struggle to pick out reasonable landmarks. For our purposes, each of these test environments must span 100 square meters at a minimum. *Lead: Robert D'Antonio*

**Task 2. Unpacking Real-World Data**

Our focus involves preprocessing datasets containing real-world bathymetric data to ensure compatibility with our SLAM implementation. The current dataset comprises .sds files, necessitating unpacking and stitching to render them usable. *Lead: Xinglin He*

**Task 3. Preprocessing Real-World Data**

Generally sonar data packets contain not just sonar data, but readings from the INS and other sensors at the moment the sonar ping was received. This additional data is used for the crucial preprocessing step, which involves transforming sonar data into a standardized coordinate system. For this project, we use the North East Down (NED) coordinate system as our standard. Also in this preprocessing step is the cleansing of the data, in which we account for the pitch, roll, heading, and depth of the UUV. There is a significant routine of trigonometry and linear algebra to integrate these four variables into each data point, which is covered at a high level in the appendix. *Lead: Robert D'Antonio*

**Task 4. Feature Detection and Extraction**

There should be a thread that can correctly identify the landmarks from every side-scan image, where a potential landmark can exist near the shadowed region because it’s not flattened. Constraints also need to be applied during landmark detection to minimize errors. Then, extract that information from every landmark and keep it in the global map for loop correction. *Lead: Lydia Jacobs-Skolik*

**Task 5. Graph Optimization**

A thread for loop correction and pose-graph optimization needs to be implemented to improve the accuracy of the vehicle’s position estimation. It should be able to adjust the trajectory of the vehicle on the map by incorporating any drifting errors from loop correction. *Lead: Xinglin He, Lucía Martínez Ruiz*

**Task 6. SLAM Algorithm Implementation**

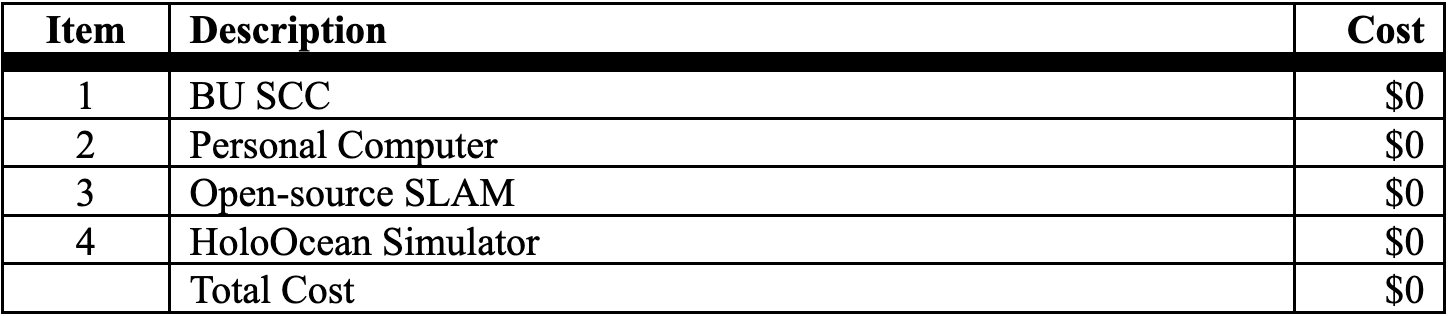
Incorporating all the previous threads into a SLAM framework for optimal performance in underwater environments is our primary goal. This implementation will be executed offline, utilizing data acquired from a UUV post-expedition, as opposed to real-time processing on the UUV's hardware. In the end, a final 2D/3D visual graph should form to visualize the trajectory of the vehicle. Also, an essential metric for evaluation involves measuring the algorithm's mean error, benchmarked against a baseline SLAM algorithm devoid of noise. Our target is an anticipated mean error of less than 10%. *Lead: Zhaowen Gu*

**Task 7. Benchmarking**

Establishing a comprehensive suite of benchmarks stands pivotal for evaluating the efficacy of various implementations. These benchmarks will be derived either from the HoloOcean simulator or real-world datasets, providing a comparative framework. *Lead: Lucía Martínez Ruiz, Lydia Jacobs-Skolik*

