# SITUATIONAL AWARENESS FOR LOW COST UNDERWATER AUTONOMY

by

John McConnell

A DISSERTATION

Submitted to the Faculty of the Stevens Institute of Technology
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

______________________________
John McConnell, Candidate

STEVENS INSTITUTE OF TECHNOLOGY
Castle Point on Hudson
Hoboken, NJ 07030
2023

# SITUATIONAL AWARENESS FOR LOW COST UNDERWATER AUTONOMY

## ABSTRACT

Underwater robots have matured over the past few decades to support the increasing need for maritime infrastructure assessment and other offshore activity. However, deploying underwater robots often involves a high startup cost associated with vehicle acquisition. It is often the case that companies not engaged in oil and gas exploration are priced out of this technology. This is primarily due to the highly accurate, often tactical-grade inertial navigation systems (INS) required to make state estimation reliable enough for long-duration autonomous operations. Moreover, sonars are the perceptual sensor of choice due to the frequent need to operate in high-turbidity water. Sonars, however, do not provide the dense 3D information required for offshore asset evaluation or autonomous navigation. In this work, we consider an underwater robot's situational awareness, i.e., the required knowledge of a robot's environment to complete its mission safely.

We propose several sub-systems that assist in bridging the situational awareness gap in low-cost underwater autonomous systems equipped with imaging sonars. Firstly, we employ a stereo pair of orthogonal sonars to recover lost 3D data. Second, we generalize the method of orthogonal sensor fusion to work in large-scale environments through object-level inference. Next, we use public domain prior information to enhance the state estimation capabilities of a low-cost underwater robot. We then use submaps to recover 3D maps, providing dense 3D maps without object-level inference. Lastly, we extend our thinking to the multi-robot case. We showcase a system that performs multi-robot simultaneous localization and mapping (SLAM) using real-world underwater imaging sonar data, the first of its kind. In future work, this will enable multi-robot systems that are capable of safely operating in marine environments and cooperatively completing their tasks efficiently.

Author: John McConnell

Advisor: Brendan Englot

Date: May 1st 2023

Department: Mechanical Engineering

Degree: Doctor of Philosophy

Per Aspera Ad Astra

## Acknowledgments

**Funding Acknowledgement:** This thesis was supported in part by the National Science Foundation (NSF), grant numbers IIS-1652064 and IIS-1723996, the Office of Naval Research (ONR), grant N00014-21-1-2161, and by a grant from Schlumberger Technology Corporation.

# Table of Contents

# Chapter 1
# Introduction

## 1.1 Problem Space

Underwater autonomy has emerged as a critical task serving markets ranging from the intelligence community to the support of growing offshore infrastructure. However, autonomous underwater vehicles (AUVs) are often too expensive for companies not engaged in oil and gas exploration. This means that smaller firms, such as civil engineering companies or users considering teams of AUVs cannot engage the capabilities of this technology. Moreover, when considering teams of robots working to complete tasks, if the robots are too expensive, it will simply not be possible to deploy the required number of agents. This work will focus on lower-cost AUVs, and the methods we have proposed to address their limitations.

AUVs operating in real-world conditions will experience low lighting, turbid water, and other challenging environmental factors. This means that when considering AUV perception, acoustic sensors become the tool of choice. When considering inertial navigation systems (INS), it is typical to use a Doppler velocity logger (DVL) for seafloor relative speed, combined with an inertial measurement unit (IMU) for attitude estimation. For perceptual systems, sonar is the tool of choice due to its robustness to the aforementioned environmental conditions. Sonar comes in several flavors, single beam scanning sonar is mechanically actuated to view the environment. Multi-beam sonar uses an array of sonar beams to image the environment. Multi-beam sonars are typically characterized by a non-zero aperture, which creates multiple returns at each bearing angle. One way of addressing this is to use profiling sonar, which has a narrow beam, reducing the multi-return effect across the vertical aperture. However, profiling sonars come at a notably higher cost than wide aperture sonars, making them unrealistic for deployment in this context. Moreover, profiling sonars only view a single slice of the environment. When compared to wide aperture imaging sonars that image an entire volume of the environment, they provide much lower situational awareness.

This yields a typical sensor payload of a DVL, IMU, and a wide aperture imaging sonar. The total cost of this payload is on the order of five figures (USD). While expensive, consider some of the recent work in the underwater space using vehicles that cost an order of magnitude more [1]. Our INS system, consisting of a DVL and IMU, has a high drift rate, meaning it's reliability for state estimation is often 1 minute or less. Thus, drift correction using the wide aperture multi-beam sonar is required, classically known as simultaneous localization and mapping (SLAM). SLAM, however, is challenging with the level of noise and ambiguity present in sonar imagery. Moreover, while sonar images a 3D volume of space, it only reports two of the three quantities in spherical coordinates. The sonar image contains range and bearing but cannot measure elevation angle. Further, because of this lack of elevation angle 3D mapping is a challenging problem.

In this work, we will address the issue of situational awareness for low-cost AUVs. Specifically, we will consider the 3D mapping problem using a 3DOF SLAM-based state estimate. The contributions of this work are as follows:

- A methodology for recovering the missing dimension using an orthogonal array of multi-beam imaging sonars [2].
- An extension of the above using deep learning and Bayesian estimation to rapidly map large scale environments [3].

- A graph SLAM-based system that incorporates overhead imagery to correct drift throughout the mission [4].
- A multi-robot SLAM system for underwater robots using imaging sonar with simulated communications [5].
- An extension and comparison of previous 3D mapping work that retains data between discrete timesteps [6].

Open source code has been furnished and video visuals as follows:

- BlueROV package: https://github.com/jake3991/Argonaut
- Sonar based SLAM: https://github.com/jake3991/sonar-SLAM
- Multi-robot Sonar SLAM: https://github.com/jake3991/DRACo-SLAM
- Orthogonal sonar fusion: https://github.com/jake3991/StereoFLS
- YouTube Channel: link

# Chapter 2

# Background

## 2.1 Wide Aperture Imaging Sonar

### 2.1.1 Sensing With Wide Aperture Imaging Sonar

An imaging sonar measures points in spherical coordinates by emitting acoustic pulses and measuring the associated intensity $\gamma \in \mathbb{R}_+$ from their returns. This information is organized into an *intensity image*, which we view as a set of range-angle-intensity vectors: $\mathbf{z} \in \mathbb{R}_+ \times [-\pi, \pi) \times \mathbb{R}_+$. While intensity image source data comes from three-dimensional observations, images only contain one angle: bearing. These observations are expressed in the robots own coordinate frame $\mathcal{R}$ and can be converted to Cartesian coordinates using equation 4.1.

$$P^{\mathcal{R}} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R \begin{pmatrix} \cos\phi\cos\theta \\ \cos\phi\sin\theta \\ \sin\phi \end{pmatrix}. \tag{2.1}$$

This means that when viewing sonar images, we are seeing a compressed version of the 3D space that the image came from. This is analogous to a monocular image, which does not contain depth, however in this case we are lacking a single angle in spherical coordinates. This means that any 3D observation will have to be inferred from the source image.

We would like to emphasize that this sensing modality is a necessary evil for AUVs operating in non-trivial environmental conditions. For all it's faults, multi-beam imaging sonar offers the AUV a sensor that is long range, robust to lighting and water quality; clearly superior in terms of situational awareness.

### 2.1.2 3D Reconstruction Using Wide Aperture Imaging Sonar

The challenges associated with wide-aperture multi-beam imaging sonars have inspired an impressive body of work to address the fundamental limitations of their under-constrained measurements. Firstly, work from Aykin [7, 8] estimates the elevation angle of sonar image pixels in scenes where objects are lying on the seafloor.

The recent work of Westman [9] extends the work of Aykin and shows excellent results in a constrained nearshore pier environment. However, these methods rely on several assumptions that may often be violated. Firstly, both [8] and [9] assume that all objects in view have their range returns monotonically increase or decrease with elevation angle. While this assumption may hold true for some objects, it hinders the application of their methods to arbitrary objects and scenes. Additionally, [8] requires the leading and trailing edge of an observed object; this is obtained by examining the shadow area behind a segmentation created by the sonar's downward grazing angle. In contrast, [9] only needs to identify the leading *or* trailing edge of the object; however, in the experiments shown, the leading edge is always the closest return because of the sonar's downward grazing angle. Using a downward grazing angle makes the problem significantly simpler to solve, but comes at a price. By tilting the sonar downward, making the upper edge of the sonar beam parallel with the water plane, the AUV's situational awareness may be hampered. In cluttered environments, an AUV would be unable to see above it before transiting upward, and a safe navigation solution may not always be possible. Moreover, if the vehicle is perturbed in a way that violates this geometric assumption, the perception system could be driven to inaccuracy.

We also note that a recent technique has employed deep learning with convolutional neural networks to estimate the elevation angle associated with imaging sonar observations [10]. However a vehicle may lack opportunities for prior training and exposure to the subsea objects and scenes it may encounter in a given mission.

Another method proposed by Aykin [11] applies a space carving approach to produce surface models from an image's low-intensity background, which outer-bound the objects of interest. The min-filtering voxel grid modeling approach from Guerneve [12] similarly removes voxels from an object model based on observations of low-intensity pixels. These approaches require the objects of interest to be observed from multiple vantage points to achieve accurate reconstructions.

A similar core issue can also be addressed from a simultaneous localization and mapping (SLAM) perspective. Rather than trying to estimate the elevation of pixels in a single frame, these works acquire features and use a series of views combined with a pose graph back-end [13] to determine 3D structure. This was proposed by Huang [14] in acoustic structure from motion (ASFM). This initial implementation has limitations, chief of which is the reliance on manually extracted features. This work was later built on by Wang [15], incorporating automated feature extraction and tracking. While these methods provide impressive results, they are focused on reconstructing the terrain under the vehicle, rather than providing adequate situational awareness around the vehicle. The limitation of these methods for AUVs in clutter is the perception system requiring a series of frames to recover 3D information, rather than a single timestep.

Lastly, much has been accomplished in the realm of stereo vision, and of specific relevance, computing the correspondences between two vantage points of the same scene. This concept is widely examined in an extensive body of literature [11-14]. Further, many feature extraction methods have been developed over the years, including SIFT [16], SURF [17], ORB [18] and KAZE [19]. Recently AKAZE [20] has shown promise in computing correspondences between acoustic images from a multi-beam imaging sonar. Westman [21] shows the utility of AKAZE features in a SLAM solution using acoustic imagery. Further, Wang [15] utilizes these same features in terrain reconstruction with acoustic imagery.

We also note that the specific concept of using two imaging sonars in stereo for 3D perception has been employed previously, but with relatively small differences in position and orientation between the sonars, and for reconstructing *sparse* sets of point features. The concept, first proposed by Assalih [22], was implemented and further analyzed by Negahdaripour [23], and used to build 3D maps of both sparse features extracted from a planar grid, and from small seafloor objects. Beyond its output of sparse features, this work stands in contrast to ours as the sensors used have aligned axes of uncertainty, rather than the orthogonal axes of uncertainty utilized in our work. A notable example of *dense* 3D mapping is the Sparus AUV's two orthogonally oriented single-beam mechanically scanning sonars (one imager and one profiler), used for cave mapping [24], [25]. However, the imager and profiler were used independently to address SLAM and 3D mapping, respectively, in separate steps.

## 2.2 Inference Aided 3D Reconstruction and Mapping

There is a large body of work that applies probabilistic inference to enhance 3D mapping and reconstruction. Most closely related are the methods that use inference to improve point cloud or voxel mapping, which often use insights from large data sets or object models provided a priori, which is not always available in an underwater robotics setting.

The use of a variational auto-encoder to infer the 3D distribution of an object given a single view from an RGB camera is explored in [26, 27]. As mentioned above, this work requires a large data set to pre-train the network weights. [28] explores the use of synthetic data from a generative adversarial network to avoid this requirement. [28] uses a single view voxel grid as the input for 3D reconstruction; the focus is 3D-to-3D densification, rather than the 2D-to-3D inference considered in our work. A similar concept to our work is explored in [29], where mapping is performed at an object level. Here, object models are produced using high-quality depth camera scans from a controlled setting. These scans are used to improve a 6-DoF SLAM solution. Similar to [26, 27], the applications of this approach in underwater robotics are limited due to the need for object models a priori. A notable use of inference in an underwater setting is [30], where CAD models of objects of interest are provided a priori, and are used to improve the map output.

A related set of methods use probabilistic inference to enhance occupancy grid maps [31, 32, 33, 34]. While these methods are excellent at improving occupancy map coverage, they solve mapping under sparse inputs by performing gap-filling and semantic inference on a data structure of the same dimensionality as those sparse inputs. In contrast, we focus on 2D-to-3D inference to enhance the dimensionality of inputs that are not directly observed in 3D, whose inference is conditioned on prior 3D observations of the same class.

## 2.3 Underwater SLAM with Sonar

Graph-based pose SLAM has been used to support many successful underwater sonar-based SLAM applications. Ship hull inspection using a forward-looking imaging sonar is demonstrated in [35], multibeam profiling sonar SLAM traversing an underwater canyon is achieved in [36], and planar SLAM using imaging sonar observations of the harbor seafloor is described in [37]. Additionally, [38] uses a factor graph to perform dense reconstruction of complex 3D structures using multibeam profiling sonar. Although underwater landmark-based SLAM with sonar has also been implemented successfully, [39, 21], the challenge of landmark SLAM with acoustic sensors is the identification of point features, or the identification of objects as landmarks, and their subsequent, repeated association as they are re-observed.

In the work that follows, we adopt pose SLAM for this reason. There is no need to solve the data association problem over landmarks, and more importantly, this implies that landmark identification is not required, and sonar observations not amenable to conversion into landmarks (vessel hulls, seawalls, etc.) is still useful in the SLAM problem.

While all of the above case studies are successful, [35], [37] and [21] rely on a ring laser gyroscope for highly accurate, low-drift heading information. The inertial package in [38] also shows highly accurate results as it is a simulation derivative of the INS in [35], [37], [21]. Lastly, [36] and [39] use SLAM to address missing, or noisy inertial information, showing improvement, but with drift still present.

While they represent impressive contributions to the state of the art, most of these works stand in contrast to our own, as they rely on expensive INS systems. In our work we focus on SLAM with a relatively low-cost vehicle, equipped with a Doppler velocity log (DVL) and a MEMS inertial measurement unit (IMU) for dead reckoning.

## 2.4 Fusion of Overhead Imagery and Ground Based Sensors

The fusion of overhead imagery with ground-based sensors has been widely explored. Several paradigms have been applied, including the use of feature association and the use of CNNs. Firstly [40], [41], fuse overhead images with ground-level perspectives by finding likely image pairs using image descriptors. These probable image pairs are then leveraged in a particle filter to localize a robot. The authors of [42], [43] and [44] utilize convolutional neural networks (CNNs) to compute image similarity. A more apt comparison to our work is [45], where live radar images are used to localize within a satellite image provided a priori, using deep learning to predict the rotation offset between images, and synthesize radar images to facilitate registration. In contrast, we use the output of overhead image registration to contribute factors to a SLAM factor graph, which is composed of a variety of measurement constraints from different sources.

Overhead image fusion has also been explored in underwater robotics, fusing sonar imagery with overhead images using a CNN [46]. However, like [42], [43] and [44], image similarity is the learned output, and direct registration is not addressed. [47] proposes using a CNN to learn the mapping between a sonar image and the companion overhead RGB image. This idea performs well on the authors' similar data, but the authors note performance limitations when generalizing; we believe this is likely due to the poor correlation between acoustic intensity and RGB values. Moreover, they do not consider the registration problem, of finding the transformation between a given satellite image footprint and the sonar image. In contrast to these works, we will use a CNN to learn a more general image representation, separately address the registration problem, and use these measurements to improve an existing graph-based pose SLAM system that also incorporates other sources of measurement.

## 2.5 Underwater Multi-Robot SLAM

In the underwater space, inter-robot constraints generally come in two forms [48]: *direct* encounters where robots observe one another via acoustic ranging, and *indirect* encounters, where robots observe the same targets in the environment and may derive inter-robot measurement constraints relating one another. We note that direct encounters require synchronized clocks, which may not be practical over long periods without GPS clock corrections due to clock drift.

We first consider [49], where a mobile base station is used to localize a team of low-cost vehicles lacking perceptual sensors using an acoustic beacon. A similar concept is considered in [50], except instead of a team of robots, a leader-follower arrangement is used. [51] considers inter-robot ranging without fixed acoustic beacons and performs a simulation study comparing the use of fixed beacons to a cooperative localization solution. [52] considers an algorithm for processing inter-robot acoustic pulses in a distributed manner. [53] proposes a pose-graph-based method for cooperative localization of a team of robots using dead-reckoning, GPS, and inter-robot ranging. The outcome of this work is a system where a robot maintains an understanding of its state, and the team's state. Moreover, when GPS measurements occur at the surface, their effect is shared across robots. [48] integrates perception into the above, and commonly observed features are shared and integrated into robot pose-graphs. [48] assumes that robots share point-landmark observations in their survey area, but the framework is only tested in simulation. Further, due to GPS, each robot is effectively localized in a common frame, enabling the sharing of range-bearing measurements to commonly observed targets without a need to solve the complex data association problem. Recall, GPS may not always be available due to under-ice operations or tactical situations (e.g., jamming). [54] considers the robot map merging problem using only similarity in feature space, in this case, ship hull curvature. While this work is able to merge trajectories and lower the data transmission requirements between robots, it requires a highly descriptive feature vector. It is only validated over a single dimension, whereas we consider a 3DOF system.

There are some notable examples of underwater multi-robot SLAM using cameras [55, 56, 57, 58] and cameras with other navigation sensors [59]. Cameras, however are not robust in all water conditions. While these works are informative, because we operate in turbid conditions with low visibility, we do not consider them further.

Another area to consider is the concept of multi-session SLAM, where a robot is provided with a prior SLAM run and merges the map it builds with the prior run as the mission progresses. This is examined for ship hull inspection [60, 61], bathymetric mapping [62], ship-wreck reconstruction [63] and environmental monitoring [64]. However, multi-session does not account for communicating information between robots, as it is a single agent system with prior information. Additionally, it may be the case that the robot has some knowledge of its location in the prior map, constraining the inter-robot loop closure search space.

In contrast to the above, we utilize no notion of an initial guess relating robot reference frames. Second, we consider the bandwidth limitations of real acoustic modem hardware and take steps to manage network utilization. Lastly, we implement a fully functioning system to detect and estimate indirect encounters in sonar data: inter-robot loop closures.

### 2.5.1 Place Recognition With Sonar

Place recognition (loop closure) is fundamental for a perception-driven multi-robot system. Place recognition has been widely studied in LIDAR sensing [65, 66, 67]. These works assemble a 2D bird's eye view image of a 3D LIDAR scan with coarse discretization to compare scenes. They also derive a 1D descriptor to support scene search and retrieval.

Place recognition has also been studied with sonar-equipped AUVs. Firstly, [68] considers building scene graphs to compare scenes and evaluate rigid-body transformations but requires at least fifteen objects in each scene to run. Machine learning has also been used to support this task in [69, 70]. However, few public sonar datasets exist, and this is often a research area unto itself. Iterative closest point (ICP) based loop closure is used in our prior work [71] to support single-agent active SLAM. This work uses sonar derived point clouds with ICP; when ICP provides a transform between keyframes, we estimate overlap between the point clouds. Point cloud overlap, then pairwise consistent measurement set maximization (PCM) [72], are used to reject outlier loop closures. Inliers are integrated into the graph-based pose SLAM solution. We note the extension of the PCM algorithm [61], but due to its additional complexity, we will not utilize it in our work.

### 2.5.2 Acoustic Modems

To communicate, AUVs generally use acoustic modems, which are low-bandwidth compared to in-air systems. One of the most well-known devices is the WHOI Micromodem [73], which can transfer a maximum of 5400 bits/s over a long-range. For shorter-range transmission (300 meters), higher bandwidth (62.5 kbits/s) units are available [74]. Moreover, recent research has demonstrated the feasibility of achieving even higher bandwidth in real-world conditions [75]. In our work, we consider bandwidth to be limited, and we will study the network utilization of our multi-robot system as a critical parameter. Further, we will take steps to minimize the transmission of large data structures.

# Chapter 3
# Fusing Concurrent Orthogonal Wide-aperture Sonar Images for Dense Underwater 3D Reconstruction

## 3.1 Data Association

This section studies methods for data association. We assume that the robot is equipped with two forward-looking acoustic sensors. The sensors are mounted such that their fields of view overlap and permit $\theta$ and $\phi$ to be simultaneously observed. This implies that with a proper calibration, two points from each image correspond to the same object location $\mathbf{p}^{(\mathcal{R})}$. We denote these points as

$$z^{(h)} = (R^{(h)}, \theta, \gamma^{(h)})^\top, \qquad z^{(v)} = (R^{(v)}, \phi, \gamma^{(v)})^\top. \tag{3.1}$$

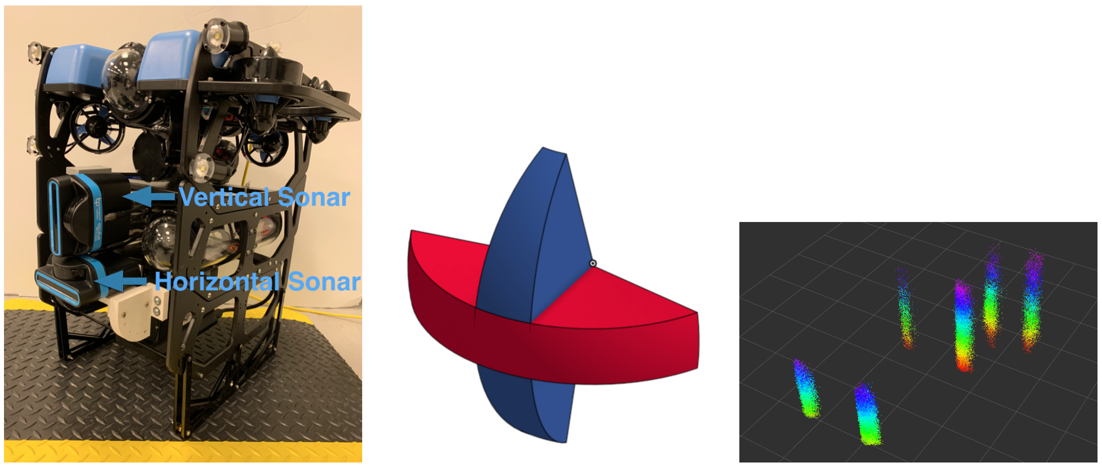
The horizontal sensor, $h$, compresses measurements in the $x$-$y$ plane of $\mathcal{R}$, whereas the vertical sensor, $v$, compresses points in $x$-$z$. Their associated images sets are denoted

$$Z^{(h)} = \{\mathbf{z}_1^{(h)}, \cdots, \mathbf{z}_N^{(h)}\}, \qquad Z^{(v)} = \{\mathbf{z}_1^{(v)}, \cdots, \mathbf{z}_N^{(v)}\},$$

where $N \in \mathbb{N}$ represents the number of observations from each sensor.

