Virtual Occupancy Grid Map with Applications to Autonomous 3D Reconstruction in Underwater Unstructured Scenes

Bing-Jui Ho

CMU-RI-TR-18-49

July 2018

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Michael Kaess (Chair)
Maxim Likhachev
Sankalp Arora

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Abstract

Reconstruction of marine structures such as pilings underneath piers presents a plethora of interesting challenges. It is one of those tasks better suited to a robot due to harsh underwater environments. Underwater reconstruction typically involves human operators remotely controlling the robot to predetermined way-points based on some prior knowledge of the location and model of the object of interest. However, it is impractical and dangerous to manually control the robot to perform the reconstruction task in an unstructured scene where prior knowledge of the locations and shapes of objects of interest is not available.

Based on the state-of-the-art mapping and planning methods, this thesis presents an approach that enables the robot to perform reconstruction task in an underwater unstructured scene autonomously without any prior knowledge except the bounding box within which the robot should operate. In particular, the key challenge lies in working with a world representation that is compatible with both mapping and planning algorithms. We address this challenge with the proposed *virtual* occupancy grid map or VOG-Map. VOG-Map is represented by a collection of local occupancy grid maps whose respective poses are optimized to account for the drift or accumulated noise.

Based on the VOG-Map, a path planning algorithm is able to plan safer way-points for collision-free paths as well as more informative way-points for accurate reconstruction than is that based on a global occupancy grid map. By adding additional constraints on the way-points and modifying how their respective information gains are computed, we show how our mapping algorithm could work well with the way-points returned by the planner based on VOG-Map.

The quality of both VOG-Map and scene reconstruction depends on the mapping algorithm. We employ smoothing-based pose graph simultaneous localization and mapping (SLAM) algorithm which is capable of correcting for drift upon loop closures when the algorithm determines that the robot has come back to a previously visited area. However, in scene that lacks geometric structures, the process of determining loop closures via methods such as iterative closest point (ICP) is prone to error. We incorporate the approach of determining degeneracy in the scene into our ICP method and add a loop closure constraint to the SLAM optimization problem, constraining only well-conditioned directions based on principal component analysis.

We demonstrate the use of VOG-Map for implementing an underwater system in which the robot actively plans paths to generate accurate 3D scene reconstructions. We evaluate our system qualitatively and quantitatively on simulated as well as real-world experiments.

Contents

Chapter 1

Introduction

1.1 Motivation

High fidelity 3D reconstruction of marine structures is of major importance for security and commercial reasons. Accurate reconstructions of underwater structures such as ship hull and pilings underneath piers allow authorities to take preventive actions to minimize security risk and economic loss. Delegating this reconstruction task to a robot can be cost-effective without jeopardizing the life of human divers. While in cases such as ship reconstruction, controlling the robot remotely suffices, reconstruction of pilings underneath piers requires the robot to autonomously explore and map in real-time since it is impractical or even impossible to manually control the robot in-between the pilings especially when prior knowledge of the pilings is not available.

Currently, Hover, et al. in [Hover et al., 2012] and Teixeira, et al. in [Teixeira et al., 2016] demonstrate accurate ship reconstruction using a robot. These approaches, however, require the robot to follow predefined trajectory [Hover et al., 2012] or be manually controlled [Teixeira et al., 2016], relying on various assumptions and prior knowledge with regard to the ship hull and the environment. In particular, assumptions about the geometric characteristics of the ship are justified given the ship's structural drawing. Such assumptions and prior knowledge are not available when it comes to reconstruction of pilings underneath piers which is an example of reconstruction in unstructured scene where there might be multiple separate objects of interest. Enabling the robot to perform such reconstruction task autonomously is hence necessary.

The task of underwater reconstruction in unstructured scene has gained increased attention as the object of interest shifts from ship to pilings. Fig. 1.1 shows an example of pilings underneath piers in a harbor environment. Fig. 1.2 shows the *Hovering Autonomous Underwater Vehicle* (HAUV) which is the marine robot developed and used for ship reconstruction tasks in [Hover et al., 2012] and [Teixeira et al., 2016]. The work presented in this thesis is also based on the HAUV.