# 6 Budget Estimate (Zhaowen Gu)

## 6.1 Figures and Tables

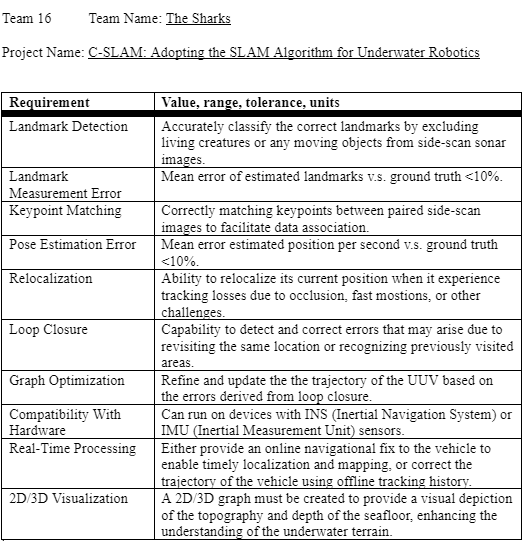


*Table 1. Budget Estimates*

Given that our project is entirely software-based, our budget is currently at $0. While we efficiently collaborate with BU SCC and utilize our personal computers, having a PC equipped with a high-performance GPU would be beneficial, particularly for seamless compatibility with the HoloOcean Simulation. BU’s support in acquiring this resource would be greatly appreciated.