Given the intensity images $Z^{(h)}, Z^{(v)}$, we formalize the data association problem as vertex matching in a bipartite graph $G = (V, E)$. The vertices $V = Z^{(h)} \cup Z^{(v)}$ contain all observed intensity points, and the sets

$$E_i = \{(z_i^{(h)}, z_1^{(v)}), \cdots, (z_i^{(h)}, z_N^{(v)})\},$$



(a) ROV With Dual Sonars (b) Sonar overlapping fields of view (c) Reconstructed Pier Pilings

Figure 3.1: **System Overview.** Using two multi-beam sonars, correspondences between their observations are computed to extract 3D point clouds. Sonar fields of view corresponding to the hardware arrangement in (a) are shown in (b) - the red swath is from the horizontal sonar and the blue is from the vertical sonar, shown at a range of 10m. Fig 1(c) shows a reconstruction of pier pilings in the Hudson River. Data was collected at a fixed depth.

define all realizable associations between points in the horizontal image and points in the vertical image. Their union defines the total edge set $E = \cup_{i=1}^{N} E_i$. Solutions are obtained by finding a set $S \subset E$ such that

$$S = \bigcup_{E_i} \underset{(z_i^{(h)}, z_j^{(v)}) \in E_i}{\arg\min} \mathcal{L}(z_i^{(h)}, z_j^{(v)}), \tag{3.2}$$

where $\mathcal{L}(\mathbf{z}_i, \mathbf{z}_j)$ denotes the loss between features. Additionally we require the association to be bijective, in that for any two edges $(z_i^{(h)}, z_i^{(v)})$, $(z_j^{(h)}, z_j^{(v)}) \in S$ it must be that $z_i^{(v)} \neq z_j^{(v)}$. We estimate $p^{(R)}$ from Eq (4.1), using the fused spherical coordinates

$$\hat{p}^{(R)} = \left( \frac{R^{(h)} + R^{(v)}}{2}, \theta^{(h)}, \phi^{(v)} \right)^{\top}. \tag{3.3}$$

Here we use the empirical mean of ranges and the angular values from the resulting association.

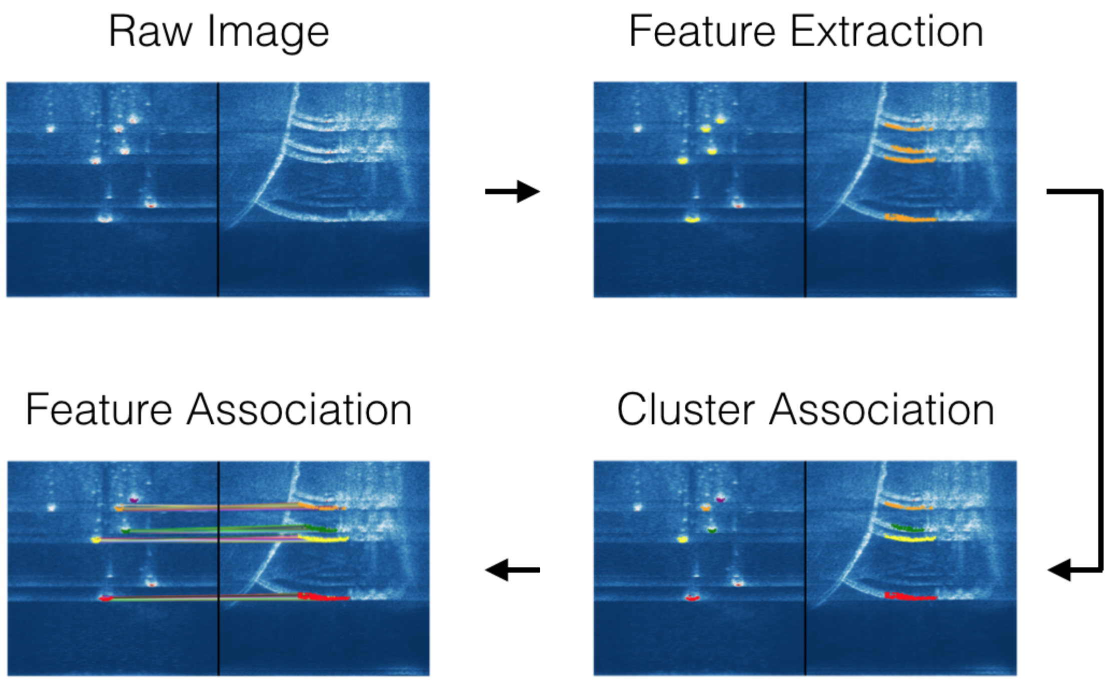


Figure 3.2: **data association architecture overview.** Raw image pairs are taken from the same time step, features are extracted, the features are clustered and then matched using cluster labels as constraints.

## 3.2 3D Reconstruction

Given a set of 3D points $\widehat{P}^{(R)} = \{\hat{p}_i^{(R)}\}_{i=1}^{M}$, we can complete the reconstruction by mapping these into a fixed inertial frame $I$. This is accomplished with the linear transformation $T \in \mathbb{R}^{3\times 3}$. When applied to the set of all points, the result is a point cloud which we call *the map*:

$$\widehat{P}^{(I)} = \{\hat{p}^{(I)} | \hat{p}^{(I)} = T\hat{p}^{(R)} \; \forall \; \hat{p}^{(R)} \in \widehat{P}^{(R)}\}. \tag{3.4}$$

## 3.3 Proposed Algorithm

Here we describe our proposed methodology for identifying feature correspondences across concurrent orthogonal sonar images (summarized in Figure 3.2). The fundamental goal of this pipeline is to associate range measurements from orthogonal, overlapping vantage points that are each lacking a single dimension in their respective spherical coordinate frames. With a set of matched features, the algorithm output (per Eq. (3.3)) is a set of fully defined points in 3D Euclidean space.

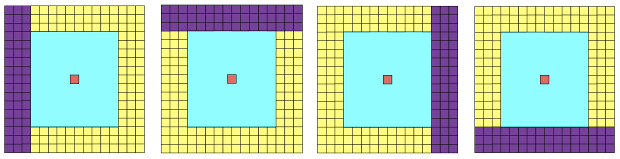


Figure 3.3: **SOCA-CFAR overview.** Purple cells show the training cells, blue the guard cells and red the cell under test.

### 3.3.1 Feature Extraction

In our acoustic imagery we do not process every pixel, as not all pixels represent meaningful returns. We must first identify which pixels belong to surfaces in the scene, and to do this, we apply feature extraction to each acoustic image.

To extract features we use the constant false alarm rate (CFAR) technique [76]. This class of algorithm has shown utility in processing radar images [76], [77] as well as side-scan sonar images [78], which are similarly noisy sensing modalities. We have found it to be the most effective feature detector in practice for reliably and consistently eliminating the second returns that regularly appear in sonar imagery.

CFAR uses a simple threshold to determine if a pixel in a given image is a contact or not a contact. However, it produces a *dynamic* threshold by computing a noise estimate for the area around the cell under test via cell averaging. The technique is sensitive to multiple targets in the image, especially when the noise estimate includes other positive contacts. It is for this reason we utilize a variant known as "smallest of cell averages" (SOCA-CFAR) [77]. SOCA-CFAR computes four noise estimates and utilizes the smallest of the four. This estimate is expressed in Eq. (3.5), with $\mathbf{x_m}$ as a training cell, $\mathbf{N}$ as the number of training cells in that quadrant and $\mu$ as the estimate of noise. This process is shown in Fig. 3.3. We note that when averages are computed (purple cells), a layer of guard cells (blue) is wrapped around the cell under test to prevent portions of the signal from leaking into the noise estimate. Next, the detection constant is computed in Eq. (3.6), with $\alpha$ as the detection constant. $\mathbf{N}$ is once again the number of cells in the quadrant and $\mathbf{P_{fa}}$ is the specified false alarm rate. The threshold, $\beta$ can be computed using Eq. (3.7), with $\mu_{\mathbf{min}}$, the minimum of the computed quadrant averages. The result of this step in the algorithm is two sets of contacts, $\mathcal{S}_t^{(h)}$ from the horizontal sonar and $\mathcal{S}_t^{(v)}$ from the vertical. Note that at each time step, there are two sonar images; these images are independently analyzed for features. A sample pair of sonar images with CFAR points at the top right of Fig. 3.2.

$$\mu = \frac{1}{N}\Sigma_{m=1}^{N} x_m \tag{3.5}$$

$$\alpha = N(P_{fa}^{-1/N} - 1) \tag{3.6}$$

$$\beta = \mu_{min}\alpha \tag{3.7}$$

### 3.3.2 Clustering and Cluster Association

Once features are identified in an image pair, these features require an additional constraint for robust association. To minimize the number of extraneous matches, we can take advantage of the fact that each cluster represents a surface in view. By first clustering the extracted features and then comparing features between matched clusters only, we can create a significantly more robust pipeline.

Clustering is performed using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [79] algorithm on both sets of features $\mathcal{S}_t^{(h)}$ and $\mathcal{S}_t^{(v)}$. DBSCAN is selected because, in each image, the number of clusters is unknown, and this approach does not require knowledge of the scene a priori. DBSCAN works by iterating through the data, catalogging all core points with more than *minSamples* neighbors lying within a radius $\epsilon$, which, along with their neighbors, form our initial clusters. All other unassigned points lying within $\epsilon$ of a cluster are assigned to that cluster. The result of this process is two sets of clusters: $\mathcal{C}_t^{(h)}$ from the horizontal sonar and $\mathcal{C}_t^{(v)}$ from the vertical sonar.

Next, to match clusters across orthogonal sonars, we compute four descriptors and then minimize a cost function to associate clusters. Each cluster is defined by its mean range $\mu$, variance in range $\boldsymbol{\sigma^2}$, and min and max in range, shown in Eq. (3.8). Each cluster $\mathbf{c}_t^{(h)}$ is assigned to the cluster in $\mathcal{C}_t^{(v)}$ that minimizes cost function (3.9):

$$\mathbf{c}_t = \begin{bmatrix} \mu \ \sigma^2 \ r_{min} \ r_{max} \end{bmatrix}^\top, \tag{3.8}$$

$$\mathcal{L}(\mathbf{c}_t^{(h)}, \mathbf{c}_t^{(h)}) = ||\mathbf{c}_t^{(h)} - \mathbf{c}_t^{(v)}||_2. \tag{3.9}$$

### 3.3.3 Feature Association

Following the clustering algorithm output, the next stage is to match individual features within our matched clusters. A feature can only be matched to another feature if they belong to the same cluster, which greatly reduces the potential for extraneous feature matches.

To match features, we once again adopt the descriptor and cost function paradigm. Each feature is defined by range, $\mathbf{r}$ intensity $\gamma$, and a mean intensity kernel. We consider two mean terms for each feature, which are broken into two axes. $\mu_{\mathbf{x}}$ is the mean intensity of $\mathbf{i}$ points right and $\mathbf{i}$ points left of the feature. $\mu_{\mathbf{y}}$ is the mean intensity of $\mathbf{i}$ points above and $\mathbf{i}$ points below the feature, in image coordinates.

The reasoning for this is straightforward; the body of work addressing elevation angle estimation in sonar imagery relies on the relationship between incident angle and intensity. Since our goal is to match similar measurements, we leverage this relationship by hypothesizing that not only should similar measurements have similar range, but also intensities, because of their similar incident angles. However, due to the noise in acoustic images, we adopt not only range and intensity as feature descriptors, but also local averages of intensity. Note that feature descriptors in Eqs. (3.10) and (3.11) have mismatched axes in their $\mu$ terms, which is due to the orthogonality of the images:

$$\mathbf{z}_i^{(h)} = \begin{bmatrix} r \ \gamma \ \mu_x \ \mu_y \end{bmatrix}^\top \tag{3.10}$$

$$\mathbf{z}_j^{(v)} = \begin{bmatrix} r \ \gamma \ \mu_y \ \mu_x \end{bmatrix}^\top \tag{3.11}$$

$$\mathcal{L}(\mathbf{z}_i^{(h)}, \mathbf{z}_j^{(v)}) = ||\mathbf{z}_i^{(h)} - \mathbf{z}_j^{(v)}||_2. \tag{3.12}$$

We next compute the minimum-loss association among the features residing in previously associated clusters. When carrying out this process, we only allow any feature to be matched with a single other feature. Moreover, before evaluating cost we *normalize* all components by subtracting from each feature the min image intensity and dividing by the difference between min and max. These min and max values are computed for each image, at every timestep.

In our implementation, we have developed two versions of this process; the first is a “brute force” method in which all possible correspondences for a given cluster are tested for each feature, and the lowest cost is used provided it is below a designated threshold. Alternatively, we take a fixed number of random samples from a cluster and again use the lowest cost, provided it is below our threshold. These two versions of our proposed algorithm will later be quantitatively compared.

## 3.4 Experiments and Results

In this section, we will examine three experiments and provide an analysis of their outcomes. We will discuss the specifics of our experimental setup and the hardware used to validate the proposed perceptual framework of Fig. 3.2.

### 3.4.1 Overlapping Area Considerations

When extracting features, we take a highly conservative approach to ensure that features are only extracted from the region of overlap between the sonars. We only extract features from the area inside the vertical aperture of each sonar’s orthogonal companion. This conservative approach ensures no features are extracted outside of the overlapping area depicted in Fig. 3.1(b).

### 3.4.2 Compensating for Sonar Misalignment

Before any data association can take place, we must transform our vertical sonar features $\mathcal{S}_t^{(v)}$ to the horizontal sonar frame. In our experimental setup, there is a 10cm vertical offset between the sonar coordinate frames, shown in Fig. 3.1(a). This is accounted for by applying Eq. (4.1) to transform our features in $\mathcal{S}_t^{(v)}$ to Cartesian coordinates, translate them a few centimeters downward to the horizontal sonar coordinate frame, and then apply the inverse of Eq. (4.1) to transform the features back into spherical coordinates for association. To apply Eq. (4.1), an elevation angle of zero is assumed for all points in the vertical sonar image (this assumption is used to facilitate data association only).

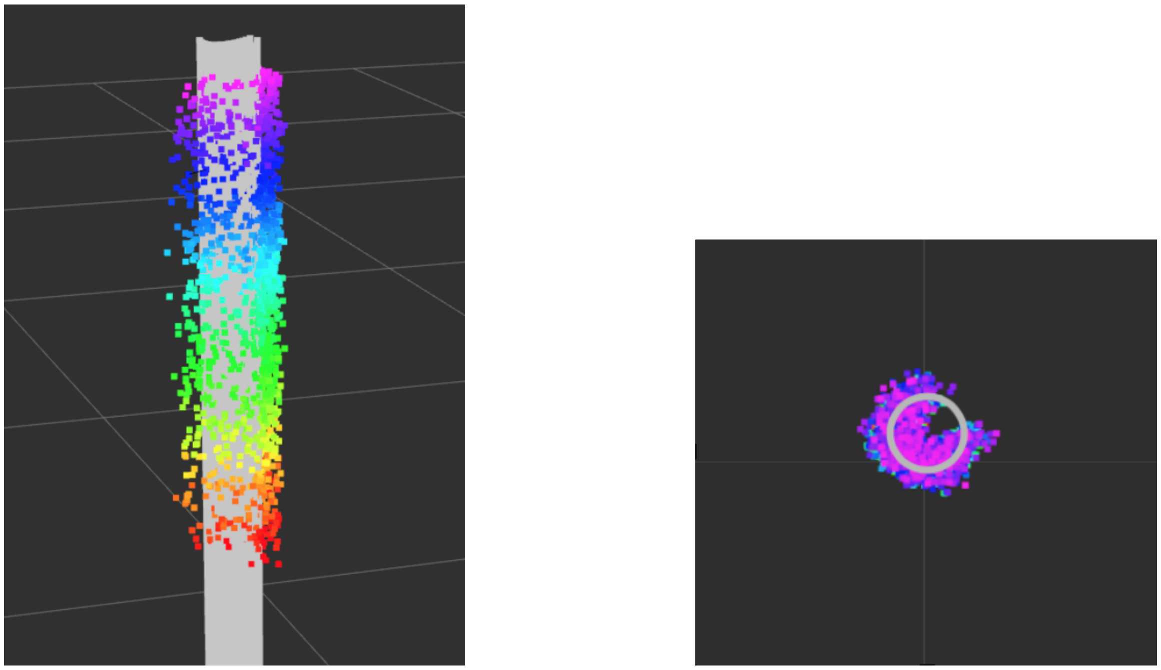
### 3.4.3 Error Metrics

To compare different parameterizations of our method against each other and a benchmark, we utilize two error metrics. Firstly we compute mean absolute error (MAE); this is calculated by comparing the generated point cloud to a CAD model of the object. Secondly, we compute the root mean squared error (RMSE); once again, this is generated by comparing the point cloud to the CAD model. For our tank experiments, to capture the transformation between the ROV frame and the CAD model with high accuracy, the objects in our sonar imagery are hand segmented to identify both the object coordinate frame origin (located at the center of the pipe comprising all structures), and its rotation in yaw relative to the ROV. The remaining angles in the transformation are obtained from the ROV's IMU.

### 3.4.4 Simple Object Reconstruction

In this first experiment, a cylindrical piling mock-up is submerged in our test tank, and a sequence of data is collected with our sonar pair oriented at a grazing angle of 20° below horizontal. This grazing angle is used in order to facilitate benchmarking, as the current state of the art [9] assumes this problem structure. When collecting data, the ROV is piloted in a circular-segment orbit pattern around the structure. During this flight pattern, the structure is held at a similar range and bearing while the ROV traverses to port or starboard. Data collection occurs at a fixed depth.

The purpose of this experiment is to evaluate the performance of the proposed algorithm on a simple structure. Additionally, this experiment allows us to evaluate



(a) Isometric view (b) Top down view

Figure 3.4: **Results from pier piling mockup.** A model of the structure's true dimensions is shown in gray (outer diameter is 9cm), with multi-colored points depicting algorithm output. Sonars were operated at 5m range, and results from our algorithm's "fast, clustering" configuration are shown. Colors indicate height.

our algorithm against the current state of the art, as [9] shows several experiments reconstructing similar objects. During the comparison, it is essential to note that we use SOCA-CFAR feature extraction in conjunction with our implementation of [9] to ensure that all systems run on the same inputs.

While the goal of our algorithm is to move toward the reconstruction of arbitrary objects with wide-aperture multi-beam sonar, we are acutely aware that the algorithm we are proposing requires a second sonar. This experiment serves to show that performance does not degrade in the case of objects that can be successfully reconstructed with a single sonar. We compare four variations of our algorithm against [9], as it achieves the best performance in our tank among the suitable algorithms in the literature. Additionally, we posit that [9] fairly represents the foundational works of [7], [8], building upon them and offering broader applicability.

We compare four configurations of our algorithm to show performance gains and trade-offs for different versions quantitatively. Firstly we evaluate the introduction of clusters as feature correspondence constraints. Secondly, we compare the "brute force" approach in which all feature combinations are checked via (3.12), and a version in which ten random samples from a feature's corresponding cluster are checked, with the best adopted if below a designated threshold. This second version runs in real-time over all data gathered, while the brute force method runs significantly slower. We also note that [9] runs in real-time in these experiments. The feature correspondence threshold is the same for all methods and experiments provided in this paper; it is set to 0.1.

Experimental results are shown in Table 3.1 and Fig. 3.4; our proposed algorithm performs comparably to the current state of the art, and moreover, the results from our implementation of [9] are in line with those shown in the original paper. Any variation in performance relative to [9] can be attributed to the fact that the sonar used in our implementation has half the angular resolution as in [9], and we analyze raw point clouds rather than filtered surface meshes.

When analyzing the results of different configurations of our proposed algorithm, it is not surprising that in this simple case, with only a single cylindrical piling in view, the cluster constraints provide little added value. Additionally, the trade-off between fast and brute force methods is evident, with a slight loss of accuracy in exchange for real-time viability.

### 3.4.5 Complex Object Reconstruction

Recall that to reconstruct objects using a single wide-aperture imaging sonar, the framework proposed in [9] must make two critical assumptions; the first being that

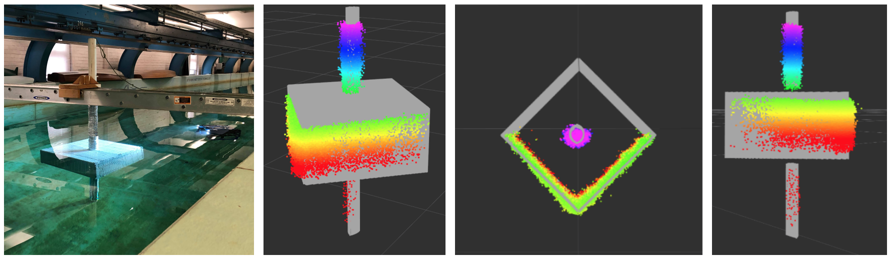
| Algorithm | MAE (cm) | RMSE (cm) |
|---|---|---|
| Westman and Kaess [9] | 2.20 | 2.64 |
| Brute Force Without Clustering | 2.16 | 2.53 |
| Fast Without Clustering | 2.34 | 2.76 |
| Brute Force With Clustering | 2.27 | 2.70 |
| Fast With Clustering | 2.35 | 2.83 |

Table 3.1: A summary of reconstruction performance corresponding to the reconstruction of a single piling (pictured in Fig. 3.4).

the range to an object increases or decreases monotonically with elevation angle. The consequence of an object violating this assumption is the inability to reconstruct the geometry accurately. Moreover, [9] states that "a violation of this assumption would cause a self-occlusion, and the corresponding pixels would presumably not be classified as surface pixels by the frontend of our algorithm." The second key assumption is that the sonar is oriented at a downward grazing angle, which enables identification of an object's leading edge. In our proposed framework, we require no assumptions about the geometry of the objects in view, nor do we require the sensor at a grazing angle.

In this experiment, we test our proposed algorithm on a mock-up of a critical piece of subsea infrastructure, the blow out preventer (BOP) - approximated by a rectangular object mounted on a cylinder. BOPs sit on the seafloor at the wellhead during offshore drilling and are increasingly subject to regulatory scrutiny and industry monitoring requirements. Critically though, like many other subsea assets, the vertical cross-section of this object does not conform to the aforementioned geometric assumptions.

Once again, data is collected by piloting the ROV in a circular-segment orbit around a portion of the structure (low clearances in our tank prevent full circumnavigation of the structure). Recall that this flight pattern keeps the structure at a similar range and bearing while the ROV traverses to port or starboard. In this



(a) Water Tank Setup (b) Isometric view (c) Top View (d) Side View

Figure 3.5: **Results from blowout preventer mockup.** Gray shows the true structure geometry (a 9cm OD pipe, with an 82 x 82 x 45cm box, whose center is 1.2m from the bottom of pipe), and in (c), the top of the structure is not shown for ease of visualization. Sonars were operated at 5m range, and results from our algorithm's "fast, clustering" configuration are shown. Color indicates height.

experiment, however, the sonar is not at a grazing angle, configured instead for maximum situational awareness. Unlike in the other experiments in this paper, data is collected at a variety of depths.

A summary of the results is provided in Table 3.2 and Fig. 3.5. Not only is our algorithm able to provide a realistic reconstruction of the object, but it can do so within 6cm of its true geometry. In analyzing the results of this experiment, the benefits of clustering as a constraint in feature association are evident; clustering dramatically improves both MAE and RMSE. Again the trade-off between brute force and fast feature-matching is evident; a small decrease in performance is required in order to run in real-time. This trend is not evident without clustering, and the reason for this is not known for certain, but the random guessing associated with the fast version may be acting as a filter. The reconstruction's appearance in Fig. 3.5 is realistic, with no evident outliers, particularly in the vertical axis. While it is unfortunate our algorithm is unable to reconstruct the top surface of the rectangular object, this was not surprising given the object's sharp angles, and the occlusions they present. From the vantage points examined, the top rectangular surface of the structure was imaged at much lower intensity than its front edges. We have omitted a direct comparison with [9] for this structure in an effort not to misrepresent their work, in a modality they explicitly state their algorithm is not intended for.

| Algorithm | MAE (cm) | RMSE (cm) |
|---|---|---|
| Brute Force Without Clustering | 62.29 | 69.39 |
| Fast Without Clustering | 30.24 | 45.91 |
| Brute Force With Clustering | 5.31 | 10.06 |
| Fast With Clustering | 5.74 | 12.75 |

Table 3.2: A summary of model reconstruction performance corresponding to the reconstruction of the BOP mockup.