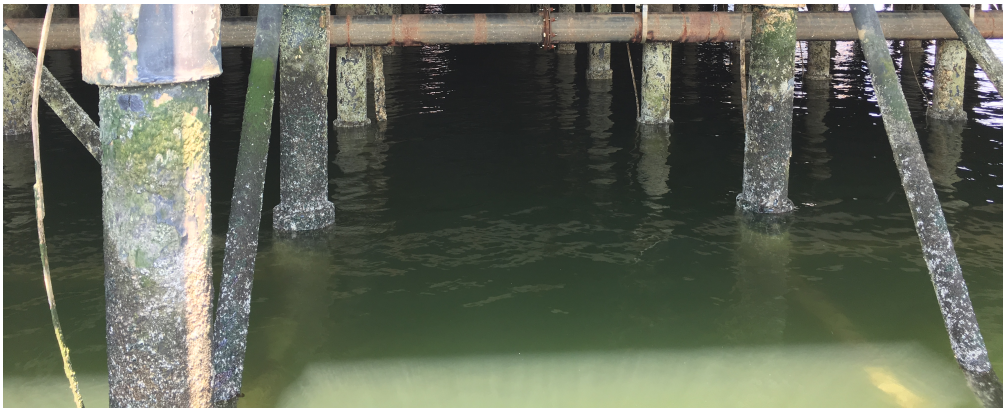

Figure 1.1: Pilings underneath a pier. The spacing between each pair of piles can differ and piles might be slanted.

1.2 Scope and Approach

In this thesis, we develop an approach that enables the HAUV to perform reconstruction of pilings underneath piers autonomously without any assumptions and prior knowledge except the bounded region in which the robot should operate. While the method developed is meant for pilings reconstruction using the HAUV, it could be extended to work with other robotic platforms for unstructured scene reconstruction tasks for which autonomy is required.

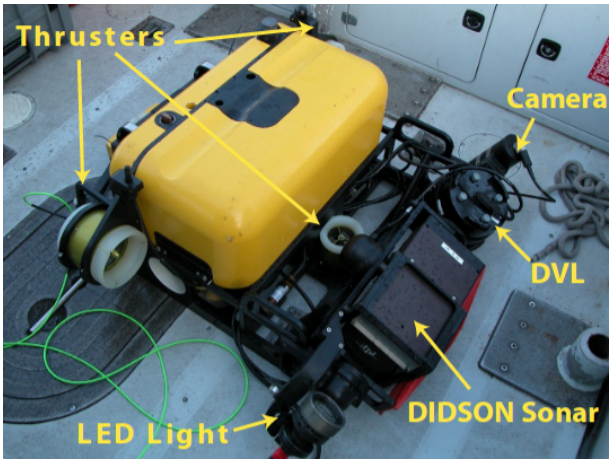

Figure 1.2: Hovering Autonomous Underwater Vehicle

Underwater reconstruction is especially challenging largely because of the uncertain state estimates in underwater environment due to underwater current. In addition, a complicating factor is the poor visibility in harbor environments, requiring the use of multi-beam sonar sensors. Teixeira, et al. in [Teixeira et al., 2016] address these challenges by formulating a smoothing-base pose graph SLAM [Kaess et al., 2007] problem where each pose node is associated with a submap, assembled from consecutive sonar scans. These submaps when registered provide additional information that helps the SLAM optimization problem arrive at better state estimates. Teixeira's sonar-based submap approach has been used in this thesis as the basis of our mapping algorithm. Fig. 1.3 shows the reconstructed running gear of a ship taken from [Teixeira et al., 2016].