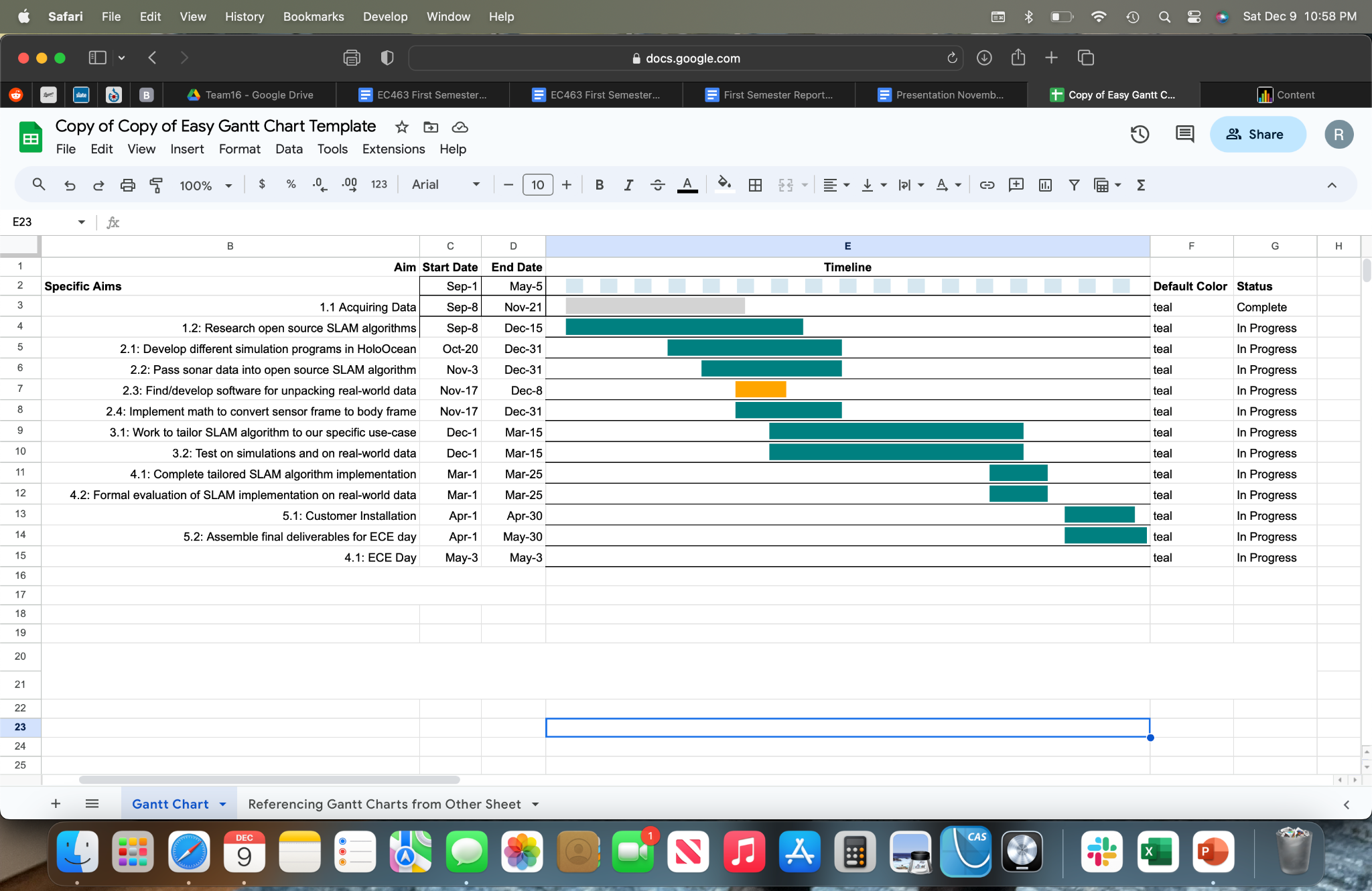
# 7 Appendix

## Appendix 1 - Engineering Requirements (Zhaowen Gu & Xinglin He)



*Table 2. Engineering Requirements*

## 7.2 Appendix 2 - Gantt Chart (Robert D’Antonio)



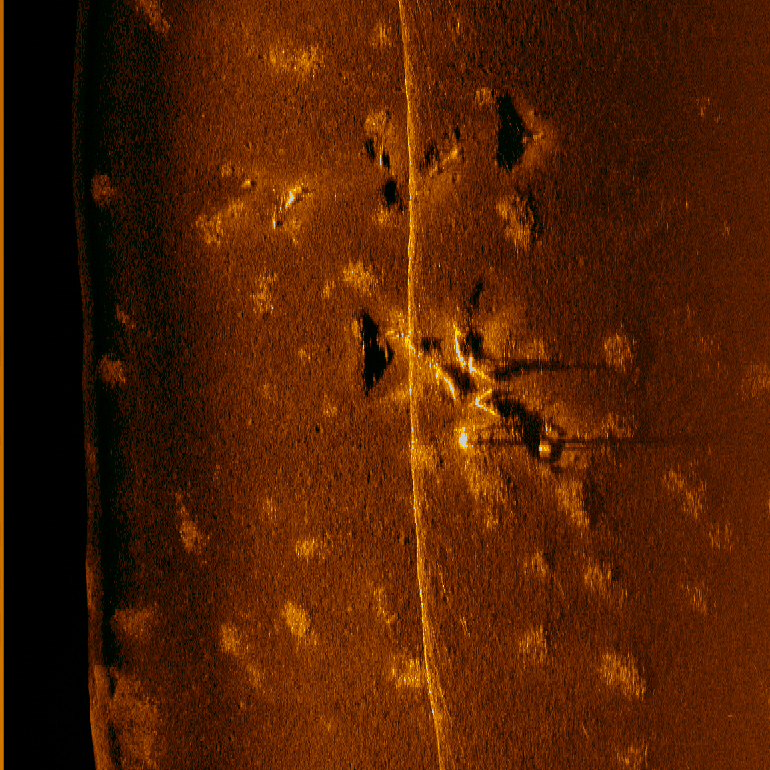
*Figure 4. Gantt Chart*

We are currently on track to complete our project on time.