### 3.4.6 Field Based Object Reconstruction

The final experiment to be presented is a demonstration of the proposed algorithm operating in the field. For this work, our ROV was deployed in the Hudson River at Hoboken, NJ. A short inspection flight was flown at the junction of two piers with six pilings supporting the structure close to each other. This exercise takes place at a fixed depth and the sonars without any grazing angle. The version of the algorithm used for this demonstration is fast, with clustering.

The results, shown in Figs. 3.1(c) and 3.6 and whose sonar images also appear in illustrative Fig. 3.2, demonstrate that a robot employing the proposed framework can achieve significant situational awareness, even while operating at a fixed depth. We can recover dense 3D point clouds at every timestep, which are registered here using only iterative closest point (ICP). For the same inspection coverage to be achieved with a profiling sonar, changes in depth would be required. Furthermore, reconstructing these pilings with a single wide-aperture sonar [9] would require a grazing angle that would limit situational awareness. Moreover, if the vehicle is perturbed, an event of significant likelihood given the wakes and currents encountered in this tidal river basin, the requirements of prior algorithms may be violated. There is another crucial trade-off here: to extract 3D information from the sensors, a reduction in the horizontal field of view to the overlapping area between sonars is required. We believe this is a reasonable trade, though, as Fig. 3.2 demonstrates that reasonable coverage can be achieved, even with the reduced-size portions of sonar images that overlap, at the sensing range of 10m explored in this experiment.

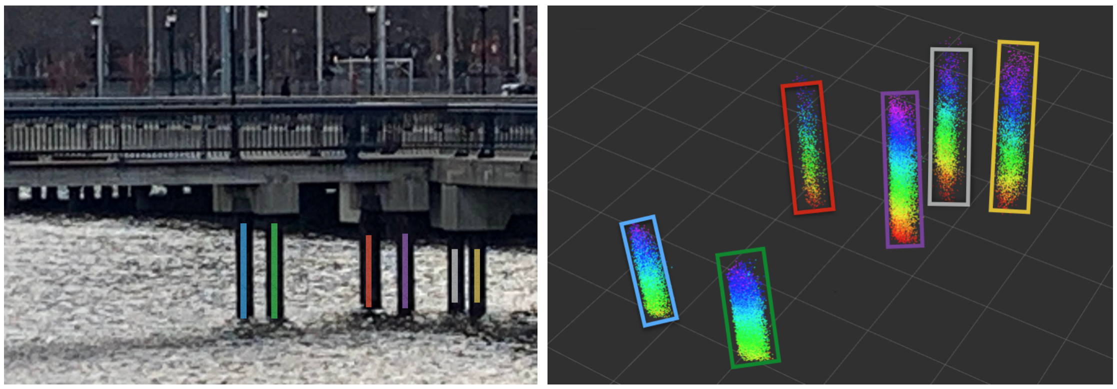

(a) Outdoor Pier Pilings (b) Piling Reconstruction

Figure 3.6: Hudson River pier pilings at left with reconstruction at right (shown also in Fig. 1(c)). Data was collected at a fixed depth, with sonars operated at 10m range, and results from our algorithm's "fast, clustering" configuration are shown. Point colors indicate height.

## 3.5 Conclusions

In this section, we have presented a new framework for achieving dense 3D reconstruction of arbitrary scenes using a pair of orthogonally oriented, wide-aperture multi-beam imaging sonars. We have provided a detailed description of a pipeline for robust feature extraction, clustering, and descriptors that facilitate accurate matching across concurrent sonar frames. Further, we have shown quantitatively that our algorithm performs competitively with the state of the art in reconstruction using a single imaging sonar, in simple cases where a comparison is possible. Most importantly, this methodology introduces a new dense sonar mapping capability for complex scenes, and unlike previous work it requires few if any assumptions about the environment, advancing progress toward reconstructing arbitrary geometries and cluttered scenes with wide-aperture multi-beam sonar.

Future work will focus on using this methodology as a basis for mapping beyond the area of explicit overlap between sonars, and incorporating this framework into a process for robust, underwater 3D active SLAM.

# Chapter 4
# Predictive 3D Sonar Mapping of Underwater Environments via Object-specific Bayesian Inference

## 4.1 Problem Description

In this work, we consider 3D reconstruction using a pair of orthogonal imaging sonars with an overlapping field of view. A robot visits a series of poses $x_t$, with transformations $\mathbf{T} \in \mathbb{R}^{4\times 4}$. Each pose has associated observations $z_t$, with two components: horizontal sonar observations $z^h$ and vertical sonar observations $z^v$. Each set of observations is defined as an intensity image in spherical coordinates with range $R \in \mathbb{R}_+$, bearing $\theta \in \Theta$, and elevation $\phi \in \Phi$, with $\Theta, \Phi \subseteq [-\pi, \pi)$, and an associated intensity value $\gamma \in \mathbb{R}_+$. These measurements can be converted to Cartesian space:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R \begin{pmatrix} \cos\phi\cos\theta \\ \cos\phi\sin\theta \\ \sin\phi \end{pmatrix}. \tag{4.1}$$

Each recorded measurement $z^h$ and $z^v$ is characterized by an omitted degree-of-freedom (DoF), and in the robot frame, due to the orthogonality of the two sonars, these DoFs differ:

$$z^h = (R^h, \theta, \gamma^h)^\top, \qquad z^v = (R^v, \phi, \gamma^v)^\top. \tag{4.2}$$

As in section 3, we associate measurements across concurrent, orthogonal images to fully define the measurements in 3D, yielding Equation (4.3):

$$z^{Fused} = \left( \frac{R^h + R^v}{2}, \theta^{(h)}, \phi^{(v)} \right)^\top . \tag{4.3}$$

However, this association can only take place within the small region of overlap between the sonars' fields of view, equivalent in size (in both bearing and elevation) to the sonar's vertical beamwidth. This leaves large portions of each image with a missing DoF and, therefore, undefined in 3D. In this work, we wish to perform a 3D reconstruction by mapping the observations into fixed frame $\mathcal{I}$ as in Eq. (4.4).

$$\mathcal{M} = \{\hat{z}^{(\mathcal{I})} | \hat{z}^{(\mathcal{I})} = \mathbf{T}\hat{z}^{Fused} \; \forall \; \hat{z} \in \widehat{z}\} \tag{4.4}$$

The information missing from a portion of all images introduces ambiguity into the application of Eqs. (4.1) and (4.4). The question to be answered in this paper becomes: How can we leverage the fully defined measurements from Eq. (4.3) to infer the unknown 3D structure of the rest of the imagery, producing a more comprehensive 3D reconstruction?

## 4.2 Proposed Algorithm

We propose to exploit the observation of repeating objects throughout the environment, using the 3D observations occasionally obtained for specific objects, to infer the 3D structure when those same objects are re-observed without a known elevation angle. First, we identify objects in the sonar image and provide a semantic class label. Next, we develop a Bayesian inference model for each object class to later query this model for points in the horizontal sonar image corresponding to that same class.

Note that we only apply this inference procedure to the horizontal sonar image. Due to the shallow depths at which we operate our vehicle, the vertical image contains fewer meaningful observations of subsea structures and is not subjected to Bayesian inference.

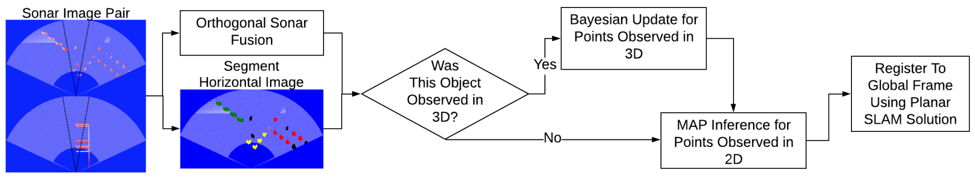

Sonar Image Pair
Orthogonal Sonar Fusion
Segment Horizontal Image
Was This Object Observed in 3D?
Yes
No
Bayesian Update for Points Observed in 3D
MAP Inference for Points Observed in 2D
Register To Global Frame Using Planar SLAM Solution

Figure 4.1: **System block diagram.** A pair of orthogonal sonar images is provided as input (black lines bound the region of overlap between the two sonar fields-of-view). The images are processed according to Section IV.B. The horizontal image is segmented as in Section IV.C (colors denote different object classes - seawall in green, rectangular pilings in yellow, cylindrical pilings in red). The resulting 3D points enrich each object's model (Eq. (4.7)), while MAP inference is applied to 2D points (Eqs. (4.8), (4.9)). We then use the planar SLAM solution to register the resulting point cloud. The synthetic sonar images shown here are sampled from the virtual environment depicted in Fig. 4.2.

### 4.2.1 SLAM Solution

In this work, we utilize a pose SLAM formulation to estimate our robot's pose history through time. We restrict our formulation to 3-DoF estimation in the plane to provide an efficient and robust SLAM pipeline that prioritizes the DoFs of greatest uncertainty (surge, sway, and yaw).

Here we formulate SLAM on a factor graph, which is solved with the aid of GTSAM [80] and iSAM2 [81]. At each keyframe, features are extracted from our robot's horizontal sonar using the method described below in Sec. IV.B, and the unknown angle $\phi$ is assigned as zero. The constraints relating adjacent keyframes are estimated using iterative closest point (ICP) [82], with global initialization via consensus set maximization [83] - we denote this step sequential scan matching (SSM). Loop closures are added by applying the same scan matching process to non-consecutive keyframes, applying ICP to frames within a designated radius of the current keyframe - we denote this step non-sequential scan matching (NSSM). To reject loop closure outliers, we use pairwise consistent measurement set maximization [72]. Our factor graph is given by

$$\mathbf{f}(\boldsymbol{\Theta}) = \mathbf{f}^0(\boldsymbol{\Theta}_0) \prod_i \mathbf{f}_i^{\mathrm{O}}(\boldsymbol{\Theta}_i) \prod_j \mathbf{f}_j^{\mathrm{SSM}}(\boldsymbol{\Theta}_j) \prod_q \mathbf{f}_q^{\mathrm{NSSM}}(\boldsymbol{\Theta}_q).$$

This planar SLAM solution is used to provide estimates of surge, sway and yaw for registering our algorithm outputs to the global frame; the remaining degrees of freedom come from our vehicle's pressure sensor and inertial measurement unit (IMU). Note that we will only analyze the sonar images corresponding to SLAM keyframes to develop our 3D map. We do not analyze images between keyframes, due to the high drift rate of our low-cost dead reckoning system.

### 4.2.2 Sonar Fusion System

At each time step, the robot receives a set of observations constituting a pair of sonar images, as described in Eq. (4.2). To fully define these in 3D as denoted in Eq. (4.3), we need to extract and associate features across the images.

Here we use a similar approach to that of our prior work in section 3, dividing the image-feature matching problem into multiple, smaller, subproblems. Recall that range $R$ is discrete within a sonar image, and since associated features should be at the same range, we use the range to define these subproblems. All features at the same range in each sonar image are gathered and processed using intensity-based association. The cost function used here is defined as follows:

$$\mathcal{L}(z_i^h, z_j^v) = ||\nu^h - \nu^v||, \tag{4.5}$$

where $\nu^h$ and $\nu^v$ are square patches of the sonar image around the given feature. These patches have a fixed size of 5x5 pixels in this work. Note that $\nu^v$ is rotated 90 degrees to account for the orthogonality of the images. Moreover, before this comparison is made, the images' intensity values are normalized at every time step. The cost function in Eq. (4.5) is used to find the solution that minimizes the sum of costs between features for each subproblem.

To estimate our confidence in these matches, we compare the two best solutions for each feature association [84]:

$$C = \frac{\mathcal{L}(z_i^h, z_j^v)_{min2} - \mathcal{L}(z_i^h, z_j^v)_{min}}{\Sigma_{0,0}^{i,j}\mathcal{L}(z_i^h, z_j^v)}. \tag{4.6}$$

While we compute subproblem cost totals in order to find sets of associated features in Eq. (4.5), confidence is evaluated on a feature-wise basis, comparing the costs for each individual feature association made in the given solution. This gives us a simple metric with which to cull uncertain associations.

### 4.2.3 Image Feature Extraction and Object Classification

To extract features from sonar imagery, we use the method employed in section 3.3.1, used in various radar and sonar applications. The technique, smallest-of cell averaging constant false alarm rate (SOCA-CFAR) detection [76], takes local area averages around the pixel in question and produces a noise estimate. If the signal is greater than a designated threshold, the pixel is identified as an image-feature.

Next, objects must be identified; for this, we utilize DBSCAN [79], since the number of clusters in an image is not known a priori. The results from this step are image-features clustered into objects with unknown class labels. Note that even with a feature detector as robust as SOCA-CFAR, noise is still present. Accordingly, only clusters with $n$ or more image-features are passed on to the next step.

Lastly, we must provide a class label for each object. In this work, we use a simple neural network to perform semantic labeling of object instances. Specifically, we use a CNN that accepts 40x40 pixel sonar image patches in grayscale, with two convolutional layers. Inputs are generated by fitting a bounding box around each object identified in a sonar image and resizing the bounding box into 40x40 pixels, while preserving the object's aspect ratio. We utilize Monte-Carlo dropout in this CNN to reject outliers and uncertain classifications by making $m$ predictions for each object. In this way, we can assess the network's confidence in its predictions, as shown in [85]. Uncertain predictions are simply provided with a label "unknown class." An example of this pipeline in action is shown in Fig. 4.1.

To train this CNN, a small hand-annotated data set of representative sonar imagery is used, which is not included in the sequences used for validation in this work. To generate sufficient training samples to properly train the model, data augmentation is used. We augment our data by applying Gaussian noise, random flips and random rotations.

### 4.2.4 Bayesian Inference for Objects Observed in 3D

Each detected object in the horizontal sonar image is now represented by a cluster of features with a class label. These features have a range, bearing, and unknown elevation angle. At this step, the dual sonar fusion system provides an elevation angle for a subset of these points, which lie inside the small region with overlapping fields of view. These are the points we concern ourselves with in this subsection.

We assume objects of the same class will have similar geometries, as is typical in the humanmade littoral environments, populated with piers, used to validate this algorithm. Semantic classes are defined with this goal in mind, so that objects with similar geometries are grouped together.

We use a Bayesian inference framework to estimate the conditional distribution $\mathrm{P}(z_Z^h|z_R^h, z_\theta^h)$ for each object class incrementally and online. Note that in this process, we estimate Cartesian $z_Z$ and not elevation angle. $z_Z$ is a more accurate indicator of the absolute, rather than relative, height at which an object is observed, since in this work we consider scenarios in which our robot maps the environment at fixed depth, employing planar SLAM. An object's distribution is updated for every measured 3D point per Bayes rule:

$$P(z_Z^h|z_R^h, z_\theta^h) = \frac{P(z_R^h, z_\theta^h|z_Z^h)P(z_Z^h)}{P(z_R^h, z_\theta^h)}. \tag{4.7}$$

Elevation angles measured by the dual sonar fusion system are treated as measurements of $z_Z^h$ at the given range and bearing, corrupted with zero-mean Gaussian noise, $\mathcal{N}(\mu, \sigma^{\epsilon})$, forming the measurement likelihood $P(z_R^h, z_\theta^h|z_Z^h)$. The prior, $P(z_Z^h)$, is simply the existing distribution corresponding to the $z_R^h$ and $z_\theta^h$ of the newly observed 3D point. We note that these distributions are maintained throughout the whole time-history of the robot's mission, so they incorporate observations from the current frame along with observations from all previous frames. If an update has never been performed, an initial uniform distribution is used.

At times we may view an object class at different distances and orientations; for this reason, we register each object to a *reference coordinate frame* before we apply Bayes rule in Eq. (4.7). The first time we see an object, we note its minimum range and median bearing as the reference frame's origin. These object points are then maintained as a "reference point cloud" to register the object class's future instances to this coordinate frame. When an object is detected and its distribution $\mathrm{P}(z_Z^h|z_R^h, z_\theta^h)$ is updated, the object is first registered to the given object class's reference coordinate frame using ICP. The transformed points are evaluated via Eq. (4.7) and are added to the points used to register future object sightings to the reference frame. Our object-specific distributions P$(z_Z^h|z_R^h, z_\theta^h)$ will next allow us to predict the height of the sonar returns observed only in 2D.

### 4.2.5 Predicting 3D Structure via MAP Estimation

At each time step, after the process detailed in Section IV.D is completed for the objects measured in 3D, we again consider all objects with class labels comprised of a minimum number of features. We now use the posterior distribution of each object's geometry, $P(z_Z^h|z_R^h, z_\theta^h)$, to predict the height of all 2D points lacking this information.

Suppose an object belongs to a class with a posterior updated by at least one application of Eq. (4.7). In that case, we proceed, first with registration to the object class's reference frame as described above, without adding new points to the reference point cloud. Due to the sonar's ambiguity, it may be that there is more than one true $z_Z^h$ for a given range and bearing. For this reason, we break maximum a posteriori (MAP) estimation into two steps, as shown in Eqs. (4.8), (4.9).

$$z_Z^h = argmax P(z_Z^h|z_R^h, z_\theta^h), z_Z^h \leq 0 \tag{4.8}$$

$$z_Z^h = argmax P(z_Z^h|z_R^h, z_\theta^h), z_Z^h > 0 \tag{4.9}$$

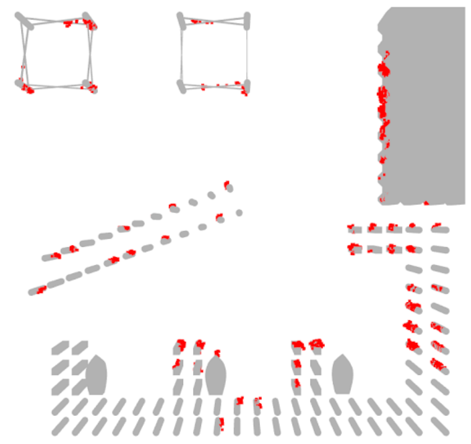
If one or both maxima correspond to confidence exceeding a designated threshold, those values are adopted for inclusion in the robot's map. Eq. (4.1) is solved to provide an output in local Cartesian coordinates, $[X, Y, Z]^T$. This process is completed for all objects in view – the result is a horizontal sonar image with more observations fully defined in 3D, rather than just the few observations inside the region of dual-sonar overlap. The observations are converted to a point cloud and registered to the global map frame per Eq. (4.4).

## 4.3 Experiments

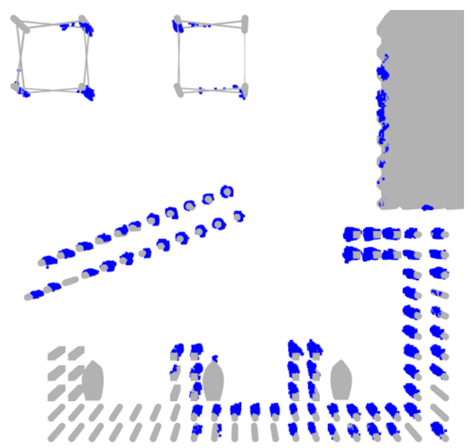
### 4.3.1 Simulation Study

In this section, we utilize Gazebo [86] with the UUV simulator [87] to quantitatively validate our method. The UUV simulator provides an implementation of [88] to simulate wide aperture multi-beam sonar. This simulation environment allows us to perform a quantitative study impossible with field data, where perfect ground truth information is available. We design a humanmade, littoral environment that resembles many marinas and waterfronts. This environment includes two differently shaped pilings (cylindrical and rectangular), boat hulls, corrugated seawalls, and trusses.

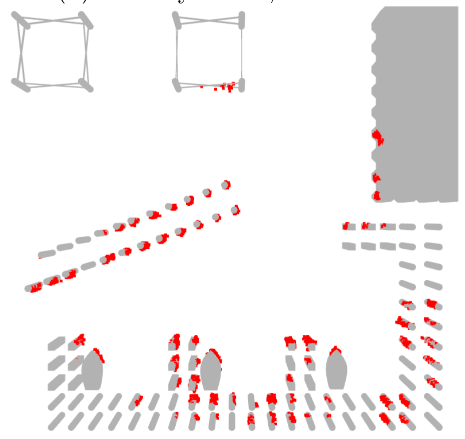
This work proposes an inference method that divides the world into semantic classes by estimating each object class's geometry. These estimates are leveraged to perform 2D-to-3D inference over wide aperture multibeam sonar data. However, not all objects can be treated this way. For example, a wall detected in a sonar image will be difficult to accurately register to a reference coordinate frame, since it may not be fully captured within a single image. Moreover, a pier piling near the edge of a sonar image cannot be distinguished from the edge of a wall that extends beyond the image. Conversely, objects that fit entirely inside the sonar image can be estimated with some confidence, as the registration process described above is readily applicable. It is for this reason that we confine our inference framework in simulation to two classes, cylindrical and rectangular pier pilings. The remaining classes of boat hull, truss and wall, though present in our Gazebo model, are not considered in our predictive mapping framework. A 200-image per class hand-annotated dataset is used



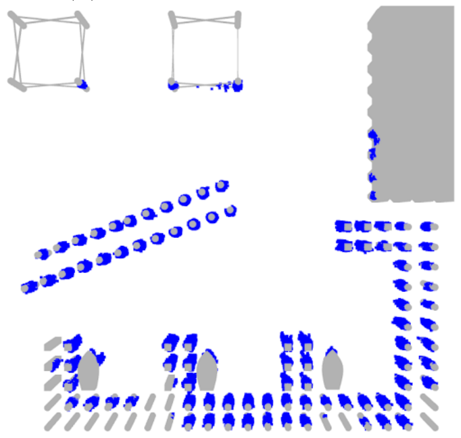
(a) 4m Keyframes, Benchmark



(b) 4m Keyframes, Proposed



(c) 2m Keyframes, Benchmark



(d) 2m Keyframes, Proposed

Figure 4.2: **Qualitative Simulation Results.** The top row and bottom row compare results from 4m keyframe spacing and 2m keyframe spacing, respectively. Our proposed method's results are displayed in the right column in blue, and benchmark results (without Bayesian inference) are displayed in the left column in red.

to train the CNN.

Recall that we use planar pose SLAM to provide transformations to the global map frame. To ensure accurate mapping, only imagery from SLAM keyframes is introduced as input to our framework. We compare two configurations of varying density in this study, a sparse keyframe spacing of 4 meters and a denser keyframe spacing of 2 meters. Note that in this simulation study, while we utilize the same down-sampling as a real SLAM solution, we provide the SLAM pipeline with the robot's true pose in order to isolate errors to only the 3D reconstruction system. As a

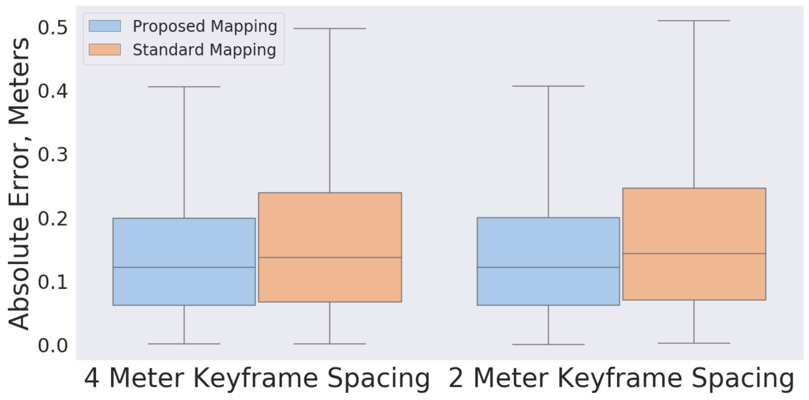


Figure 4.3: **Simulation Error.** On the left we show the distribution of absolute errors for a keyframe spacing of 4m. On the right we show the distribution of absolute errors for a keyframe spacing of 2m. Errors of our proposed method are displayed in blue; the gold shows errors of simply registering dual sonar fusion outputs. Outliers not shown comprise between 1.7-2.0 percent of each dataset.

benchmark for comparison we use a simple method, registering the dual sonar fusion system's 3D output over each keyframe, without classification or Bayesian inference. While naive, this method represents the state of the art in 3D reconstruction with wide aperture multibeam sonar (see section 3), while minimizing assumptions with regard to the appearance of the environment.