Underwater reconstruction task usually involves a human operator who either specifies a path for the robot to follow or remotely controls the robot by sending motion commands on the fly. As are the cases in both [Hover et al., 2012] and Teixeira et al. [2016], predefined trajectory resembling

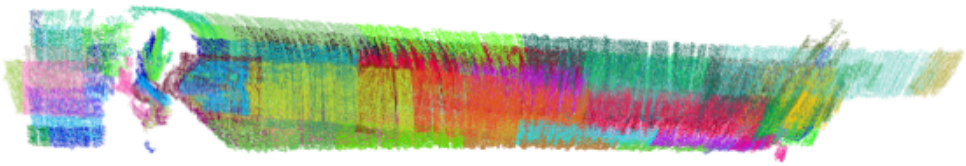

Figure 1.3: Reconstruction of *SS Curtiss*' running gear using sonar-based submap approach Teixeira et al. [2016]

a vertical lawn-mower pattern is followed by the robot to ensure efficient coverage and sufficient overlaps for loop closure registration (or equivalently, *pairwise submap registration* in [Teixeira et al., 2016]). However, in unstructured scene such as pilings underneath piers as shown by Fig. 1.1, it is impractical or even impossible to determine in advance all the way-points that the robot should follow. To ensure full coverage, for each way-point, in addition to position in 3D space, the robot's heading now has to be considered in unstructured scene. Yet, knowledge of free-space, the place in space that is safe for robot to move freely, is required to specify a coverage trajectory beforehand. For ship reconstruction task, assuming unknown-space as free-space is usually justified as the ship is docked in rather open space. However, in unstructured scene, it is dangerous to make such free-space assumption since the area of operation is crowded with multiple unknown objects of interest separate in space.

Bircher, et al. in [Bircher et al., 2016] address the challenge of efficient unknown-space exploration by using sampling-based planner based on rapidly exploring random tree or RRT [LaValle, 1998] and OctoMap, an efficient octree-based occupancy grid space representation [Hornung et al., 2013]. Bircher's sampling-based exploration approach has been used in this thesis as the basis of our planning algorithm. Fig 1.4 shows the result of exploring and mapping a simulated bridge taken from Bircher et al., 2016.

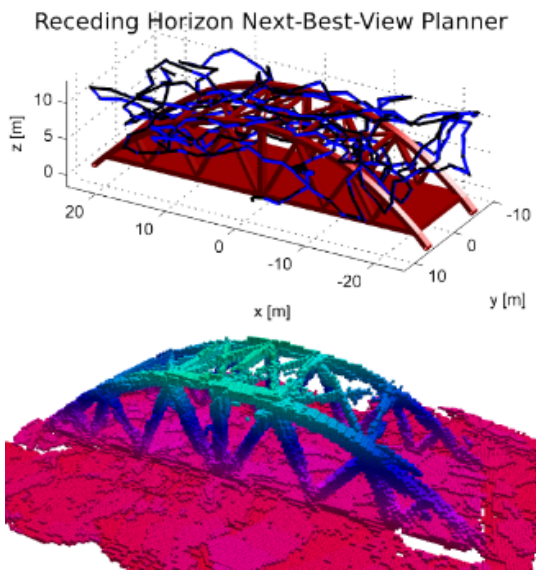

Figure 1.4: Simulated bridge reconstruction using sampling-based planner and Octomap Bircher et al. [2016]

Consider now the combined problem of unknown space exploration and reconstruction in an unstructured underwater scene. Naively adopting state-of-the-art planning approaches such as *"next-best-view" planner* or NBVP in Bircher et al. [2016] which assume the current state estimate coincides with the true state estimate would produce poor reconstructions due to drift or accumulated noise. This is where the state-of-the-art mapping approaches based on SLAM as presented in [Teixeira et al., 2016] could come to rescue, but it is not straightforward how to bring the two techniques together. To be specific, the problem with state-of-the-art planning approaches is that their space representations cannot be corrected for drift. On the other hand, space represen-