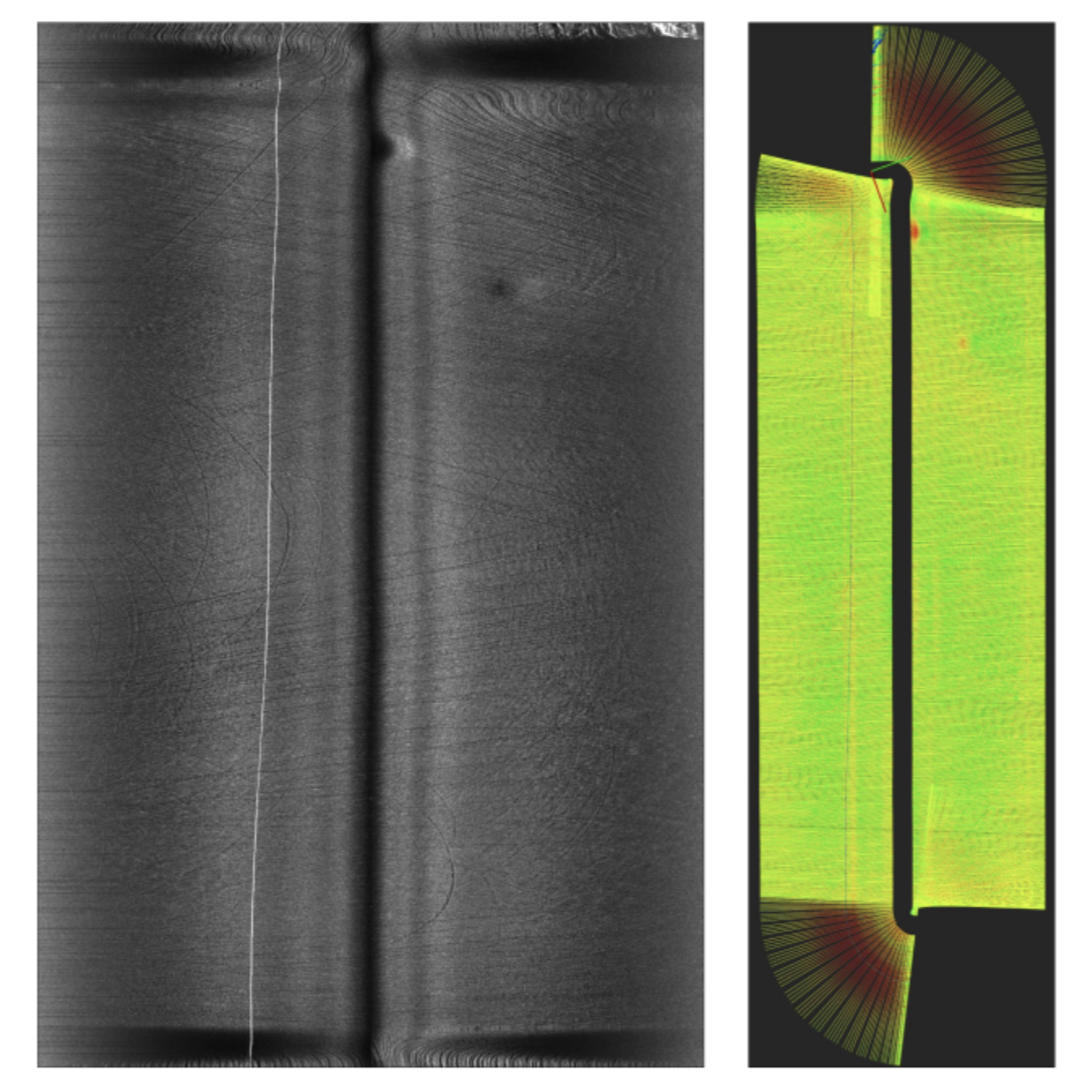
## 7.3 Appendix 3 – Github Repository (Zhaowen Gu)