For every trial, our robot navigates to each of 20 randomly sampled goals, sequencing them using nearest neighbor; starting at a random location. Collision avoidance is achieved using a 2D, in-plane roadmap for navigation purposes only, which is in no way used in our 3D mapping algorithm. We use A* in conjunction with the roadmap to generate trajectories from one goal to the next. Ten trials are run for each SLAM configuration. Fig. 4.3 shows that our predictive 3D mapping method has comparable error values to the benchmark 3D mapping method. Moreover, per Table 4.1, the proposed method provides an order of magnitude greater coverage for each SLAM configuration. Most critically though, our method with half as many keyframes (spaced 4m apart) has an order of magnitude better coverage than the benchmark with twice as many keyframes (2m). Qualitative results are shown in Fig.

4.2, which clearly illustrates the improvement in point cloud density over repeating objects, in this case the two different types of pier pilings, without impacting the reconstruction of the trusses, boats, or seawall.

| Algorithm | Mean Voxel Count | Std. Deviation |
|---|---|---|
| Standard Mapping, 4 Meter | 2938.6 | 984.18 |
| **Semantic Mapping, 4 Meter** | **60091.4** | **20038.116** |
| Standard Mapping, 2 Meter | 3549.4 | 1144.72 |
| **Semantic Mapping, 2 Meter** | **68422.2** | **25518.37** |

Table 4.1: **Simulation coverage results**, where point clouds are voxelized using a 10cm grid cell resolution.

We use two metrics to quantify performance: absolute error and voxel count. Absolute error is calculated by finding the shortest distance between a given point in the final point cloud and the CAD model of the environment (Fig. 4.3). Voxel count (Table 4.1) is calculated by taking the final point cloud and voxelizing it, iterating over all the points and placing them in their respective discrete bins. If a voxel contains one or more points, it is counted; otherwise, it is not. This study uses voxel count to quantify coverage, while not using redundant information contained in the point cloud.

### 4.3.2 Field Results

We next apply our method to real sonar data, using two datasets collected in SUNY Maritime College's marina on the East River in Bronx, NY. This environment represents the canonical humanmade littoral environment populated by piers, seawalls, and steel floating docks. Further, the East River poses serious environmental challenges such as low visibility, drastic tidal changes, and currents up to 2 knots.

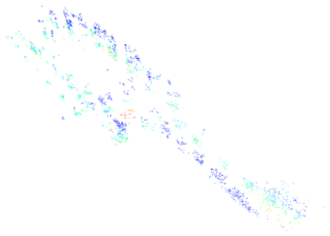
To generate data, our BlueROV is manually piloted along the perimeter of

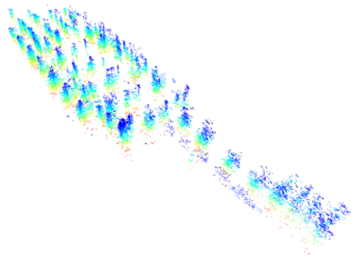
| Algorithm | Voxel Count 2m Keyframe | Voxel Count 4m Keyframe |
|---|---|---|
| Standard Mapping, Pier | 4147 | 2040 |
| **Semantic Mapping, Pier** | **38143** | **20766** |
| Standard Mapping, Waterfront | 5891 | 2819 |
| **Semantic Mapping, Waterfront** | **32131** | **13135** |

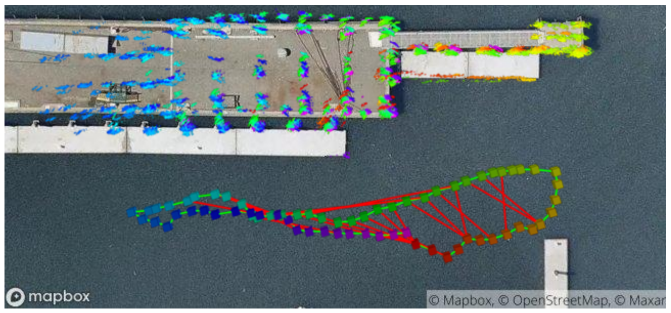
Table 4.2: **Field coverage results**, where point clouds are voxelized using a 10cm grid cell resolution. "Pier" (shown in Fig. 4.4) and "Waterfront" refer to two large structures in the SUNY Maritime marina.

structures in the marina while the vehicle heading is held near constant, with the vehicle strafing to either port or starboard. In the 300-image, hand-annotated dataset used to train our system, we use two classes: cylindrical pier piling and wall. Due to the aforementioned uncertainty associated with edge features, we only apply our inference method to the cylindrical pier piling class.

Since ground truth for mapping is not available, we analyze voxel count to measure coverage and assess the resulting point clouds. We produce separate maps for two structures in different areas of the marina, which we term "Pier" and "Waterfront". Once again, our method provides a dramatic increase in coverage, mapping many areas the benchmark leaves blank. Coverage results are shown in Table 4.2, and detailed maps are provided in our supplementary video. An overview of the Pier dataset is provided in Fig. 4.4. Roughly 3000 object classifications occur in this dataset. While we only apply our inference method to one class, because this class is constantly repeated throughout the environment, map coverage is greatly improved. Most critically, these datasets show the utility of our method on real-world sonar data gathered in a complex littoral environment.


(a) 3D Reconstruction without Bayesian inference


(b) 3D Reconstruction with Bayesian inference



(c) Satellite image overlay of 3D reconstruction

Figure 4.4: **System Overview:** (a)(b) shows a sample reconstruction from SUNY Maritime's marina in the East River ((a) shows raw 3D observations and (b) shows with Bayesian inference) and (c) shows a top-down view of (b) with a satellite image overlay, and the corresponding SLAM pose graph. Squares in (c) show AUV poses with color corresponding to time; green lines are sequential factors; red lines are loop closures. Note: The point cloud in (c) uses the same time mapping as the corresponding poses, while the point cloud in (b) has color mapped to the vertical axis.

## 4.4 Conclusions

In this section we have proposed using semantic classes to aid 2D-to-3D Bayesian inference over wide aperture multibeam sonar data. We have shown through experimental validation that exploiting the repeated observation of common structures in a littoral environment can permit highly accurate predictive mapping, without the need for CAD models a priori. In the absence of these structures, this method can still be employed, but with reduced coverage.

# Chapter 5

# Submapping for Building Dense 3D Maps of Underwater Environments

This section will describe our newest contribution to 3D mapping using orthogonal sonar images. As noted in chapters 3 and 4, we only add to the 3D map at the SLAM keyframes. These SLAM keyframes are only added when enough distance or rotation has accumulated to warrant adding a new keyframe. However, sonar imagery is available at 5 hz, leaving most of the sonar image pairs unused in the above 3D mapping systems. This motivates our newest iteration on this system, building *submaps*. In this subsystem, we will retain the sonar imagery between keyframes to build denser, more detailed 3D maps without requiring the prior information in our inference based mapping system.

### 5.0.1 Submap Construction

Constructing submaps is a simple process using the dead reckoning pose $o_t$ and the fused sonar images $z^{Fused}$. We note that keyframes are discrete steps in time, denoted by iterator $k$. Once a keyframe is added, we log all the $z^{Fused}$ and their poses relative to the most recent keyframe. A submap at step $k$, $\mathcal{S}_k$ is defined as the set of observations between keyframes registered in the local reference frame

$$\mathcal{S}_k = \{\mathbf{T}_t^0 \hat{z}_0^{Fused}, \mathbf{T}_t^1 \hat{z}_1^{Fused}, \mathbf{T}_t^2 \hat{z}_2^{Fused}, ..., \mathbf{T}_t^N \hat{z}_N^{Fused}\}. \tag{5.1}$$

Note that the submap $\mathcal{S}_k$, contains $N$ the fused sonar observations, $\hat{z}^{Fused}$, where $N$ is the number of observations until the next keyframe is instantiated. Each of the $N$ observations, $\hat{z}_i^{Fused}$ has an associated transformation, $\mathbf{T}_t^i$, that registers $\hat{z}_i^{Fused}$

into the keyframe's local reference frame at step $k$, derived from the dead reckoning system providing position. When considering time synchronization issues, linear interpolation is used to solve for a transform between dead reckoning steps, if required. The *robot map*, $\mathcal{M}$ can be built from the submaps as

$$\mathcal{M} = \{\mathbf{T}_0\mathcal{S}_0, \mathbf{T}_1\mathcal{S}_1, \mathbf{T}_2\mathcal{S}_2, ..., \mathbf{T}_t\mathcal{S}_t, \}, \tag{5.2}$$

where $\mathbf{T}_k$ is from the SLAM based pose estimate, $x$. The key difference between submapping and the other mapping techniques in this paper is the collection and use of sonar data *between* the time discrete SLAM keyframes. Note, this system does require a source of accurate dead reckoning over the short term between keyframes.

This sets up the tradeoffs to be examined in the experiments section of this paper. Is a semantic labeling model available for the environment? Are repeating objects present? Does the vehicle has an accurate dead reckoning system? We will compare systems using these questions and their applicability in varying use cases.

## 5.1 Experiments

In this section, we will perform experimental validation and comparison of three versions of our mapping system. These systems will be referred to as below:

- **Sonar fusion mapping:** using only sonar image pairs at discrete timesteps as in chapter 3.
- **Inference Based Mapping:** using only sonar image pairs at discrete timesteps and inference on those timesteps, as in chapter 4
- **Submapping:** using sonar image pairs *at and between* timesteps as in this chapter.

We note that each method has its advantages, disadvantages, and requirements for operation. Sonar fusion mapping may provide inadequate detail due to its generally poor coverage. Inference based mapping improves on this but requires a trained model to segment sonar images. Moreover, inference based mapping's coverage rate is correlated with the number of simple, repeating objects in the environment, for example, circular pier pilings. Lastly, submapping will increase map coverage by retaining the data between timesteps but requires a dead reckoning system that's accurate between keyframes. A summary of operational requirements is shown in Table 5.1.

| System | Semantic Model | Repeating Objects for High Coverage | Short Term Dead Reckoning |
|---|---|---|---|
| Sonar Fusion Mapping | x | x | x |
| Inference Based Mapping | ✓ | ✓ | x |
| Submapping | x | x | ✓ |

Table 5.1: **Comparison of system requirements.** We compare our three mapping systems, each with its operational requirements. A check denotes the requirement of a category, while an x means it is not necessarily required.

### 5.1.1 Hardware Overview

In order to perform real-world experiments and derive a simulation environment for this work, we use our customized BlueROV2-heavy robot, shown in Fig. 3.1. This vehicle is equipped with an onboard Pixhawk, Raspberry Pi, and NVIDIA Jetson Nano for control and computation. We use a Rowe SeaPilot doppler velocity log (DVL), VectorNav VN100 inertial measurement unit (IMU), KVH DSP-1760 3-axis fiber optic gyroscope, and a Bar30 pressure sensor. We use a pair of wide aperture multi-beam imaging sonars for perceptual sensors, a Blueprint subsea Oculus M750d and M1200d. We use the M750d as our horizontal sonar and the M1200d as the vertical sonar. Note that the entirety of this work takes place with these sonars and their simulated versions operating at a range of 30 meters, with a 5cm range resolution.

In order to manage the BlueROV's sensors, SLAM system, and companion 3D mapping system, we use the Robot Operating System [93], both for operating the vehicle and for playback of its data. The 3D mapping algorithms are applied to real-time playback of our data using a computer equipped with an NVIDIA Titan RTX GPU and Intel i9 CPU. Note that **all** experiments take place at a fixed depth.

### 5.1.2 Simulation Study

In this section, we study our three mapping systems using simulated underwater environments. We utilize Gazebo [86] with UUV Simulator [87] to simulate the environment, vehicle, and sensors, including the wide aperture multi-beam imaging sonar [88]. Simulated environments are selected to capture variability in complex structures like ship hulls and aircraft and the number of repeating simple objects, such as round pier pilings. The four environments are described below:

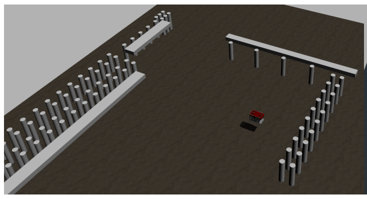
- **Simulated Marina 1:** a marina environment consisting of floating docks and circular pier pilings. Note the heavy presence of repeating objects; the circular pier pilings. Designed to be similar in appearance to SUNY Maritime college's marina in The Bronx, NY. The simulation environment is shown in Figure 5.1(a).

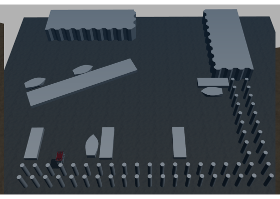
- **Simulated Marina 2:** a marina environment consisting of floating docks, small boats, circular pier pilings, and corrugated seawall. Again, note the heavy presence of repeating objects. Shown in Figure 5.1(b).

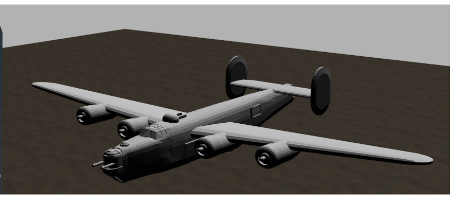
- **Simulated Ships harbor:** a harbor environment with floating docks, a long corrugated seawall, and two large vessels. One large sailing ship and a WWI-era cruiser. This environment only has some repeating structure. Designed to be similar in appearance to Penn's Landing Marina in Philadelphia, PA. The simulation environment is shown in Figure 5.1(d).

- **Simulated Aircraft Site:** a single large aircraft on the seafloor. Note that there is no repeating structure. Moreover, this is the most complex geometry we have examined to date. The simulation environment is shown in Figure 5.1(c).

Since our systems utilize discrete SLAM keyframes in one way or another, which are added by distance/rotation threshold, we vary this distance and rotation denoted as keyframe density. It is critical to vary keyframe density since this directly influences mapping coverage. In our work, we test combinations of keyframe distance and rotation. Keyframe distance [1,2,3,4,5] meters and keyframe rotation [30,60,90] degrees. Note that we test all 15 combinations of distance and rotation.


(a) **Simulated Marina 1**


(b) **Simulated Marina 2**


(c) **Simulated Plane**

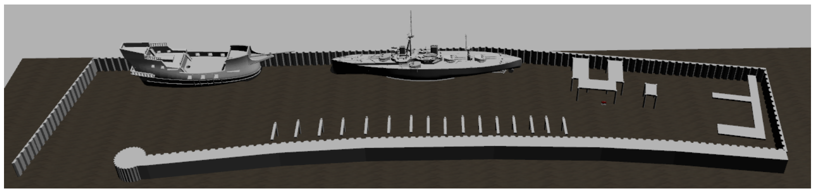

(d) **Simulated harbor**

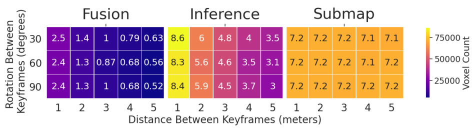
Figure 5.1: **Simulation Environments.**

To study the performance of each method, we consider several metrics. Firstly we define a coverage metric, the total number of voxels occupied when the point cloud map is discretized. We use voxel count as a coverage metric to avoid double-counting redundant data in the same location. Pointcloud maps are voxelized using a 0.1-meter voxel size. Next, we can quantify pointcloud map error using the environment CAD model. Here we consider the absolute distance between a point in the map and the CAD model. Accuracy is reported using mean-absolute-error (MAE) and root-mean-squared-error (RMSE) to characterize the error distribution's mean and size.

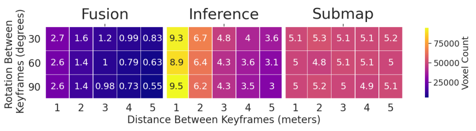
#### 5.1.2.1 Simulated Marina 1

Firstly we consider simulated marina 1. Recall that this environment contains floating docks and many repeating circular pier pilings, shown in Figure 5.1(a). Figure 5.2(a) shows a summary of coverage, measured in voxel count. Qualitative sonar fusion results are shown in Figure 5.3(a), inference based mapping results are shown in Figure 5.3(b), and submapping is shown in Figure 5.3(c). For sonar fusion mapping, we note the lowest coverage of the three methods, with coverage increasing as keyframe density increases, left and up in figure 5.2(a). When considering inference based mapping, the center of figure 5.2(a), we again note increased coverage with increased keyframe density and generally improved coverage compared to sonar fusion mapping, due to the prevalence of simple, repeating objects. However, we also note that inference based mapping has higher coverage than submapping when the distance between keyframes is 1 meter. When considering submapping, on the right of figure 5.2(a), coverage is generally flat across keyframe density, and coverage is higher than other methods, except when the inference based mapping system is applied in the densest keyframe category.

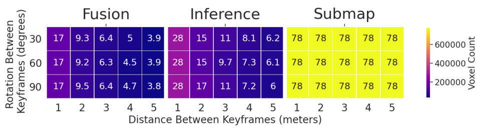
Again, we consider mapping error by comparing the point cloud map to the simulation environment CAD file. Mapping error is reported in table 5.2. We note that mean mapping error is similar across all three methods in simulated marina 1,



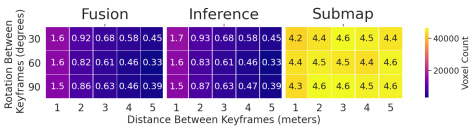
(a) **Simulated Marina 1**



(b) **Simulated Marina 2**



(c) **Simulated harbor**



(d) **Simulated Plane**

Figure 5.2: **Simulation coverage results.** Coverage in both table and color format. Each cell reports voxel count, with color mapping from blue to orange as low to high coverage. The vertical axis shows the varying keyframe rotations in degrees, with the horizontal axis showing the Euclidean distance between keyframes in meters. Each system type is shown in this figure, with sonar fusion mapping at left, inference based mapping at center and submapping at right.

with submapping having a slightly smaller RMSE.

#### 5.1.2.2 Simulated Marina 2

Next, we consider simulated marina 2, shown in Figure 5.1(a). Recall that this environment contains floating docks, small boats, many repeated circular pilings, and corrugated seawalls. Coverage results for simulated marina 2 are shown in Figure 5.2(b). Qualitative results are shown in Figures 5.3(d), 5.3(e) and 5.3(f). Again we see a similar trend; sonar fusion mapping shows the least coverage, with inference based mapping improving due to the repeating objects in the environment. However, inference based mapping only improves coverage over submapping when keyframes are 2 or fewer meters. Submapping again shows a generally flat performance in all keyframe densities. When considering mapping accuracy, we note comparable MAE, with slight RMSE change when moving to submapping. Simulated marina 2 accuracy results are shown in figure table 5.2.

#### 5.1.2.3 Simulated Ships harbor

This simulated ship's harbor contains only a few repeating objects, floating docks, a long corrugated seawall, and two ships, shown in Figure 5.1(d). Qualitative results are shown in Figures 5.5(d), 5.5(e) and 5.5(f). When considering the coverage in this environment, shown in Figure 5.2(c), we note a smaller coverage gap between sonar fusion mapping and inference-based mapping due to the reduced prevalence of simple repeating objects for the inference base mapping system to leverage. Additionally, we note that submapping outperforms inference based mapping in all keyframe density cases. The performance difference between submapping and inference based mapping is likely a combination to two key reasons—first, the lack of repeating objects in the environment. Second, the complex geometry of ship hulls is difficult to map using only keyframes; submapping uses the data *at and between* the keyframes, enabling denser mapping of complex structures. When considering the simulated harbor environment error reported in table 5.2 we note similar means, with submapping having a larger RMSE.

#### 5.1.2.4 Simulated Aircraft Site

This environment consists of one large structure, a WWII-era B-24 Liberator sitting on the seafloor, shown in Figure 5.1(c). We use this structure as it presents highly complex geometry not previously analyzed with our mapping systems. Further, it does contain *any* repeating objects, only one large structure, the aircraft. This specific aircraft is selected due to its distinctive dual tail arrangement, which, if mapped correctly, will be easily recognizable. Qualitative results are shown in Figures 5.3(g), 5.3(h) and 5.3(i). Coverage results are shown in Figure 5.2(d). Due to the total lack of repeating objects in this environment, there is a negligible coverage difference comparing the inference-based mapping system to the sonar fusion map. When considering the submapping system, Figure 5.2(d) shows a coverage improvement compared to the other two systems. Most importantly though, when considering the qualitative results in Figure 5.3(i), there is a more complete representation of the aircraft's main wing and tail section. Regarding map accuracy, shown in table 5.2, we note similar means with the submapping RMSE slightly larger.

| | Environment | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Marina 1 | | Marina 2 | | Harbor | | Plane | |
| System | MAE | RMSE | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| Sonar Fusion Mapping | 0.16 | 0.21 | 0.19 | 0.25 | 0.21 | 0.30 | 0.17 | 0.24 |
| Inference Based Mapping | 0.16 | 0.20 | 0.17 | 0.22 | 0.22 | 0.30 | 0.17 | 0.24 |
| Submapping | 0.14 | 0.17 | 0.16 | 0.19 | 0.26 | 0.37 | 0.21 | 0.28 |

Table 5.2: **Simulated mapping accuracy in meters.** We report aggregated results for each system in each environment. MAE and RMSE are both reported in meters.

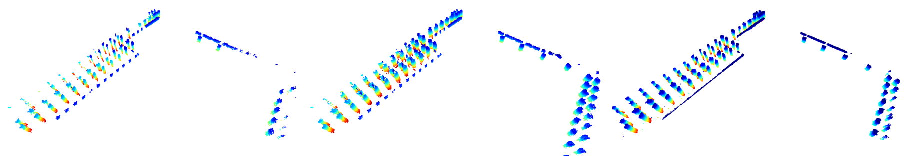
| | Environment | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Marina 1 | | Marina 2 | | Harbor | | Plane | |
| System | Mean | SD | Mean | SD | Mean | SD | Mean | SD |
| Sonar Fusion | 0.088 | 0.018 | 0.080 | 0.011 | 0.086 | 0.014 | 0.075 | 0.010 |
| Mapping Inference | 0.617 | 0.267 | 0.529 | 0.387 | 0.277 | 0.172 | 0.233 | 0.193 |
| Submap Construction | 0.005 | 0.009 | 0.001 | 0.003 | 0.006 | 0.010 | 0.001 | 0.003 |

Table 5.3: **Runtime in seconds.** We report mean and standard deviation (SD) runtime for all experiments for a particular system in each environment. Sonar fusion refers to the proccess of fusing a single sonar image pair. Mapping inference is the time required to apply the method in chapter 4 to a single keyframe. Submap construction is the time required to assemble a submap from a single timestep once all the data is collected.

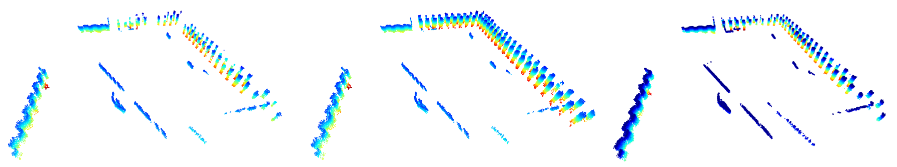
#### 5.1.2.5 Summary of Simulation Results

The results of the simulation study can be summarized as follows. Firstly the sonar fusion mapping system provides accurate maps but potentially low coverage, especially when keyframe density is sparse. However, the sonar fusion mapping system has the least system requirements of the three compared methods, requiring no trained semantic model, no repeating objects to enhance coverage, and no short term dead reckoning system. Next, the inference based mapping system makes coverage improvements in three out of four environments compared to sonar fusion mapping. We note that this increased mapping coverage is due to the presence of simple repeating objects to perform object-specific inference using a trained semantic labeling model. Lastly, submapping only outperforms inference based mapping when there are few if any, repeating objects in the environment. However, submapping recovers reasonably accurate, high-coverage maps in cases where the structures are complex. It is essential to note the value of submapping when potentially deploying an autonomous system to an unknown environment. Dense, 3D maps can be recovered with *no prior information* regarding the environment. Moreover, in the case of complex 3D geometry, parts of the structure can be mapped that the other systems in this paper miss, such as the wing of the airplane shown in Figure 5.3(i).