<https://github.com/peterguzw0927/Senior_Design.git>

**7.4 Appendix 4 – Complementary Images (Zhaowen Gu)**



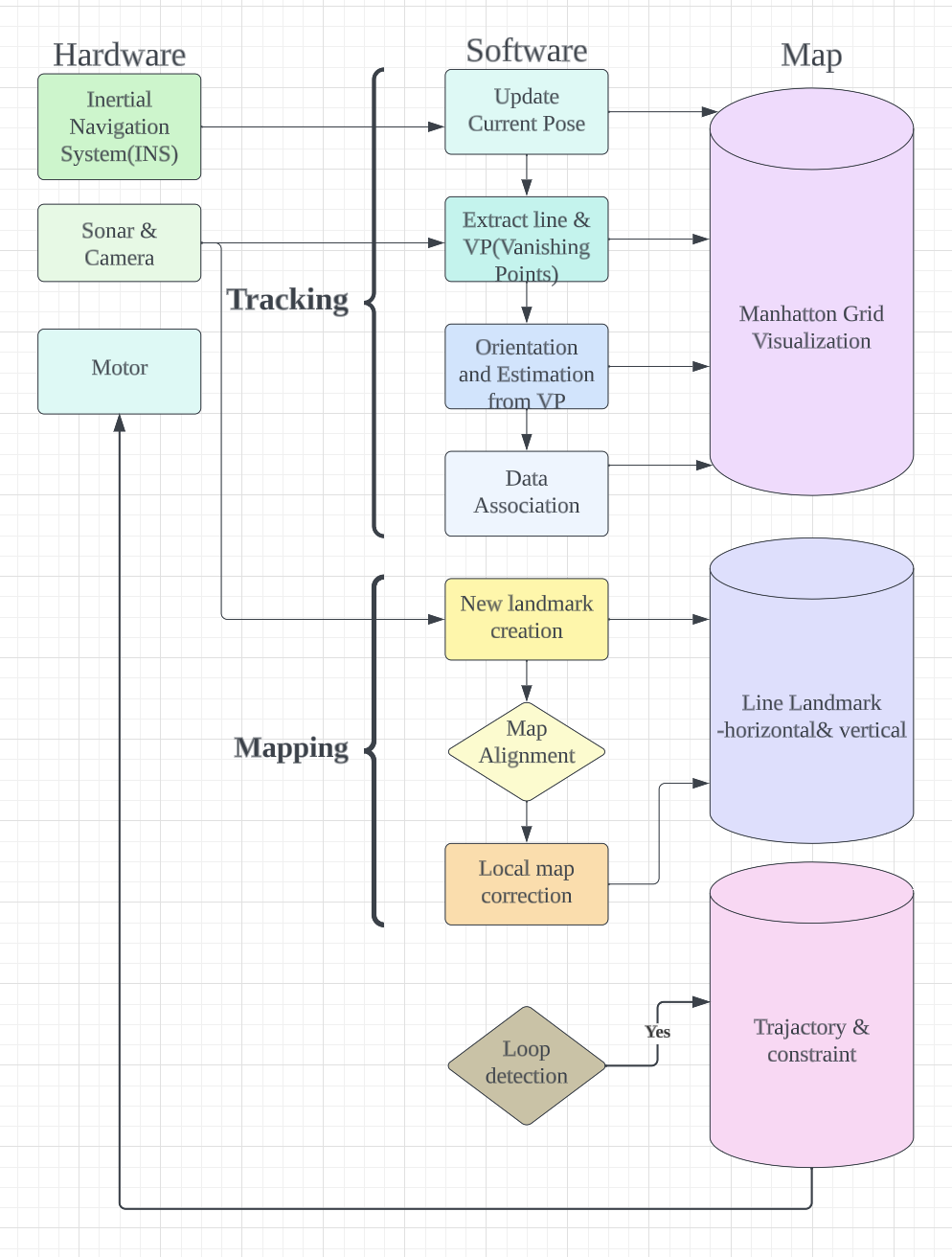
*Figure 5. Side-scan Sonar data depicting a WWII-era torpedo dump off of the island of Palau. Image courtesy of Scripps’ Eric Terrill.*



*Figure 6. Left: sample of our collected side-scan sonar images after processing, where trawling marks can be observed on the seabed. Right: the corresponding geo-referenced image storing pixel positions relative to the AUV poses. Note that the color encodes pixel intensity that is best viewed in the left image.*

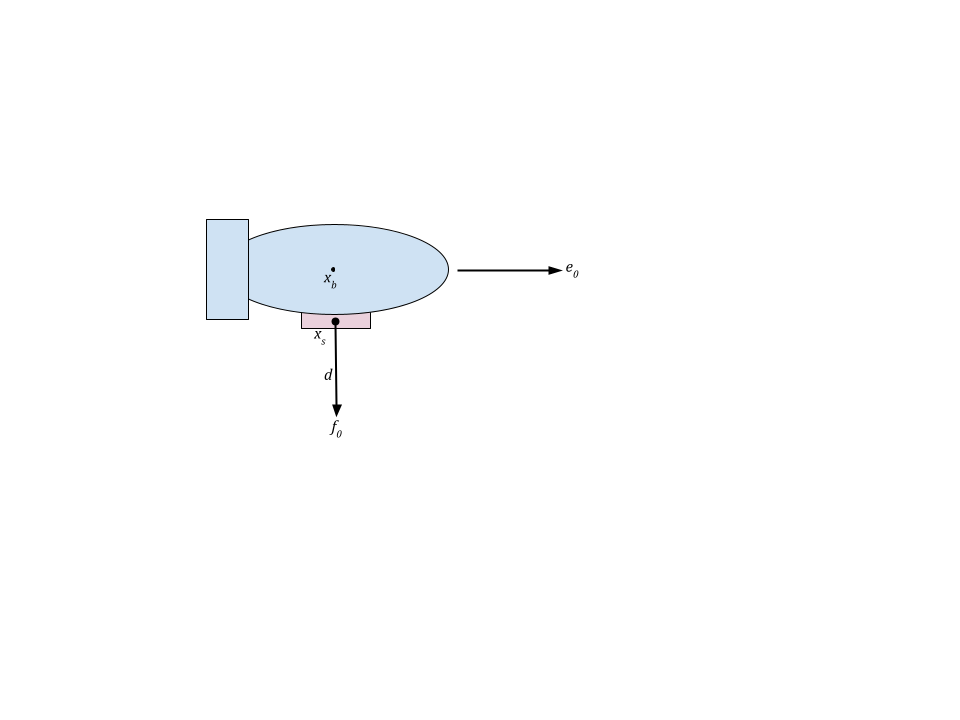


*Figure 7. Example of keypoint correspondences generated by proposed algorithm (left) and manual annotations (right).*



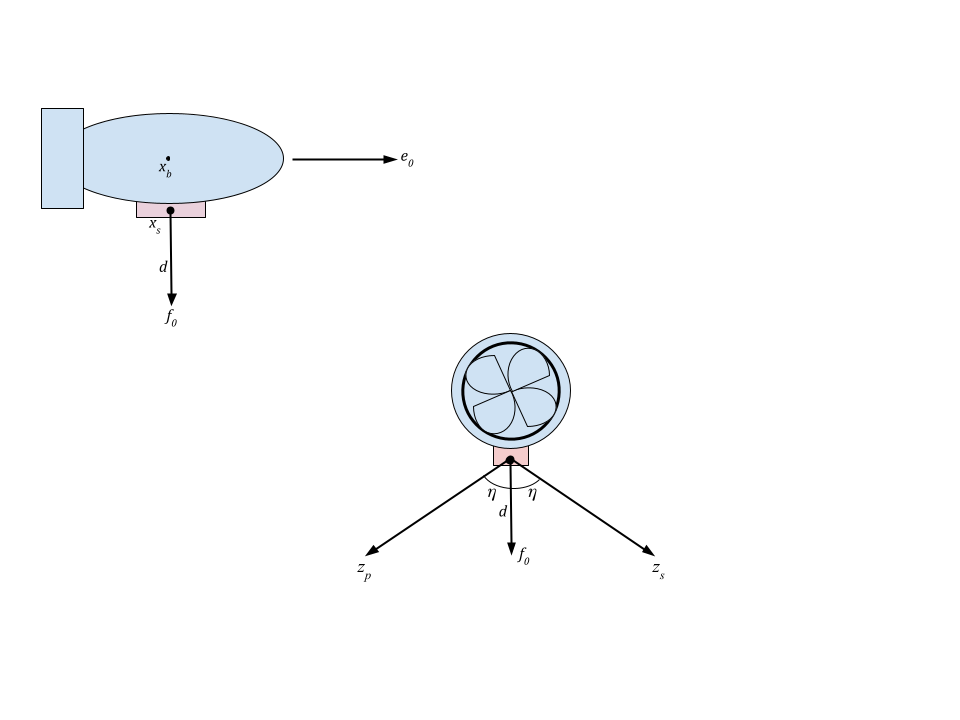
*Figure 8. A more sophisticated SLAM algorithm block diagram.*