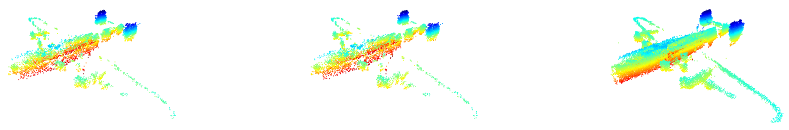
When considering runtime, reported in table 5.3, we provide both mean and standard deviation (SD) in seconds. In the row marked “orthogonal sensor fusion” in table 5.3, we provide the time required to fuse a single pair of sonar images, not build a map. We note that this runtime is sufficient to keep pace with the simulated sonar sensor, which runs at 5hz (0.2 seconds). The row marked “Mapping inference” in table 5.3 records the time required to apply the inference based mapping system to a single keyframe. We note that runtime for this category is slower when repeating objects are present. Moreover, a key detail some readers may note is that we do not apply the inference systems to data between keyframes. While the runtime is sufficient to keep up with the addition of keyframes, applying this method to the $N$ sonar image pairs between keyframes would be impractical. Lastly, when considering the runtime of the "submap construction" row in table 5.3, this is the required time to construct a submap once all the data is collected between keyframes. We note that this process is low runtime and does not add undue computation to our system.



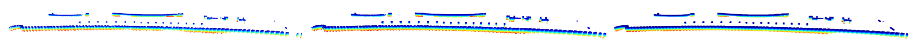
(a) **Marina 1 SF** (b) **Marina 1 IB** (c) **Marina 1 S**



(d) **Marina 2 SF** (e) **Marina 2 IB** (f) **Marina 2 S**



(g) **Plane SF** (h) **Plane IB** (i) **Plane S**

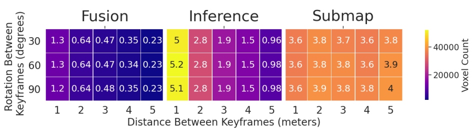


(j) **harbor SF** (k) **harbor IB** (l) **harbor S**

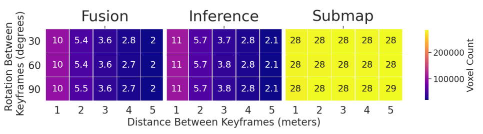
Figure 5.3: **Simulation Environment Mapping Results.** Each column shows one of our three methods. Sonar fusion mapping (SF), inference based mapping (IB) and submapping (S).

### 5.1.3 Real World Study

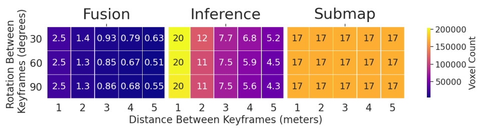
This section will showcase three experiments with real-world data collected with our robot. In this real-world case study, ground truth data is unavailable, so we will analyze only coverage and runtime using the same means as the simulation study. The three environments are SUNY Maritime College's marina in The Bronx, NY, and Penn's Landing marina in Philadelphia, PA and a marina adjacent to Norfolk Naval base in Norfolk, VA. SUNY Maritime is shown in Figure 5.6(a) and from satellite view in Figure 5.6(b), note the heavy presence of circular pilings, repeating simple objects. Penn's landing is shown in Figure 5.6(c) and from satellite view in Figure 5.6(d), note the presence of several large complex structures and few repeating objects. Norfolk is shown in Figure 5.6(e) with satellite view show in Figure 5.6(f). Note that this environment has two long edge structures, the concrete wall on the left of the marina and the long floating dock on the bottom, with the rest of the structure being repeating pier pilings.



(a) **SUNY Maritime College**

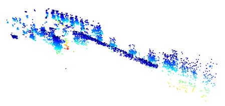


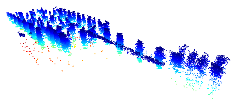
(b) **Penn's Landing**

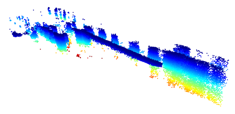


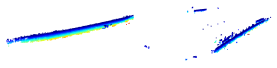
(c) **Norfolk**

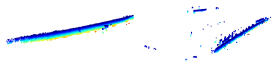
Figure 5.4: **Real world coverage results.** Coverage in both table and color format. Each cell reports voxel count, with color mapping from blue to orange as low to high coverage. The vertical axis shows the varying keyframe rotations in degrees with the horizontal axis showing Euclidean distance between keyframes in meters. Each system type is shown in this figure, with sonar fusion mapping at left, inference based mapping at center and submapping at right.

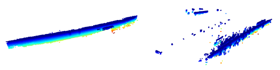

(a) **SUNY Maritime SF**

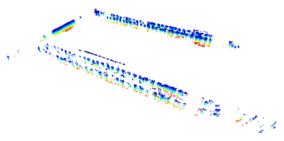

(b) **SUNY Maritime IB**

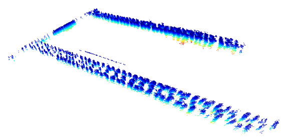

(c) **SUNY Maritime S**


(d) **Penn's Landing SF**


(e) **Penn's Landing IB**


(f) **Penn's Landing S**


(g) **Norfolk SF**


(h) **Norfolk IB**

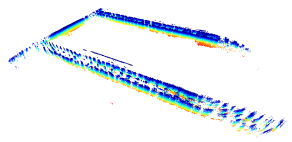

(i) **Norfolk S**

Figure 5.5: **Real World Results.** Each column shows one of our three methods. Sonar fusion mapping (SF), inference based mapping (IB) and submapping (S).

#### 5.1.3.1 SUNY Maritime Results

At SUNY Maritime college, we see a coverage trend consistent with our simulation experiments, shown in Figure 5.4(a). Qualitative results are shown in Figures 5.5(a), 5.5(b) and 5.5(c). Inference based mapping makes significant coverage improvements over fusion based mapping. Moreover, submapping shows good coverage, better than fusion based mapping. However, inference based mapping shows better coverage results than submapping with its densest set of keyframes. Note that SUNY Maritime has very little complex structure and is characterized by many repeating pier pilings, an excellent use case for inference based mapping.

#### 5.1.3.2 Penns Landing Results

Penn's Landing has very few repeating objects but more complex structure and our trajectories include a ship hull and barge. Qualitative results are shown in Figures 5.5(d), 5.5(e) and 5.5(f). Figure 5.4(b) shows coverage results from Penn's Landing. We note the small performance improvement in inference mapping compared to fusion based mapping, likely due to the small number of repeating objects. Submapping shows the highest coverage in this environment and is flat across varying keyframe densities. Note the dense point clouds recovered from the barge and ship in Figure 5.5(f).

#### 5.1.3.3 Norfolk Results

Norfolk shows a similar trend to SUNY Maritime college and our simulation experiments, as shown in Figure 5.4(c). The presence of repeating objects means inference based mapping is able to recover the most dense version of the map, but submapping is consistent across all parameterizations. We especially note the increase in coverage on the concrete wall at the back of the map when using submapping, shown in 5.5(i). We do note some rotational distortion, likely due to gyroscope error.

#### 5.1.3.4 Real World Runtime

Table 5.4 shows the runtime for these real-world datasets. Submap construction again does not add undue computational burden to our system, and performing inference on keyframes takes time, making it clear that inference should only be applied to keyframes. Orthogonal sensor fusion time runs faster or near the sonar refresh rate of 5hz.

#### 5.1.3.5 Real World Results Summary

These real-world datasets make the results of this study clear; when repeating objects are present, and a semantic model can be provided to leverage those objects, inference based mapping provides excellent coverage. However, when a model is unavailable or when these repeating objects are not prevalent, submapping provides better coverage, especially when considering complex objects like ship hulls. However, sonar fusion mapping provides adequate results when neither a model nor a short-term dead reckoning system is available.

| | Environment | | | |
|---|---|---|---|---|
| | SUNY Maritime | | Penn's Landing | |
| System | Mean | SD | Mean | SD |
| Orthogonal Sensor Fusion | 0.172 | 0.030 | 0.207 | 0.024 |
| Mapping Inference | 0.947 | 0.251 | 0.405 | 0.253 |
| Submap Construction | 0.003 | 0.005 | 0.003 | 0.003 |

Table 5.4: **Real world runtime in seconds.** We report mean and standard deviation (SD) runtime for all experiments for a particular system in each environment. Orthogonal sensor fusion refers to the proccess of fusing a single sonar image pair. Mapping inference is the time required to apply the method in chapter 4 to a single keyframe. Submap construction is the time required to assemble a submap from a single timestep once all the data is collected.

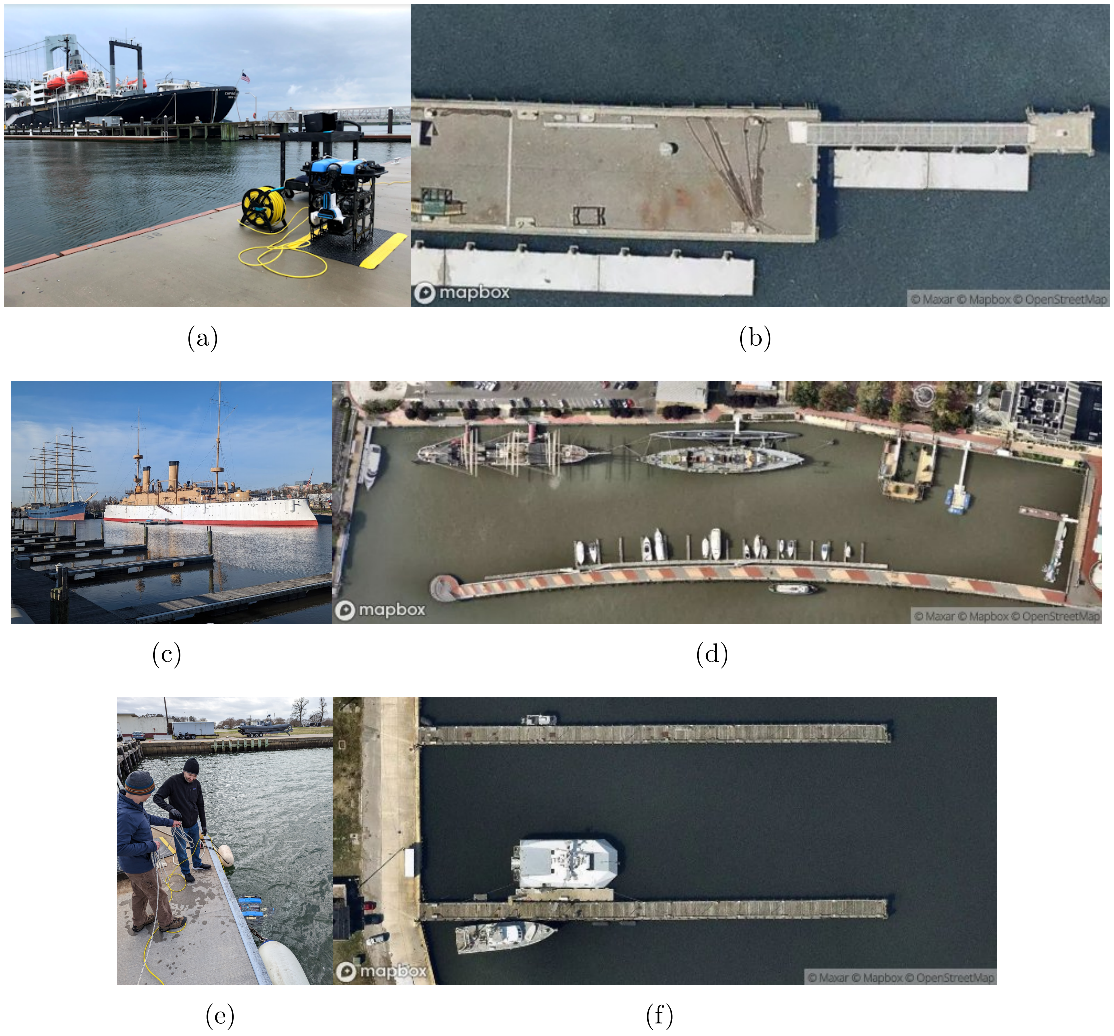


Figure 5.6: **Real world environments.** (a) shows our experimental setup at SUNY Maritime college. (b) shows a satellite image of (a). (c) shows a ground level view of Penn's landing with (d) showing a satellite view. (e) shows a ground level view of the marina in Norfolk, VA with (f) showing a satellite image view.

# Chapter 6

# Overhead Image Factors for Underwater Sonar-based SLAM

## 6.1 Problem Description

We formulate a three degree-of-freedom planar SLAM problem across discrete time steps $t$, each with an associated pose $x_t$. We define each pose in the plane as

$$x_t = \begin{pmatrix} x \; y \; \theta \end{pmatrix}^\top . \tag{6.1}$$

Each pose has a set of observations $z_t$. The observations include sonar returns in spherical coordinates with range $R \in \mathbb{R}_+$, bearing $\theta \in \Theta$, and elevation $\phi \in \Phi$, with $\Theta, \Phi \subseteq [-\pi, \pi)$, and an associated intensity value $\gamma \in \mathbb{R}_+$. These observations can be mapped into Cartesian space by

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R \begin{pmatrix} \cos\phi\cos\theta \\ \cos\phi\sin\theta \\ \sin\phi \end{pmatrix} . \tag{6.2}$$

Additionally, each set of observations $z_t$ includes odometry information: linear velocity $V \in \mathbb{V}_+$, linear acceleration $A \in \mathbb{A}_+$ and rotational velocity $W \in \mathbb{W} \subseteq [-\pi, \pi)$.

The robot moves through the environment at fixed depth, transitioning from state to state according to the dynamics

$$x_t = g(u_t, x_{t-1}) + \delta_t, \tag{6.3}$$

where $x_{t-1}$ is the pose at the previous timestep, $u_t$ is the actuation command and $\delta_t$

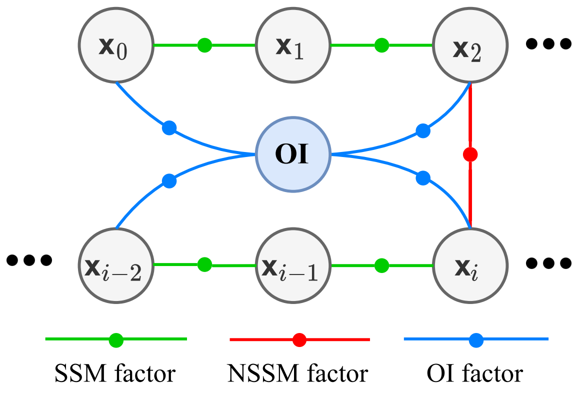


Figure 6.1: **SLAM Factor Graph.** Robot poses $x$ are connected by three kinds of factors: sequential scan matching factors (SSM) in green, non-sequential scan matching factors (NSSM, or loop closures) shown in red and overhead image factors (OI) shown in blue.

is process noise. The posterior probability over the time history of poses is defined as

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}), \tag{6.4}$$

with map $m$. The question addressed in this work, is how we can leverage the provided observations, in conjunction with reasonable prior information, to improve state estimation.

## 6.2 Proposed Algorithm

### 6.2.1 Sonar Image Processing

As described in Section III, each sonar observation contains a set of returns of varying intensity, in spherical coordinates, which are recorded in a 2D image. However, we must identify which points in the image are true sonar contacts and which are noise and/or second returns.

In this work, we utilize constant false alarm rate (CFAR) detection [76] to identify returns from the surrounding environment in the sonar imagery. CFAR has

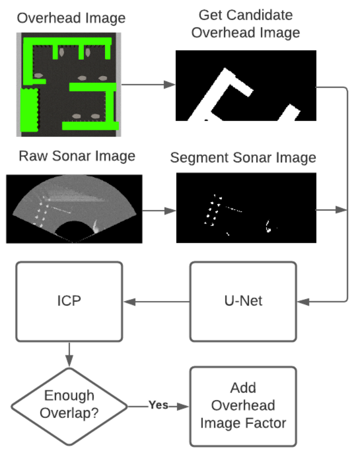
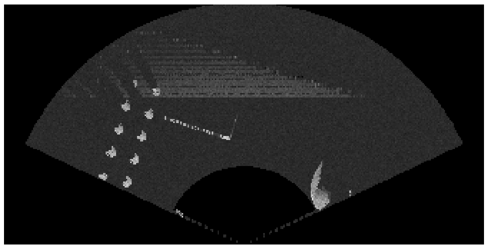


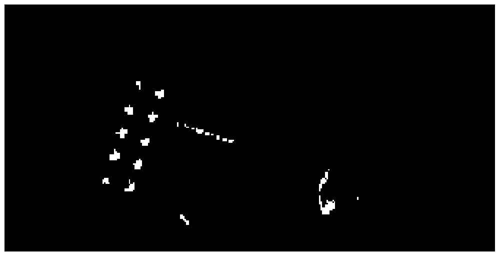
Figure 6.2: **System block diagram.** The overhead imagery is segmented offline, before the mission. As the mission progresses, a candidate overhead image is retrieved using the current state estimate. Raw sonar images are segmented using CFAR detection. Candidate overhead and segmented sonar images are presented in a query to U-Net. ICP is then applied, to register U-Net's synthetic image output to the globally referenced candidate overhead image. If there is enough overlap, the measurement is added to the factor graph.

been widely successful in radar and acoustic image processing [77], [78], and has supported our prior work. Specifically, we use the smallest-of cell averaging (SOCA) variant of CFAR, which takes local area averages around the pixel in question and produces a noise estimate. If the signal is greater than a designated threshold, the pixel is identified as an image-feature. An example of its output is shown in Fig. 6.3(b); this perceptual data product is used to support SLAM throughout the work described in this paper. This is described in more detail in section 3.3.1.

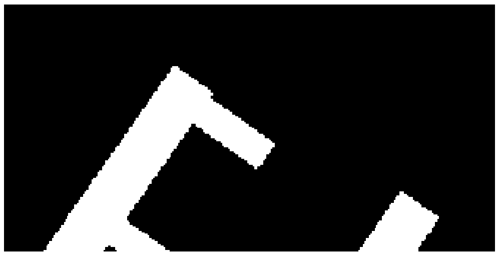
All CFAR-extracted points in the image are converted to Cartesian coordinates (Eq. (7.2)). Because we confine our state to the plane, and because the elevation angle $\phi$ in Eq. (7.2) is not recorded in our sonar imagery, we set $\phi$ as zero.



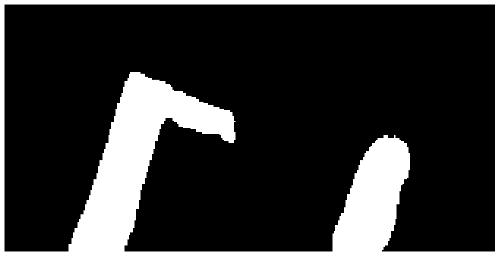
(a) Simulated Sonar Image



(b) CFAR Image from (a)



(c) Candidate Overhead Image



(d) Example Predictions

Figure 6.3: **Example Images:** (a) A sonar image from our simulation environment, (b) the CFAR-segmented sonar image, (c) the corresponding candidate overhead image, and (d) the prediction of (b)'s appearance at the surface, from U-Net's processing of (b) and (c).

### 6.2.2 Overhead Image Processing

We segment the available overhead imagery into three classes: water, structures, and vessels. We then generate a binary mask of the overhead imagery; pixels comprising static structures are set to 1, and all others (water and vessels) are set to 0. An example of this mask is shown in green at the top left of Fig. 6.2. At each time-step, we carve out of the binary mask a footprint representing the sonar field of view at the respective pose estimate, shown at top right of Fig. 6.2, and Fig. 6.3(c). This is performed using the current SLAM solution, in conjunction with the robot's known initial pose at the outset of the mission, to estimate the robot's current pose within the overhead imagery. The resulting image is referred to in this paper as the *candidate overhead image*.

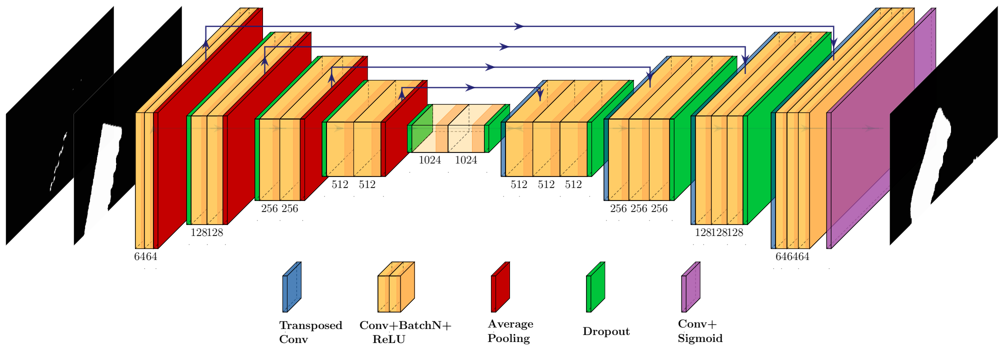


Figure 6.4: **Network Overview.** We utilize a U-Net architecture to generate synthetic overhead imagery predicting the above-surface appearance of an input sonar image's contents. The inputs to the network are a processed sonar image containing CFAR-detected features, and a candidate (partially overlapping) overhead image, each with the same spatial area. Input images are 256x128 pixels.

### 6.2.3 Learning a Unified Representation

Key challenges to overcome in registering a sonar image to a candidate overhead image are the differences in perspective, contents, and modality. In general, an environment contains two kinds of structures: static structures, such as walls and docks, and dynamic structures, such as boats, which even when stationary, may be docked in different locations at different times. In this work, we assume dynamic structures may have moved between the time the overhead image was captured and the time of the mission, but not during the mission. Moreover, while overhead images contain RGB channels, sonar images only contain acoustic intensity, not necessarily related to the RGB channels of a given object.

These fundamental differences are what make registering a sonar image to an overhead RGB image challenging. For this reason, we propose to learn a unified representation that will aid in the registration problem. The first of our inputs is the candidate overhead image, with an example shown in Fig. 6.3(c). The second input is the sonar CFAR image, with an example shown in Fig. 6.3(b). These two images are stacked channel-wise into a single image as shown at left in Fig. 6.4. Note that these images may have a non-zero transformation between them, as the state estimate may or may not have drifted. We would like to emphasize that their initial alignment does not need to be perfect, but simply provide reasonable overlap between the sonar image and candidate overhead image. The network output is the static structure observed in the overhead image transformed to the sonar image's frame; a synthetic overhead image. We utilize a well-known network architecture, U-Net [89], to learn the mapping between input and output with the addition of dropout layers. The key advantage of this method is learning a unified representation (a prediction of the above-surface appearance of the structures observed by the sonar) that can be registered to overhead imagery, offering the same perspective, contents and appearance as overhead imagery.

### 6.2.4 Acoustic Image Registration

We can now attempt to register the network output, a synthetic overhead image, to the candidate overhead image. To do so we first convert the images from pixels to meters. Next, we extract the outline of any structure present and apply voxel-based down-sampling to reduce the number of points. We then use ICP to estimate the transformation between the candidate overhead image points and the predicted synthetic overhead image points.