**7.5 Appendix 5 – Preprocessing of Sonar Data, High-Level Overview (Robert D’Antonio)**



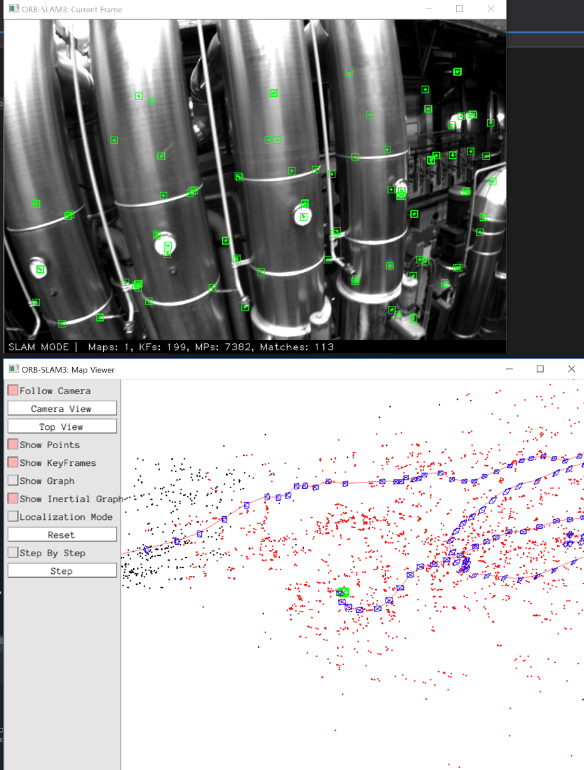
*Figure 9: Side view of underwater robot*

Preprocessing our data is a crucial step in developing a well-performing implementation of the SLAM algorithm. Without this step, it is difficult to get any usable information out of real-world data. In this project, we are first going to focus on side-scan sonar, as most of the data we have been using is captured with side-scan hardware (compared with forward look or profiling sonar).



*Figure 10: Rear view of an underwater robot, where zp and zs represent the endpoints of the port and starboard sonar pings, respectively*

Side-scan sonar utilizes two separate sonar modules, with one pointing port-side and the other starboard. Each of these modules will scan an area starting from a fixed distance from center-ground point  *f0* and terminating at its respective point *z*. This point *z* is directly linked to the module’s fixed (constant) mounting angle *η.* Each point *z* collected at time *t* is also directly related to and taken with respect to the vessel’s heading *ψ*, pitch *θ*, heading *φ*, and depth *d* at that time *t*. The primary preprocessing step is to calculate from this data new points  *ap* and  *as*, where *a* is the endpoints of the sonar pings in the NED plane at time *t*. Also calculated here is the vessel’s body representative *xb*, the coordinate of the vessel in the NED plane. With these calculations, data from time *t* can now be correctly compared and used with data from any time *t’*.

**7.6 Appendix 6 – ORB-SLAM3 Simulation (Xinglin He)**

*Figure 11: A map viewer appears alongside the video captured by the camera showing the trajectory of the camera’s movement.*

# 8 References

[1] Campos, C., Elvira, R., Rodriguez, J. J. G., M. Montiel, J. M., & D. Tardos, J. (2021). ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. IEEE Transactions on Robotics, 37(6), 1–17. <https://doi.org/10.1109/tro.2021.3075644>

[2] Engel, J., Schöps, T., & Cremers, D. (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, Vol 8690 (pp. 834–849). Springer. <https://doi.org/10.1007/978-3-319-10605-2_54>

[3] Zhang, Jun, et al. A Fully-Automatic Side-Scan Sonar SLAM Framework, arxiv, 4 Apr. 2023, [https://arxiv.org/abs/2304.0185](https://arxiv.org/abs/2304.01854)