The resulting transformation is used to estimate the overlap between the two sets of points; if the overlap does not exceed a designated threshold, it is considered to be a bad registration and discarded. Overlap is evaluated by applying nearest-neighbor association and counting the percentage of points in the source cloud with a Euclidean distance of less than one meter to their neighbor. Note that here we do not apply PCM as a means of outlier rejection, as it is simply not required; the level of noise and ambiguity is not as severe as for standard loop closure detection. If the overlap is sufficient, then the ICP-derived overhead image transformation is added to the current SLAM state estimate to derive the transformation between the initial pose and the current pose. We then add this *overhead image factor* to our factor graph, linking the initial and current pose. For the sake of generality, the factor graph in Fig. 7.2 shows an overhead image frame that can be associated with any point of reference, not just the initial robot pose.

## 6.3 Experiments and Results

### 6.3.1 Data Gathering

To train our CNN, a significant amount of labeled data is required. While the authors of [46] and [47] use a sonar dataset with known locations from GPS, this data is not fully publicly available and is limited in scope, as it comes from a single environment with few dynamic structures. We perform a simulation study due to the lack of available data from underwater settings with ground truth, and our inability to perform new large-scale field experiments during the COVID-19 pandemic. Using a simulator allows us to generate a diverse body of relevant labeled imagery for our CNN, and to evaluate SLAM performance, as ground truth is known. Moreover, we will later demonstrate our algorithm's applicability to real-world data after being trained in simulation. Future work will focus on validating this methodology on real data from complex trajectories in large-scale environments with ground truth.

To generate training data we use two environments, one resembling a marina (Fig. 6.6(g)), and another of a ship's pier (Fig. 6.6(h)). To generate validation data, and to evaluate our SLAM solution, we design two other marina-like environments different from the first, shown in Figures 6.6(i) and 6.6(j). To gather data, we place the vehicle at a random, collision-free pose and capture a sonar image. We then generate a random transformation in-plane, with translations up to 5 meters and $\pm$ $22^\circ$ yaw. We then capture a candidate overhead image at the random transformed pose. Labels are generated by capturing a candidate overhead image at the same pose as the sonar image. We generate 5000 samples from each training environment and 2500 samples from the validation environment in Fig. 6.6(i) for ROC curve analysis. The environment in Fig. 6.6(j) is used separately to evaluate long-duration SLAM performance.

Each training dataset requires a segmented overhead image of the full environment, for generating candidate overhead images as described in Section 5.2.2. Overhead images of each environment are generated at an altitude of 60 meters using a simulated camera. The aforementioned overhead image segmentation (structure, water, vessels) is generated by hand labeling, as shown at the top left of Fig. 6.2. It may be desirable for this step in the process to be fully automated, but we also contend that with environments of this size, it may often be a reasonable cost to human AUV operators in the mission planning step, especially for settings that will be visited frequently. Moreover, deep learning methods for image segmentation in this setting are growing in maturity [90]. Therefore, we confine the technical scope of this paper to include pre-segmented overhead imagery, as its preparation is simply a part of the mission planning process (automated or not), and not an integral part of the live functionally of the proposed algorithm.

### 6.3.2 CNN Training

Using the gathered dataset, we implement the model architecture illustrated in Fig. 6.4 using TensorFlow [91]. The model is trained for 10 epochs using a categorical cross-entropy loss function and the Adam optimizer [92]. We evaluate our model

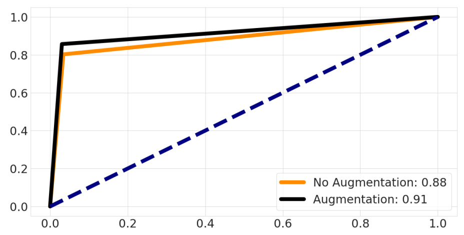


Figure 6.5: **Training ROC Curves.** Orange shows validation results without data augmentation; black shows validation results with data augmentation; the legend reports the area under the curve (AUC).

using the validation dataset. We note that our validation environment (pictured in Fig. 6.6(i)) has a difference in static structure, and its vessels are all in new locations, relative to the training environments. To quantitatively measure model performance we use receiver operator characteristic (ROC) curves, shown in Fig. 6.5. To increase validation performance, we perform two types of simple data augmentation. First, we randomly flip each set of images. Secondly, we introduce Gaussian speckle noise to the set of input images. More advanced data augmentation (rotations, etc.) is not used to preserve the contents of each data point, as our imagery is often sparse on the sonar side.

### 6.3.3 SLAM Performance Metrics

To evaluate the performance of a SLAM framework aided by overhead image factors, we consider two key metrics, mean absolute error (MAE) and root mean squared error (RMSE). These metrics are chosen to not only characterize the mean of the error distributions but the goodness of fit the SLAM solution provides with respect to the true trajectory. We consider both the Euclidean distance of our SLAM estimate from the corresponding true pose, and the difference in estimated yaw angle $\theta$ from that of the true pose. Note that we do not consider the global pose, but the pose relative to the initial keyframe, ensuring a fair comparison across competing algorithms. In the results to follow, sonar imagery is sampled at 5Hz, and as in [71], sonar keyframes are generated upon every 2m of translation or $30^\circ$ of rotation.

### 6.3.4 Simulated SLAM Results

In addition to sonar imagery, our SLAM method requires a source of dead reckoning, in this case, a simulated DVL and IMU. Zero mean Gaussian noise is introduced into these sensors with $\sigma_{velocity}$ and $\sigma_{theta}$ as 0.1 m/s and $1^\circ$ respectively. These values are chosen as they produce results similar to those experienced in the field with our baseline method. Lastly, the proposed method requires a robot position fix defined within the overhead imagery. We denote this as the overhead image frame, which we choose to coincide with the location of the robot's initial pose, defined with respect to the overhead imagery. Each mission is initialized by applying zero-mean Gaussian noise to the true initial pose using $\sigma_{theta}$, and $\sigma_{xy}$ of 1 meter. Note that in a real world mission, we believe this information could be easily sourced by collecting an initial GPS fix at the surface, at the beginning of a mission. Recall that we use point-cloud overlap as a means of outlier rejection, and for this simulation study, the minimum required overlap to accept an overhead image factor is 80%.

Equipped with a trained CNN, we can apply our full proposed pipeline to the validation environment of Fig. 6.6(i). To evaluate our SLAM system we generate two "fly-through" trajectories at a fixed depth of approximately 1 meter; one transits from left to right and the other from right to left. This style of trajectory is selected since it efficiently transits the environment, while observing nearly all structures in the marina along the way. Using these two trajectories, we evaluate our proposed SLAM method as well as a baseline method, which is simply our SLAM method without the addition of overhead image factors.

Secondly, in order to evaluate the efficacy of our algorithm over long distances,

| Metric | Fly Through | | Long Distance | |
|---|---|---|---|---|
| | Proposed | Baseline | Proposed | Baseline |
| Euclidean Error | | | | |
| MAE (meters) | **0.83** | 2.06 | **0.87** | 3.13 |
| RMSE (meters) | **0.95** | 2.29 | **0.93** | 3.17 |
| Yaw Angle Error | | | | |
| MAE (degrees) | **1.70** | 5.95 | **3.65** | 3.98 |
| RMSE (degrees) | **2.09** | 6.48 | 10.5 | **10.27** |

Table 6.1: **SLAM Results.** MAE and RMSE. "Fly through" refers to the two benchmark trajectories in validation environment one as in Fig. 6.6(i). "Long distance" refers to the 2.7km transit through validation environment two shown in Fig. 6.6(j).

we run a 2.7-kilometer "long distance" trajectory using the simulation environment shown in Fig. 6.6(j). This trajectory is generated by looping around the environment ten times. With overhead image factors, an improvement is shown in position accuracy, with a comparable yaw angle to the benchmark; results are summarized in Table 6.1. We note that no modifications are made to our implementation when running the long-distance trajectory experiment, and online performance is still achieved.

### 6.3.5 Real World SLAM Results

While we confine our quantitative case study to simulation, we demonstrate the promise of future work by testing our system on real data. Due to restrictions resulting from the COVID-19 pandemic, gathering a new large-scale, ground-truthed dataset for this study was not possible; instead, previously-gathered datasets were utilized. These feature data collected with our custom BlueROV2 in two marina environments: the SUNY Maritime College marina on the East River in The Bronx, NY and the U.S. Merchant Marine Academy (USMMA) on the Long Island Sound in Kings Point, NY. Our training maps, shown in Figs. 6.6(g) and 6.6(h), are intended to capture the general appearance of these environments. These real-world environments pose several challenges to our overhead image factor system. Firstly, we train in simulation, and it is unknown how well our trained CNN will generalize to the real world. Secondly, the simulator, while realistic, is a controlled environment. Moreover, our real-world settings present environmental challenges, such as currents up to 2kts.

The datasets are recorded at a fixed depth of approximately 2m for SUNY Maritime and 1m for USMMA. Overhead imagery is hand segmented and provided to the system a priori, in addition to the robot's initial location in the imagery. The initial location was determined by manual alignment. Again we use point-cloud overlap as a means of outlier rejection, and for these field datasets, the minimum required overlap to accept an overhead image factor is 95%.

At SUNY Maritime, we consider an "out and back" trajectory where the robot moves away from its starting location and returns along a similar path within close proximity of the starting location. Baseline results for SUNY Maritime are shown in Fig. 6.6(d). Although our system was trained in simulation, it contributes several overhead image factors, mitigating the drift compared to the baseline method as shown in Fig. 6.6(c). The baseline method mainly shows drift in $\theta$, and even though several loop closures occur, error still accumulates in the map.

At USMMA, we consider a different trajectory, a straight-line transit where no loop closures occur in the standard SLAM solution. We consider this type of trajectory to demonstrate the utility of overhead image factors, which can correct drift even when loop closures are unavailable. The drift of the baseline trajectory, shown in Fig. 6.6(f), results in a highly inaccurate map. Once again our system, trained only in simulation, can contribute several overhead image factors that improve the state estimate, as shown in Fig. 6.6(e).

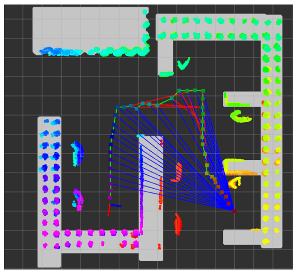
We wish to underscore the significance of these results and the *potential* of the proposed framework. Although the system was trained in simulation, it can, and in these two environments, does generalize to real sonar data. However, we are keenly aware of the challenges associated with deploying deep learning methods in the field. Future work will focus on expanding our methodology and testing on more significant, more complex trajectories and developing a ground-truthed dataset.

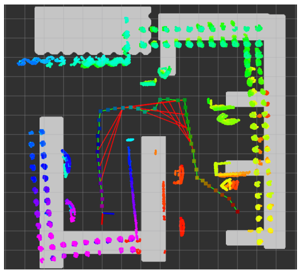
### 6.3.6 Computation Time

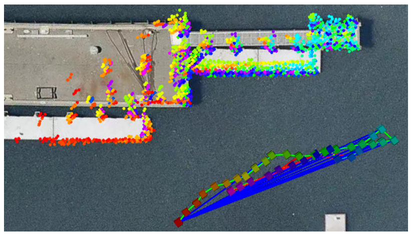
Our framework results in the following sources of additional computational overhead: a single GPU query, a single ICP query, and a single instance of overlap estimation per keyframe, which are implemented in a multi-threaded manner. If an overhead image factor is detected, the result is passed to the SLAM thread, where it is added to the factor graph. The result is a SLAM system that runs faster than keyframes are added. A runtime test of our overhead image factor system, which encompassed network prediction, ICP, and overlap estimation, yielded a rate of 55Hz, which is above and beyond the requirements of sonar keyframe processing. This test was performed over the 2500 examples from our validation set. These experiments were conducted using an Intel i9-9900K CPU @ 3.60GHz and an Nvidia Titan RTX GPU.

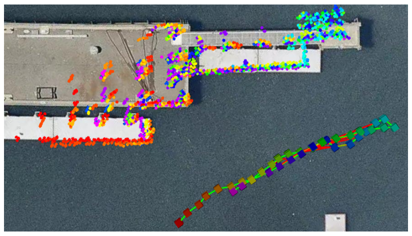
## 6.4 Conclusions

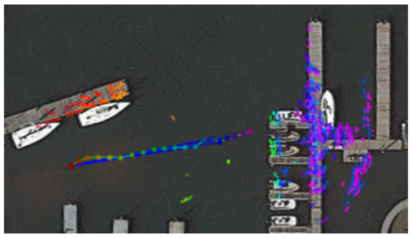
In this work, we have presented a novel method for addressing drift in a sonar-based underwater SLAM solution, using overhead image factors. We have shown that our method provides significant added value in terms of pose estimation error. Moreover, we do not use complex prior information such as environment CAD models. Instead we use data that can be sourced from the public domain, and an initial position fix. Additionally, we do not attempt to localize a robot using only an overhead image; we use all of the available information in conjunction with state-of-the-art sensor fusion methods [80] to build a robust and low drift state estimate. We do concede that this method is mostly applicable in littoral environments, where structures are observable above and below the water. However, we would contend that this represents many AUV/ROV applications, such as hull inspection, harbor security, and operating near offshore structures. In this work, we train and evaluate primarily in simulation. Future work will focus on developing the data (simulation and real) to extend this work to more extensive, outdoor, real world environments in the littoral zone.

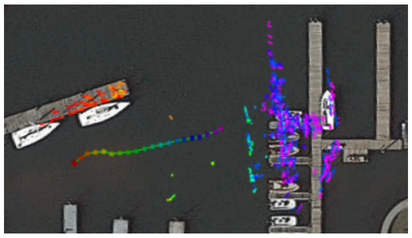

(a) Example simulated SLAM mission with overhead image factors

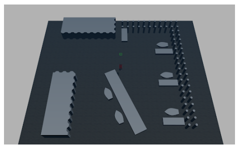

(b) Example simulated SLAM mission without overhead image factors

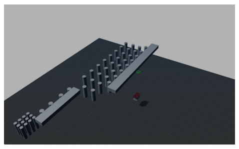

(c) Example real-world SLAM mission with overhead image factors at SUNY Maritime

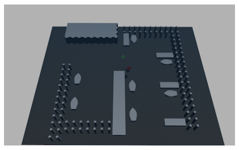

(d) Example real-world SLAM mission without overhead image factors at SUNY Maritime


(e) Example real-world SLAM mission with overhead image factors at USMMA


(f) Example real-world SLAM mission without overhead image factors at USMMA


(g) Training environment one


(h) Training environment two


(i) Validation environment one

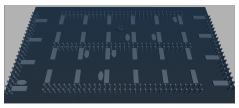

(j) Validation environment two

Figure 6.6: **Example outcomes:** (a) SLAM mission with overhead image (OI) factors, and (b) the same trajectory without OI factors. The gray background of (a) and (b) shows the OI mask; grid boxes are 5m. (c) Results from the SUNY Maritime dataset using the proposed OI system, and (d) results without OI factors using the same trajectory shown in (c). (e) Results from USMMA with OI factors, and (f) results from the same trajectory without OI factors. SSM factors are shown as green lines, loop closures are shown as red lines, and OI factors are shown as blue lines. Poses are shown as boxes, where color changes with time. The planar point cloud map utilizes the same color scheme as the poses the points are derived from. (g),(h) Our simulation training environments, and (i),(j) our validation environments.

# Chapter 7

# DRACo-SLAM: Distributed Robust Acoustic Communication-efficient SLAM for Imaging Sonar Equipped Underwater Robot Teams

## 7.1 Problem Description

In this work we consider a team of robots $N$. Each robot $n \in N$ maintains its own SLAM solution, in its own reference frame $\mathcal{I}_n$. Each robot receives a set of observations $\mathbf{z}_{n,t}$ at a given pose $\mathbf{x}_{n,t}$ across discrete time steps $t$. Each robot pose is defined in the plane (fixed depth) as

$$\mathbf{x}_{n,t} = \begin{pmatrix} x_r \ y_r \ \theta_r \end{pmatrix}^{\top}. \tag{7.1}$$

Each set of observations consists of sonar returns in spherical coordinates with ranges $r \in \mathbb{R}_+$, bearings $\theta \subseteq [-\pi, \pi)$, elevation angles $\phi \subseteq [-\pi, \pi)$, and associated intensity values $\gamma \in \mathbb{R}_+$. These sonar observations can be mapped into Cartesian coordinates by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = r \begin{pmatrix} \cos\phi\cos\theta \\ \cos\phi\sin\theta \\ \sin\phi \end{pmatrix}. \tag{7.2}$$

Each robot moves through the environment according to the dynamics

$$\mathbf{x}_{n,t} = \mathbf{g}(\mathbf{u}_{n,t}, \mathbf{x}_{n,t-1}) + \sigma_{n,t}, \tag{7.3}$$

where $\mathbf{x}_{n,t-1}$ is the previous timestep's pose, $\mathbf{u}_{n,t}$ is the control command and $\sigma_{n,t}$ is process noise. The posterior probability over the time history of poses is defined as

$$p(\mathbf{x}_{n,1:t}, \mathbf{m}_n | \mathbf{z}_{n,1:t}, \mathbf{u}_{n,1:t}), \tag{7.4}$$

with map $\mathbf{m}_n$. While we consider a distributed multi-robot SLAM solution, there is no initial transform between each of the reference frames $\mathcal{I}_n$. This work aims to use each robot's observations to derive inter-robot loop closures and estimate team member trajectories in each robot's reference frame.

## 7.2 Algorithm

This section will provide the technical details of our distributed multi-robot SLAM system. An overview of this system from the perspective of one robot is shown in Fig. 7.1. Each robot logs messages from team members, using a k-d tree to search for potential inter-robot loop closures. We then attempt registration and reject outliers. Inter-robot loop closures are added to the factor graph and broadcast to the rest of the team. The pose graph is optimized, and significant state changes are communicated to the rest of the team.

### 7.2.1 Sonar image processing

At each pose, $\mathbf{x}_{n,t}$, the sonar observations $\mathbf{z}_{n,t}$ consist of a sonar image. The 2D image is populated with acoustic intensity values. However, not all the pixels in the image represent contact with structures in the environment. Our first step is to identify which pixels constitute a sonar contact and which do not. We use constant false alarm rate (CFAR) detection, [76] which is derived from radar processing and has supported our previous work [3].

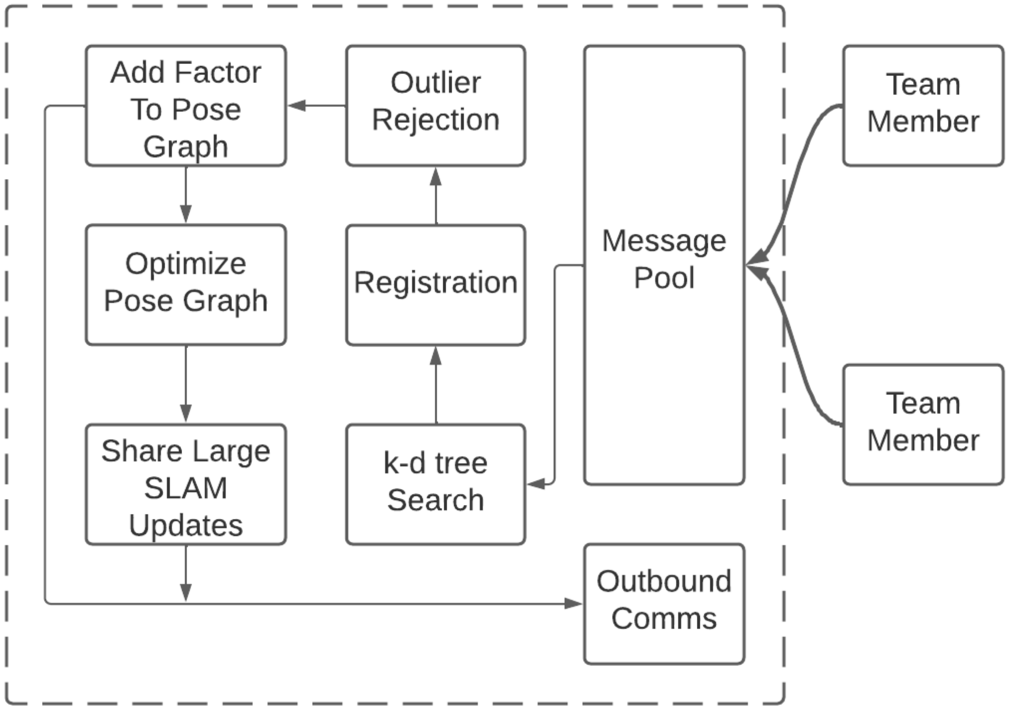


Figure 7.1: **System diagram for a single robot.** Each robot maintains a message pool of received scene descriptors. Based on k-d tree search, registration is attempted and outliers rejected. Inlier inter-robot loop closures are added to the pose graph and sent to team members. Large SLAM corrections are also sent to team members.

Once contacts are identified, the pixels are mapped to meters as an in-plane point cloud. Note that while imaging sonar observes a 3D volume of water, the sensor does not return 3D information, only a 2D projection with $\phi$ as zero. The consequence is that our system is confined to fixed depth, in-plane pose estimation. The point cloud is subject to voxel down-sampling, with each voxel's output being the medoid of the contained points. Example point clouds are shown in Fig. 7.5.

### 7.2.2 Point Cloud Compression

In this work, each point in a point cloud is a pair of 32-bit floating points (x,y), which may overwhelm the data link. For this reason, we consider a simple voxel-based compression algorithm. We take a point cloud, cast it onto a planar voxel grid, and only retain centers of occupied voxels. This compression is performed first by discretizing each point, $p_{x,y}$, to a designated resolution, $\Delta_{compression}$.

$$p_{i,j} \approx \frac{p_{x,y}}{\Delta_{compression}} \tag{7.5}$$

Note that the $p_{i,j}$ are discrete. Each cell in the voxel grid is populated according to Eq. (7.5). If a voxel contains a non-zero number of points, then that voxel's center is recorded for transmission. In our implementation, each voxel center is a pair of 8-bit unsigned integers, offering significant savings over a pair of 32-bit floating points. This compression, though, results in a loss of some geometric information. Our experiments will evaluate compression efficacy by characterizing SLAM error and network utilization in Section V.

### 7.2.3 Single robot SLAM

For multi-robot SLAM, first, each robot must estimate its state. We utilize the vehicles' onboard Doppler velocity log (DVL) and inertial measurement unit (IMU) with the uncompressed point clouds from Section IV-A. We formulate this as a graph-based pose SLAM problem and use the GTSAM [80] implementation of iSAM2 [81]. We use odometry factors $\mathbf{f}^{0}$ from the vehicle dead reckoning system between sequential poses. Next, we add sequential scan matching (SSM) factors. $\mathbf{f}^{\mathrm{SSM}}$ are derived from calling ICP [82] between sequential frames. Lastly, we consider non-sequential scan matching factors (i.e., intra-robot loop closures). $\mathbf{f}^{\mathrm{NSSM}}$ factors are derived by calling ICP between non-sequential frames, outlier factors are rejected using a minimum required point cloud overlap, then PCM [72].

$$\mathbf{f}(\boldsymbol{\Theta}) = \mathbf{f}^0(\boldsymbol{\Theta}_0) \prod_i \mathbf{f}_i^{\mathrm{O}}(\boldsymbol{\Theta}_i) \prod_j \mathbf{f}_j^{\mathrm{SSM}}(\boldsymbol{\Theta}_j) \prod_q \mathbf{f}_q^{\mathrm{NSSM}}(\boldsymbol{\Theta}_q)$$

Note that poses (keyframes) are instantiated if the dead reckoning system indicates a distance or rotation larger than a threshold compared to the previous pose. When each robot passes its state to the team members, it incrementally passes its optimized poses as keyframes are instantiated.

### 7.2.4 Distributed Multi-robot SLAM

Now that each robot has a system to estimate its state, we extend it to include the rest of the team's states in a distributed multi-robot SLAM solution. We use the existing pose graph to integrate inter-robot loop closures relating team members' trajectories. Note that each robot is responsible for its SLAM solution while passing the required information to build a multi-robot state estimate that includes each robot.

Each robot will incrementally share its pose estimates via the datalink as keyframes are added. We add any detected inter-robot factors, $\mathbf{f}^{\mathrm{IR}}$, as in Section V-F. Next, we will add the entire trajectory for each robot in the team if that robot has at least one $\mathbf{f}^{\mathrm{IR}}$. The trajectory will be added as a series of sequential factors between robot poses, denoted $\mathbf{f}^{\mathrm{PR}}$, *partner robot* factors. This completes the factor graph as shown in Fig. 7.2.

### 7.2.5 Inter-robot Loop Closure Search

We formulate the search for inter-robot loop closure candidates as a retrieval problem. Given a candidate and a database, we wish to search the database for possible matches. To do so, we build a scene descriptor using a range-based histogram similar

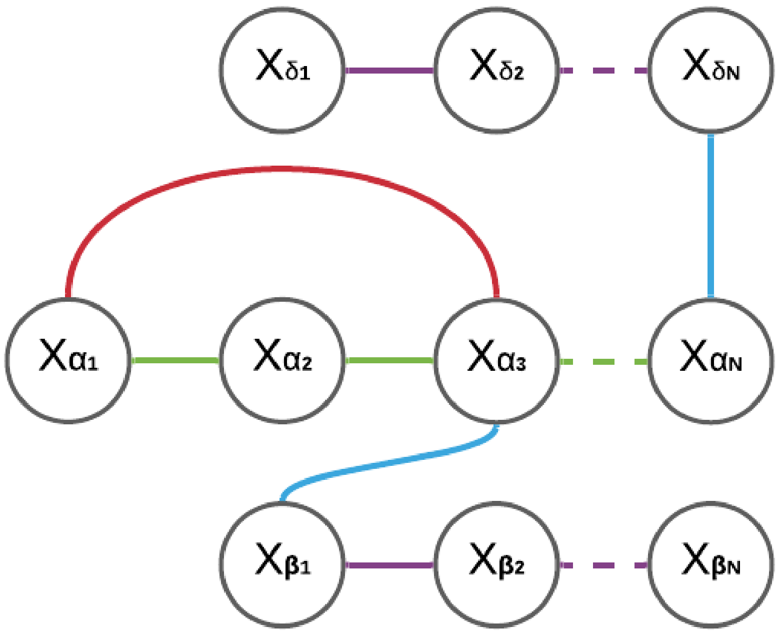


Figure 7.2: **SLAM Factor Graph.** Robot poses $\mathbf{x}$ for three robots, $\alpha$, $\beta$, $\delta$ are considered with several factors: sequential scan matching factors (SSM) in green, non-sequential scan matching factors (NSSM, intra-robot loop closures) shown in red, inter-robot (IR) loop closures in blue and partner robot (PR) factors in purple.

to [94] for each scene. These histograms encode a scene according to the number of measurements at each range bin, making it rotationally invariant. Note that range bins are discrete, the size and range of which are provided a priori. Using the point cloud, each histogram bin is the number of points in its respective range bin.

Each robot encodes and communicates its scene descriptors to the rest of the team incrementally as the keyframes are instantiated. Note that when this message is shared, it also includes the pose estimate of the keyframe in the sender robot's reference frame. Additionally, each robot maintains a k-d tree of its scene descriptors. As a robot receives scene descriptors from team members, it compares them to its own using a nearest neighbor search, with a designated maximum neighbor distance. Once nearest neighbors are identified, registration is attempted.

### 7.2.6 Registration

When performing registration between two scenes, we estimate a 3DOF rigid body transformation

$$T = \begin{pmatrix} x_\vartriangle & y_\vartriangle \end{pmatrix}^\top \tag{7.6}$$

$$R = \begin{pmatrix} \cos\theta_\vartriangle & -\sin\theta_\vartriangle \\ \sin\theta_\vartriangle & \cos\theta_\vartriangle \end{pmatrix}, \tag{7.7}$$

in order to derive inter-robot factors, $\mathbf{f}^{\mathrm{IR}}$. In the registration problem we minimize the squared Euclidean distance between two sets of points $A$ and $B$, that have been associated according to nearest neighbor.

$$E(R,T) = \Sigma ||a_i - Rb_i + T||^2. \tag{7.8}$$

Note we do not have any notion of an initial guess between robot reference frames, and therefore have no notion of initial guess when solving the registration problem. It is for this reason we utilize Go-ICP [95]. Go-ICP finds a globally optimal registration result, showing good performance under partial overlap and poor initial guess conditions. For Go-ICP initial alignment, we apply Euclidean mean subtraction to the point clouds and set $R$ and $T$ as identity and zero, respectively. Once Go-ICP completes, we call standard ICP to refine the transformation using the Go-ICP result as an initial guess.

### 7.2.7 Outlier Rejection

Once registration is complete, the result must be evaluated as legitimate or erroneous. In our implementation, we consider outlier rejection before and after registration. Before a registration call is made, we check that the point cloud has a minimum number of points, to avoid attempting registration with a cloud containing insufficient information to provide a reasonable transformation estimate. Next, we consider the ratio between the point clouds; if the point ratio is too large or small, the scenes are not likely to be of the same content and therefore discarded. Next, we cast sonar points onto a coarse grid to enable rapid scene-to-scene comparison, which we term the *scene image*. When comparing loop closure candidates, the two relevant scene images are compared, and if the sum of absolute differences between them is high, the scenes are considered different and do not warrant registration. Registration is attempted if the point clouds have the minimum number of points, comply with our ratio requirements, and have a small-enough sum of absolute differences between scene images.

Once registration is completed, we evaluate the overlap between the two clouds using the resultant transform estimate. Overlap is computed by evaluating the percentage of points with the nearest neighbor inside 0.5 meters. If the overlap is sufficient, we pass the inter-robot loop closure to PCM [72] for geometric verification. Once a loop closure is validated by PCM, it is added to the pose graph and communicated to the rest of the robots in the team.

### 7.2.8 Communication Strategy

We note that communication between robots is bandwidth limited. For this reason, we formulate our communications strategy to minimize the transmission of large data structures by first exchanging small ones. After the robot has completed its pose graph optimization at each step, it shares the newest scene histogram, with the associated pose. When another member of the multi-robot system receives this message, the histogram is used to query a k-d tree. If any of the nearest neighbors comply with the maximum tree distance, then that point cloud is requested. Upon receipt of this cloud, registration is attempted. This model aims to eliminate the exchange of raw sensor data at each time step. Raw sensor data is only exchanged if scenes are close in feature space, potentially resulting in an inter-robot loop closure.

Note that we incrementally share pose estimates of robot keyframes as keyframes are instantiated. AUV state estimation, however, is prone to high SLAM drift and subsequent drift correction when loop closures are detected. These corrections mean that robots may need to share updated pose estimates with their team; otherwise, they will be operating with deprecated knowledge of other robots in the system. We handle this by setting a change threshold for a robot's state; if any pose changes significantly, that pose is re-sent to the team. The team then uses this new information to implement the $\mathbf{f}^{\mathrm{PR}}$.

| | Entire Trajectory | | | | Poses With Inter-robot Loop Closures | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Euc. Dist. (m) | | Theta (deg.) | | Euclidean Dist. (m) | | Theta (deg.) | | Network Use (bits/second) | |
| Case | MAE | RMSE | MAE | RMSE | MAE | RMSE | MAE | RMSE | Average | Min/Max |
| SM 1 | 2.83 | 2.89 | 5.12 | 5.12 | 2.2 | 2.2 | 5.12 | 5.12 | - | - |
| SM 2 | 2.36 | 2.39 | **3.47** | **3.47** | 2.06 | 2.06 | 3.47 | 3.47 | - | - |
| SM 3 | **1.92** | **1.98** | 3.5 | 3.51 | **1.5** | **1.97** | **3.27** | **3.28** | 337.62 | 217.05 / 468.05 |
| SM 4 | 2.63 | 2.68 | 4.74 | 4.74 | 2.03 | 2.04 | 4.75 | 4.75 | **161.07** | 134.93 / **208.55** |
| SM 5 | 7.25 | 7.58 | 20.27 | 26.76 | 6.84 | 6.88 | 14.47 | 15.83 | 213.65 | **124.26** / 486.54 |
| USMMA 1 | 2.28 | 2.66 | 3.24 | 3.89 | 1.22 | 1.44 | 1.78 | 2.38 | - | - |
| USMMA 2 | 2.09 | 2.41 | 2.21 | 2.63 | 1.13 | 1.39 | 1.38 | 1.78 | - | - |
| USMMA 3 | **1.17** | **1.29** | **1.93** | **2.27** | **0.66** | **0.73** | **1.27** | **1.47** | 3176.61 | 2556.64 / 5579.54 |
| USMMA 4 | 1.44 | 1.62 | 2.62 | 3.07 | 0.82 | 0.97 | 1.71 | 2.09 | **1244.84** | **1050.19 / 1284.24** |
| USMMA 5 | 3.09 | 3.59 | 17.76 | 22.66 | 3.26 | 3.31 | 11.51 | 13.33 | 2126.56 | 2079.58 / 2173.16 |

Table 7.1: **Real world multi-robot SLAM results.** This table summarizes inter-robot error and network utilization for our five test cases. We analyze inter-robot error in two ways; the entire robot trajectory and only poses with inter-robot loop closures. We report the average network utilization and the min/max of the 10 trials for each test case. Note SM denotes SUNY Maritime.

## 7.3 Experiments

### 7.3.1 Hardware Overview

In this work, we utilize our customized BlueROV2-Heavy. This vehicle features depth and attitude control implemented with an onboard PixHawk. The sensor payload includes a Rowe SeaPilot DVL, VectorNav VN-100 MEMS IMU, Bar30 pressure sensor, and a twin sonar arrangement. The vertical sonar is a Blueprint Subsea Oculus M1200d, and the horizontal sonar is an Oculus M750d. We use the horizontal sonar as the SLAM perceptual input; with a max range of 30 meters, the sensor has a 20-degree vertical aperture and a 130-degree horizontal field of view. In order to manage sensor data and our SLAM system, we utilize the robot operating system (ROS). The computer used to manage all instances of SLAM in our multi-robot system is equipped with an Intel i9-9880H 2.30GHz CPU.

In order to perform multi-robot experiments, we record several datasets and concurrently play them back on a single computer. We use $|N|$ instances of our system during playback to create a multi-robot system. Please note that we play back *real data* (sonar, DVL, IMU) and simulate acoustic communication between robots. To simulate communications between robots, we use the existing ROS message passing system to pass the relevant messages between instances of our system.

### 7.3.2 Metrics

In order to consider different parameterizations of our system, we use several metrics. Note that there is no ground truth in our data, nor is there enough publicly available sonar data with ground truth information to perform a study of this type. However, since the goal of our system is to provide awareness of the team relative to a single robot, we formulate the following. Each dataset has a stable single robot SLAM solution; this solution is transformed into the frame of each team member. The location of each team member in a common reference frame is determined using manual alignment. We then compare the single-agent SLAM outcome, transformed into each team member's reference frame, to the estimate each robot has built of its team members using our system. We denote this as inter-robot error, characterized with mean absolute error (MAE) and root mean squared error (RMSE).

Our second metric is network utilization, measured by monitoring the communication channel between robots and measuring total traffic. Note that we exclude network overhead and only report the cost of message contents, as network overhead may be hardware-specific. Network utilization is reported as a time series plot and considering the total usage divided by the mission duration. Note that we perform ten trials for each test case, and results for inter-robot error and network utilization are aggregated across all ten trials.

### 7.3.3 Multi-robot SLAM Ablation Study

To validate our system, we consider two environments: SUNY Maritime College in The Bronx, New York, and the United States Merchant Marine Academy (USMMA) in Kings Point, New York. At SUNY Maritime, we use a system of two robots crossing paths from opposite sides of the environment, with a single opportunity for inter-robot loop closure. At USMMA, we consider a system of three robots traveling in similar directions, with several opportunities for inter-robot loop closure. At SUNY Maritime, the robots start on opposite sides of the environment, while at USMMA all robots start close together.

To test the algorithmic components of our multi-robot SLAM architecture, we present an ablation study in which we progressively add components of our system until the system is complete. Specifically, we vary the contents of our outlier rejection system and specific components of the communication strategy. We note that there is no difference in system parameterization between datasets. For each dataset, we test each of the following cases:

- **Case 1:** min. no. pts., point ratio, and overlap conditions
- **Case 2:** min. no. pts., point ratio, overlap, and scene image conditions
- **Case 3:** min. no. pts., point ratio, overlap, scene image conditions, and PCM
- **Case 4:** min. no. pts., point ratio, overlap, scene image conditions, PCM, and point cloud compression (**The complete proposed system**)
- **Case 5:** min. no. pts., point ratio, overlap, scene image conditions and PCM without re-sending updated pose information

Our results are summarized in Table 7.1, and qualitative examples are shown in Fig. 7.3. We require a minimum number of 75 points, a maximum point ratio of 2.0, a scene image max. of 0.8, minimum overlap of 55% and compression voxel size of 0.25 meters. When computing SLAM metrics, we consider Euclidean distance in meters and yaw (theta) in units of degrees. We break our SLAM metrics into two categories: (1) the entire trajectory and (2) only the poses with inter-robot loop closures. We also report network utilization in units of bits/second for each case. Note that cases 1 and 2 do not have network utilization reported. Their communication configuration does not differ from case 3, nor does their difference in outlier rejection change any form of communication logic.

When considering the results of our multi-robot SLAM ablation study, there are several important takeaways. Firstly there is added value in using a scene image as a method of outlier rejection (case 2) at both USMMA and SUNY Maritime in terms of error. Secondly, the addition of PCM (case 3) yields additional value in terms of inter-robot error. Next, when considering the use of point cloud compression (case 4), there is a slight increase in inter-robot SLAM error in exchange for cutting network utilization in half compared to uncompressed clouds (case 3). Time series plots showing network utilization are given in Fig. 7.4. We note the reduction in network usage when using the compression method, as well as the spike at the beginning of the USMMA mission, which is due to heavy point cloud exchange. Lastly, when removing the sending of significant updates in a robot's state estimate to the rest of the team (case 5), a greatly deteriorated inter-robot SLAM result is observed with some savings in network utilization relative to case 3.

We note that our system runs faster than keyframes are added; an aggregate summary of runtimes from USMMA and SUNY Maritime is shown in Table 7.2. We note that our registration time in Table 7.2 *includes* our outlier rejection system (without PCM, which is captured separately in the table). It is important to note that network utilization with three agents is reasonable when considering the limitations of commercial off-the-shelf hardware options. We want to underscore the significance of this result. This work is the first instance of multi-robot SLAM using real imaging sonar data to utilize indirect encounters, to the best of our knowledge. Not only does our system result in inter-robot loop closures, but these loop closures are also leveraged to build estimates of team members in the frame of each robot in a distributed manner. Further, it does so with an efficient exchange of information, as shown by the network utilization results. Lastly we note the success rate, i.e., how often inter-robot loop closures are successfully detected. For cases 1-4 in both data sets and USMMA case 5 we note a 100% success rate. For SUNY Maritime case 5, we note a 70% success rate, but given this is a data point meant to show the value of an otherwise included feature, we consider it no further.

| Algorithm | Mean (ms) | Standard Deviation (ms) |
|---|---|---|
| PCM | 0.32 | 0.33 |
| Kd-tree Search | 0.88 | 4.49 |
| Registration (w/ rejection) | 258.88 | 265.43 |

Table 7.2: **Runtime.** PCM is used for loop closure verification, k-d tree search is for scene comparison, and registration refers to the Go-ICP based registration method in Section IV-F, including the outlier rejection computations.

| Method | Mean Count | Std. Dev. | Mean KBits |
|---|---|---|---|
| Scene Descriptor | 1 | 1 | .128 |
| Point cloud - float32 | 138.69 | 142.94 | 9.14 |
| Point cloud - compressed | 138.69 | 142.94 | **2.28** |
| KAZE | 48.68 | 56.73 | 116.20 |
| AKAZE | 37.05 | 41.14 | 20.07 |
| ORB | 168.62 | 322.43 | 82.54 |

Table 7.3: **Perception message overhead.** Mean count and std. dev. refer to the average number of points extracted with the given method. Mean Kbits are computed by taking the overhead of a single point and multiplying it by the mean number of points.

### 7.3.4 Analysis of Perception Message Overhead

Some readers may note that using a computer vision paradigm is common when attempting to register a pair of keyframes with no initial guess. This general paradigm extracts salient point features, associates them, and then computes a transform with RANSAC or similar. We note the use of KAZE [19] and AKAZE [20] features on sonar imagery [21, 15] and the use of ORB [18] in RADAR imagery. However, in the multi-robot case, that would require broadcasting feature points *and descriptors*. We provide a summary of some relevant feature extraction tools for sonar imagery in Table 7.3. Here we consider the data-overhead per feature and the number of features per sonar image in our datasets. We use the OpenCV implementations of KAZE, AKAZE, and ORB to perform this comparison.

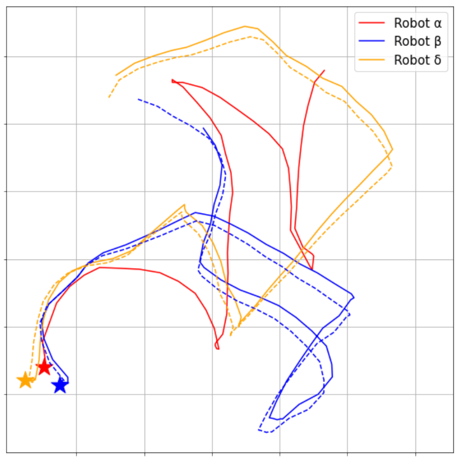
As shown in Table 7.3, our scene descriptor is a data-efficient means of summarizing content. Moreover, even passing simple 32-bit floating-point-based point clouds is significantly less costly than KAZE, AKAZE, and ORB. Lastly, our compression method, while simple, shows a considerable reduction in data requirements, and as we have shown in Section V-C and Table 7.1, only results in a minor increase in inter-robot error.

### 7.3.5 Multi-agent Mapping

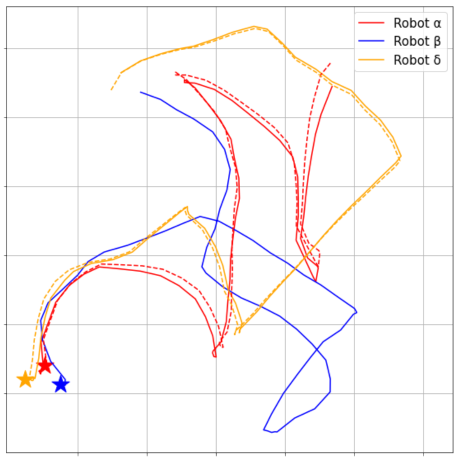
While the purpose of our system is to provide each robot with knowledge of the states of the other agents in the system, it can also be used to derive merged maps. Consider a system of robots tasked with mapping an area in a collaborative modality, a core task for a multi-robot system. This section showcases an example of a map that was merged offline, using the multi-robot SLAM results. We specifically consider the SUNY Maritime dataset, which has two robots. Using the multi-robot state estimate, we can transform the sonar point clouds in the frame of each robot, shown in Fig. 7.5. We note the coverage increase with the use of a multi-robot system and the drift in our SLAM system. This drift is primarily due to the high drift rate of our low-cost MEMS IMU. We also note that the inter-robot drift is comparable to that of a single robot sweeping the entire workspace shown in Fig. 7.5.

## 7.4 Conclusions

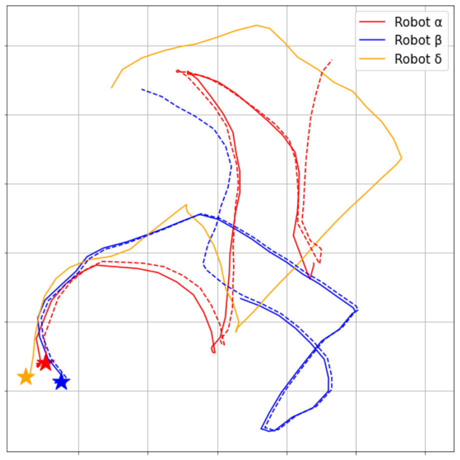
In this work, we have proposed a system to find and integrate inter-robot loop closures in a distributed, graph-based pose SLAM architecture. We have demonstrated the real-time viability of our system, the accuracy of accepted inter-robot constraints, and the efficacy of our communications system. When considering the potential shortcomings of our system, while we take steps to prevent perceptual aliasing from corrupting the SLAM solution, this effect can overwork the communications link. If scene descriptors are similar, we exchange point clouds, meaning that perceptual aliasing can result in the over-exchange of information that our outlier rejection system will cull. Future work will consider geometric verification in scene descriptors to ensure only useful perceptual data is exchanged. However, this is the first example of imaging sonar-based underwater multi-robot SLAM, to the best of our knowledge. There is much potential for future work, including achieving a wide variety of cooperative tasks, and multi-robot active SLAM. Furthermore, our system enables multi-agent autonomous operations in settings where GPS is unavailable or clock synchronization is impractical, which are relevant constraints in many real-world applications.



(a)



(b)



(c)

Figure 7.3: **Example trajectories from USMMA.** This dataset includes three robots, (a) shows robot $\alpha$ (red), (b) shows robot $\beta$ (blue) and (c) shows robot $\delta$ (orange). Solid lines show trajectory estimates with dotted lines showing ground truth used to compute Table 7.1. Note that there is only robot relative ground truth, so there is no true (dotted) line for a robot's own trajectory. We also note that estimated trajectories (solid) may be shorter than true (dotted) due to the varying shutdown times of robots. Stars show starting locations for robots. We avoid using $\gamma$ since it refers to sonar intensity.

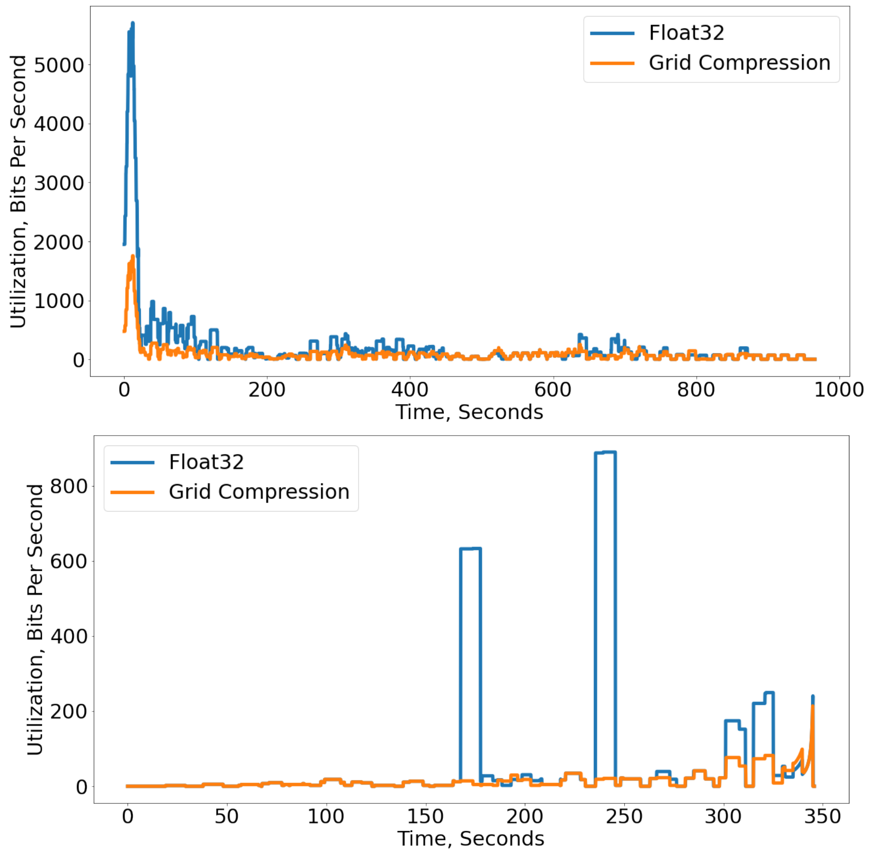


Figure 7.4: **Network utilization.** A comparison of network utilization with and without the voxel compression method described in Section IV-B. The y-axis shows network utilization using a sliding window (window of 100) average. The x-axis shows time. A representative run is shown from the three-robot USMMA datset at top, and from the two-robot SUNY Maritime dataset at bottom.

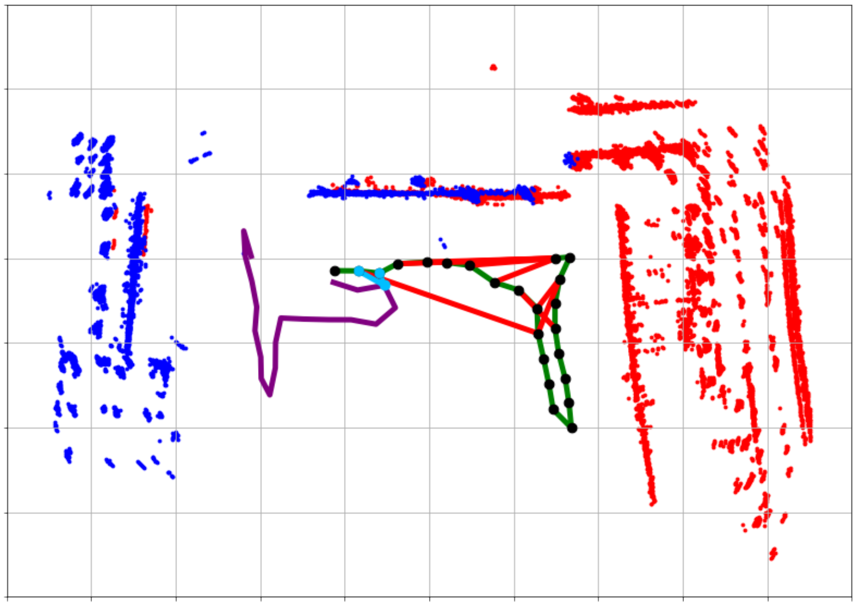


Figure 7.5: **SUNY Maritime results.** An example multi-robot SLAM run from SUNY Maritime, from the perspective of one robot. This robot's poses are shown as black dots connected by several factors: sequential scan matching factors (SSM) in green, non-sequential scan matching factors (NSSM, intra-robot loop closures) shown in red, inter-robot (IR) loop closures in blue and partner robot (PR) factors in purple. This robot's map is shown in red with blue points merged from the other robot in the mission.

# Chapter 8

# Conclusions and Future Work

Thus far our work has consisted of situational awareness for low-cost underwater vehicles, single agent and homogeneous robot teams. We have presented methods for recovering 3D data from sonar, large-scale 3D mapping, using prior information to improve SLAM, and a multi-robot SLAM system. These methods allow low-cost vehicles to operate in complex environments, and even recover 3D data.

Our future work will consider this functionality for multi-agent, inter-domain marine robotics systems. We will consider multi-agent navigation problems, specifically focusing on using groups of robots for ISR, survey, and security missions. We will consider on heterogeneous systems in marine environments, where differently equipped robots can assist each other during a mission.

In our future work, we will focus on extending the robot state estimation problem to multiple robots, even those with heterogeneous sensor payloads. Moreover, we will consider the mapping problem, both 2D and 3D for multiple agents in marine environments. Using DRACo-SLAM, we can consider multi-agent decision-making coupled with the SLAM problem, known as active-SLAM. In this area, we can consider robots cooperatively performing a mission, minimizing a variety of metrics such as time, travel distance, and energy expenditure.

## Bibliography

[1] J. Vincent, S. Vannuffelen, S. Ossia, A. Speck, G. Strunk, A. Croux, A Jarrot, G. Choi, T.P. Osedach, A. Gelman, S. Grall, G. Eudeline, “Supervised Multi-Agent Autonomy for Cost-Effective Subsea Operations,” *Offshore Technology Conference*, 2020.

[2] J. McConnell, J. D. Martin and B. Englot “Fusing Concurrent Orthogonal Wide-aperture Sonar Images for Dense Underwater 3D Reconstruction,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020

[3] J. McConnell and B. Englot “Predictive 3D Sonar Mapping of Underwater Environments via Object-specific Bayesian Inference,” *Proceedings of the IEEE International Conference on Robotics and Automation*, 2021

[4] J. McConnell, F. Chen and B. Englot “Overhead Image Factors for Underwater Sonar-based SLAM,” *IEEE Robotics and Automation Letters*, 2022

[5] J. McConnell, Y. Huang, P. Szenher, I. Collado-Gonzalez and B. Englot, ”DRACo-SLAM: Distributed Robust Acoustic Communication-efficient SLAM for Imaging Sonar Equipped Underwater Robot Teams,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022

[6] J. McConnell, I. Ivana Collado-Gonzalez, P. Szenher and B. Englot, ”Sonar based 3D mapping via submapping,” *In preperation for IEEE JOE*

[7] M.D. Aykin and S. Negahdaripour, "On Feature Matching and Image Registration for Two-dimensional Forward-scan Sonar Imaging," *Journal of Field Robotics*, vol. 30(4), pp. 602-623, 2013.

[8] M.D. Aykin and S. Negahdaripour, "Forward-look 2-D sonar image formation and 3-D reconstruction," *Proceedings of the IEEE/MTS OCEANS Conference*, 2013.

[9] E. Westman and M. Kaess, "Wide Aperture Imaging Sonar Reconstruction using Generative Models," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 8067-8074, 2019.

[10] R. DeBortoli, F. Li, and G. Hollinger, "ElevateNet: A Convolutional Neural Network for Estimating the Mission Dimension in 2D Underwater Sonar Images," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 8040-8047, 2019.

[11] M.D. Aykin and S. Negahdaripour, "Three-Dimensional Target Reconstruction From Multiple 2-D Forward-Scan Sonar Views by Space Carving," *IEEE Journal of Oceanic Engineering*, vol. 42(3), pp. 574-589, 2017.

[12] T. Guerneve, K. Subr, and Y. Petillot, "Three-dimensional reconstruction of underwater objects using wide-aperture imaging SONAR," *Journal of Field Robotics*, vol. 35(6), pp. 890-905, 2018.

[13] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24(6), pp. 1365–1378, 2008.

[14] T.A. Huang and M. Kaess, "Incremental data association for acoustic structure from motion," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1334-1341, 2016.

[15] J. Wang, T. Shan, and B. Englot, "Underwater Terrain Reconstruction from Forward-Looking Sonar Imagery," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3471-3477, 2019.

[16] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[17] H. Bay, A. Ess, T. Tuytelaars, L. Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110(3), pp. 346-359, 2007.

[18] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564-2571, 2011.

[19] P. Alcantarilla, A.Bartoli and A. Davison, "KAZE Features," *Proceedings of the European Conference on Computer Vision*, pp. 214-227, 2012.

[20] P. Alcantarilla, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," *Proceedings of the British Machine Vision Conference*, 2013.

[21] E. Westman, A. Hinduja, and M. Kaess, "Feature-Based SLAM for Imaging Sonar with Under-Constrained Landmarks," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3629-3636, 2018.

[22] H. Assalih, Y. Petillot, and J. Bell, "Acoustic Stereo Imaging (ASI) System," *Proceedings of the IEEE/MTS OCEANS Conference*, 2009.

[23] S. Negahdaripour, "Application of Forward-Scan Sonar Stereo for 3-D Scene Reconstruction," *IEEE Journal of Oceanic Engineering*, vol. 45(2), pp. 547-562, 2020.

[24] A. Mallios, P. Ridao, D. Ribas, M. Carreras, and R. Camilli, "Toward autonomous exploration in confined underwater environments," *Journal of Field Robotics*, vol. 33(7), pp. 994-1012, 2016.

[25] A. Mallios, E. Vidal, R. Campos, and M. Carreras, "Underwater caves sonar data set," *The International Journal of Robotics Research*, vol. 36(12), pp. 1247-1251, 2017.

[26] H.W. Yu and B.H. Lee, "A Variational Feature Encoding Method of 3D Object for Probabilistic Semantic SLAM," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3605-3612, 2018.

[27] H.W. Yu, J.Y. Moon and B.H. Lee, "A Variational Observation Model of 3D Object for Probabilistic Semantic SLAM," *Proceedings of the International Conference on Robotics and Automation*, pp. 5866-5872, 2019.

[28] B. Yang, S. Rosa, A. Markham, N. Trigoni and H. Wen, "Dense 3D Object Reconstruction from a Single Depth View," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 12, pp. 2820-2834, 2019.

[29] R.F. Salas-Moreno, R.A. Newcombe, H. Strasdat, P.H.J. Kelly, and A.J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1352-1359, 2013.

[30] T. Guerneve, K. Subr and Y. Petillot, “Underwater 3D structures as semantic landmarks in SONAR mapping,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 614-619, 2017.

[31] J. Wang and B. Englot, “Fast, Accurate Gaussian Process Occupancy Maps via Test-data Octrees and Nested Bayesian Fusion,” *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1003-1010, 2016.

[32] K. Doherty, J. Wang, and B. Englot, “Learning-aided 3-D Occupancy Mapping with Bayesian Generalized Kernel Inference,” *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 953-966, 2019.

[33] C. O’Meadhra, W. Tabib, and N. Michael, “Variable Resolution Occupancy Mapping Using Gaussian Mixture Models,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2015-2022, 2019.

[34] L. Gan, R. Zhang, J.W. Grizzle, R.M. Eustice, and M. Ghaffari, “Bayesian Spatial Kernel Smoothing for Scalable Dense Semantic Mapping,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 790-797, 2020.

[35] J. Li, M. Kaess, R. M. Eustice and M. Johnson-Roberson, “Pose-Graph SLAM Using Forward-Looking Sonar,” *IEEE Robotics and Automation Letters*, vol. 3(3), pp. 2330-2337, 2018.

[36] M. Hammond and S. M. Rock, “A SLAM-based approach for underwater mapping using AUVs with poor inertial information,” *Proceedings of the IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*, 2014.

[37] H. Johannsson, M. Kaess, B. Englot, F. Hover and J. Leonard, “Imaging sonar-aided navigation for autonomous underwater harbor surveillance,” *Proceedings*

*of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4396-4403, 2010.

[38] P. V. Teixeira, D. Fourie, M. Kaess and J. J. Leonard, "Dense, Sonar-based Reconstruction of Underwater Scenes," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8060-8066, 2019.

[39] J. Wang, S. Bai and B. Englot, "Underwater localization and 3D mapping of submerged structures with a single-beam scanning sonar," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4898-4905, 2017.

[40] K. Y. K. Leung, C. M. Clark and J. P. Huissoon, "Localization in urban environments by matching ground level video images with an aerial image," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 551-556, 2008.

[41] A. Viswanathan, B. R. Pires and D. Huber, "Vision based robot localization by ground to satellite matching in GPS-denied situations," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 192-198, 2014.

[42] S. Workman, R. Souvenir and N. Jacobs, "Wide-Area Image Geolocalization with Aerial Reference Imagery," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 3961-3969, 2015.

[43] D. Kim and M. R. Walter, "Satellite image-based localization via learned embeddings," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2073-2080, 2017.

[44] A. Shetty and G. X. Gao, “UAV Pose Estimation using Cross-view Geolocalization with Satellite Imagery,” *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 1827-1833, 2019.

[45] T. Y. Tang, D. De Martini, D. Barnes and P. Newman, “RSL-Net: Localising in Satellite Images From a Radar on the Ground,” *IEEE Robotics and Automation Letters*, vol. 5(2), pp. 1087-1094, 2020.

[46] M. Machado Dos Santos, G. G. De Giacomo, P. L. J. Drews and S. S. C. Botelho, “Matching Color Aerial Images and Underwater Sonar Images Using Deep Learning for Underwater Localization,” *IEEE Robotics and Automation Letters*, vol. 5(4), pp. 6365-6370, 2020.

[47] G.G. De Giacomo, M.M. dos Santos, P.L.J. Drews and S.S.C. Botelho, “Guided Sonar-to-Satellite Translation,” *Journal of Intelligent and Robotic Systems*, vol. 101, article 46, 2021.

[48] L. Paull, G. Huang, M. Seto and J.J. Leonard, “Communication-constrained multi-AUV cooperative SLAM,” *Proc. IEEE Int. Conf. Robotics Automation*, 509-516, 2015.

[49] E.M. Fischell, N.R. Rypkema and H. Schmidt, “Relative Autonomy and Navigation for Command and Control of Low-Cost Autonomous Underwater Vehicles,” *IEEE Robotics Automation Lett.*, 4(2): 1800-1806, 2019.

[50] Y. Yao, D. Xu and W. Yan, “Cooperative localization with communication delays for MAUVs,” *Proc. IEEE Int. Conf. Intelligent Computing and Intelligent Syst.*, 244-249, 2009.

[51] Y. Li, Y. Wang, W. Yu and X. Guan, "Multiple Autonomous Underwater Vehicle Cooperative Localization in Anchor-Free Environments," *IEEE J. Oceanic Engineering*, 44(4): 895-911, 2019.

[52] T. Halsted and M. Schwager, "Distributed multi-robot localization from acoustic pulses using Euclidean distance geometry," *Proc. Int. Symp. Multi-Robot and Multi-Agent Systems*, 104-111, 2017.

[53] L. Paull, M. Seto and J.J. Leonard, "Decentralized cooperative trajectory estimation for autonomous underwater vehicles," *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Syst.*, 184-191, 2014.

[54] J.G. Mangelson, R. Vasudevan and R.M. Eustice, "Communication Constrained Trajectory Alignment For Multi-Agent Inspection via Linear Programming," *Proc. IEEE/MTS OCEANS Conf.*, 2018.

[55] A. Elibol, J. Kim, N. Gracias and R. Garcia, "Efficient image mosaicing for multi-robot visual underwater mapping," *Pattern Recognition Lett.* 46: 20–26, 2014.

[56] M. Pfingsthorn, A. Birk and H. Bülow, "An efficient strategy for data exchange in multi-robot mapping under underwater communication constraints," *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Syst.*, 4886-4893, 2010.

[57] F. Bonin-Font and A. Burguera, "Towards Multi-Robot Visual Graph-SLAM for Autonomous Marine Vehicles," *J. Marine Science and Engineering*, 8(6): 437, 2020.

[58] M. Xanthidis et al., "Towards Multi-Robot Shipwreck Mapping," *IEEE Int. Conf. Robotics Automation (ICRA) Workshop on Underwater Active Perception*, 2021.

[59] R. Campos, N. Gracias and P. Ridao, "Underwater Multi-Vehicle Trajectory Alignment and Mapping Using Acoustic and Optical Constraints," *Sensors*, 16(3): 387, 2016.

[60] P. Ozog, N. Carlevaris-Bianco, A. Kim and R.M. Eustice, "Long-term Mapping Techniques for Ship Hull Inspection and Surveillance using an Autonomous Underwater Vehicle," *J. Field Robotics*, 33(3): 265-289, 2016.

[61] H. Do, S. Hong and J. Kim, "Robust Loop Closure Method for Multi-Robot Map Fusion by Integration of Consistency and Data Similarity," *IEEE Robotics Automation Lett.*, 5(4): 5701-5708, 2020.

[62] E. Galceran, S. Nagappa, M. Carreras, P. Ridao and A. Palomer, "Uncertainty-driven survey path planning for bathymetric mapping," *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Syst.*, 6006-6012, 2013.

[63] S.B. Williams O. Pizarro, B. Foley, "Return to Antikythera: Multi-session SLAM Based AUV Mapping of a First Century B.C. Wreck Site", *Proc. Int. Conf. Field and Service Robotics*, 2015.

[64] M. Bryson, M. Johnson-Roberson, O. Pizarro and S. Williams, "Automated registration for multi-year robotic surveys of marine benthic habitats," *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Syst.*, 3344-3349, 2013.

[65] G. Kim and A. Kim, "Scan Context: Egocentric spatial descriptor for place recognition within 3D point cloud map," *Proc. IEEE/RSJ Int. Conf. Intelligent Robots Syst.*, 4802-4809, 2018.

[66] H. Wang, C. Wang and L. Xie, "Intensity Scan Context: Coding Intensity and Geometry Relations for Loop Closure Detection," *Proc. IEEE Int. Conf. Robotics Automation*, 2095-2101, 2020.

[67] G. Kim, S. Choi and A. Kim, "Scan Context++: Structural Place Recognition Robust to Rotation and Lateral Variations in Urban Environments," *IEEE Trans. Robotics*, DOI: 10.1109/TRO.2021.3116424, 2021.

[68] M.M. Santos, G.B. Zaffari, P. Ribeiro, P.L. J. Drews-Jr. and S.S.C. Botelho, "Underwater place recognition using forward-looking sonar images. A topological approach," *J. Field Robotics*, 36(2): 355-369, 2018.

[69] P.O.C.S. Ribeiro et al., "Underwater Place Recognition in Unknown Environments with Triplet Based Acoustic Image Retrieval," *Proc. IEEE Int. Conf. Machine Learning and Applications*, 524-529, 2018.

[70] M. Larsson, N. Bore and J. Folkesson, "Latent Space Metric Learning For Sidescan Sonar Place Recognition," *Proc. IEEE/OES Autonomous Underwater Vehicles Symp.*, 2020.

[71] J. Wang, F. Chen, Y.Huang, J. McConnell, T. Shan, B. Englot, "Virtual Maps for Autonomous Exploration of Cluttered Underwater Environments," *International Conference on Robotics and Automation (ICRA) Marine Robotics Workshop: Active Perception*

[72] J.G. Mangelson, D. Dominic, R. M. Eustice and R. Vasudevan, "Pairwise Consistent Measurement Set Maximization for Robust Multi-Robot Map Merging," *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2916-2923, 2018.

[73] Woods Hole Oceanographic Institute, "WHOI Micromodem," https://acomms.whoi.edu/micro-modem/

[74] EvoLogics, "HS Devices," https://evologics.de/acoustic-modem/hs

[75] E. Demirors, G. Sklivanitis, G.E. Santagati, T. Melodia and S.N. Batalama, "A High-Rate Software-Defined Underwater Acoustic Modem With Real-Time Adaptation Capabilities," *IEEE Access*, 6: 18602-18615, 2018.

[76] M. Richards, *Fundamentals of Radar Signal Processing*, McGraw Hill, 2005.

[77] K. El-Darymli, P. McGuire, D. Power and C. Moloney, "Target detection in synthetic aperture radar imagery: a state-of-the-art survey," *Journal of Applied Remote Sensing*, vol. 7(1), pp. 6014-6058, 2013.

[78] G. Acosta and S. Villar, "Accumulated CA–CFAR Process in 2-D for Online Object Detection From Sidescan Sonar Data," *IEEE Journal of Oceanic Engineering*, vol. 40(3), pp. 558-569, 2015.

[79] M. Ester, H. Kriegel, J. Sander, X, Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226-231, 1996.

[80] F. Dellaert, "Factor Graphs and GTSAM: A Hands-on Introduction," Georgia Institute of Technology, Technical Report No. GT-RIM-CPR-2012-002, 2012.

[81] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, “iSAM2: Incremental Smoothing and Mapping using the Bayes Tree,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216-235, 2012.

[82] P.J. Besl and N.D. McKay, “A Method for Registration of 3-D Shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*. vol. 14, no. 2, pp. 239–256, 1992.

[83] M. Fischler and R. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981.

[84] X. Hu and P. Mordohai, ”Evaluation of Stereo Confidence Indoors and Outdoors,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1466-1473, 2010.

[85] A. Loquercio, M. Segu and D. Scaramuzza, “A General Framework for Uncertainty Estimation in Deep Learning,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3153-3160, 2020.

[86] N. Koenig and A. Howard, “Design and Use Paradigms for Gazebo, an Open-source Multi-robot Simulator,” *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2149-2154, 2004.

[87] 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,” *Proceedings of the IEEE/MTS OCEANS Conference*, 2016.

[88] R. Cerqueira, T. Trocoli, G. Neves, S. Joyeux, J. Albiez, and L. Oliveira, "A Novel GPU-based Sonar Simulator for Real-time Applications," *Computers and Graphics*, vol. 68, pp. 66-76, 2017.

[89] O. Ronneberger and P.Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[90] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, R. Raskar, "DeepGlobe 2018: A Challenge to Parse the Earth Through Satellite Images," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 172-181, 2018.

[91] M. Abadi, A. Agarwal, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," arXiv:1603.04467, 2016.

[92] D. P. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization," *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[93] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: An Open-source Robot Operating System" *IEEE ICRA Workshop on Open Source Software*, 2009.

[94] S. Thrun, W. Burgard, D. Fox , "Probablistic Robotics" *MIT Press*, 2005.

[95] J. Yang, H. Li and Y. Jia, "Go-ICP: Solving 3D Registration Efficiently and Globally Optimally," *Proc. IEEE Int. Conf. Computer Vision*, 1457-1464, 2013.

## Vita

**John McConnell**

### Education

| | |
|---|---|
| Ph.D. in Mechanical Engineering | January 2018 - May 2023 |
| Stevens Institute of Technology, Hoboken, NJ | |
| M.E. in Mechanical Engineering | August 2018 - December 2020 |
| Stevens Institute of Technology, Hoboken, NJ | |
| B.E. in Mechanical Engineering | September 2011 - May 2015 |
| SUNY Maritime College, Bronx, NY | |

### Work Experience

| | |
|---|---|
| Argo AI, Pittsburgh, PA | |
| Software Engineer Internship Localization | June 2021 - August 2021 |
| Duro Underwater Autonomous Systems, Bronx, NY | |
| Software Engineer | July 2016 - June 2018 |
| SeaRiver Maritime, ExxonMobil, Alaska and Pacific Northwest | |
| Marine Engineering Officer | July 2016 - June 2018 |
| American Bureau of Shipping, Houston, TX | |
| Rotational Engineer | July 2015 - July 2016 |

## Publications

**John McConnell**, John D. Martin and Brendan Englot, “ Fusing Concurrent Orthogonal Wide-aperture Sonar Images for Dense Underwater 3D Reconstruction,” *International Conference on Intelligent Robots and Systems (IROS),* 2020.

**John McConnell** and Brendan Englot, “ Predictive 3D Sonar Mapping of Underwater Environments via Object-specific Bayesian Inference ,” *International Conference on Robotics and Automation (ICRA),* 2021.

**John McConnell** Fanfei Chen, and Brendan Englot, “Overhead Image Factors for Underwater Sonar-based SLAM ,” *Robotics and Automation Letters (RA-L),* 2022.

**John McConnell**, Yewei Huang, Paul Szenher, Ivana Collado-Gonzalez and Brendan Englot, “DRACo-SLAM: Distributed Robust Acoustic Communication-efficient SLAM for Imaging Sonar Equipped Underwater Robot Teams,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022

**John McConnell**, Ivana Collado-Gonzalez and Brendan Englot, “Perception for Underwater Robots,” *Current Robotics Reports*, 2022

**John McConnell**, Ivana Collado-Gonzalez, Paul Szenher and Brendan Englot, “Sonar based 3D mapping via submappings,” *In preperation for IEEE Journal of Oceanic Engineering*, 2022

Jinkun Wang, Fanfei Chen, Yewei Huang, **John McConnell**, Tixaio Shan and Brendan Englot, “Virtual maps for autonomous exploration of cluttered underwater environments,” *IEEE Journal of Oceanic Engineering*, 2022