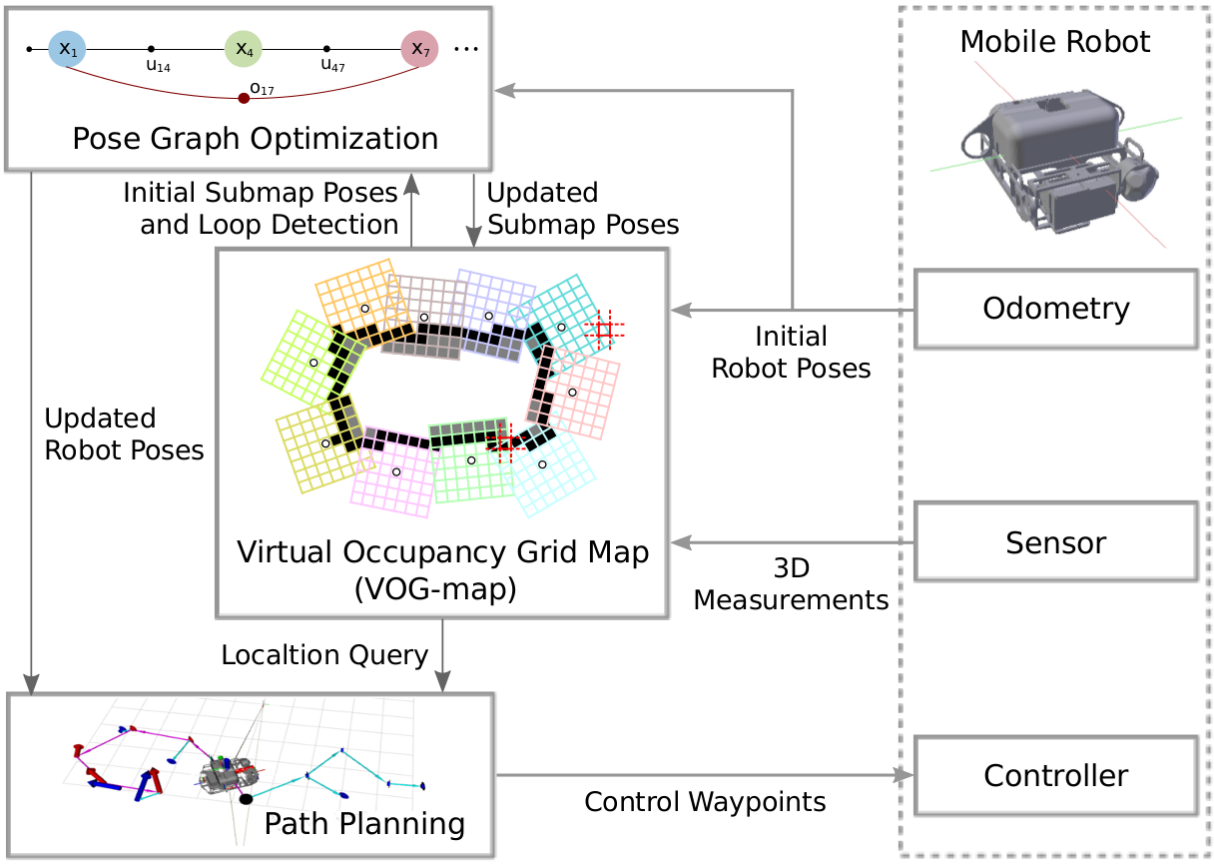

Figure 1.5: Overall system employing a *virtual* occupancy grid map or VOG-Map for mapping and planning in order to enable the robot to map autonomously an unknown underwater unstructured scene

tations maintained by state-of-the-art mapping algorithms do not provide free-space information which is required by the planner. The key challenge lies in working with a space representation that is compatible with both mapping and planning algorithms. In this thesis, we address this challenge by proposing *virtual* occupancy grid map or VOG-Map. VOG-Map is a collection of local occupancy grid maps, each anchored at the pose nodes of the smoothing-based pose graph SLAM framework. VOG-Map can be corrected for drift when pose estimates of the poses nodes get updated upon loop closures. Yet, when considered together, this collection of local occupancy grid maps has the same expressive power as that of a single global occupancy grid map that can distinguish between free-space, occupied-space, and unknown-space.

The use of VOG-Map for implementing an underwater system in which the robot autonomously plans paths to generate accurate 3D scene reconstructions is illustrated in Fig. 1.5. The key to understand this overall system chart is to consider the mapping and planning algorithms each running on its own thread. Mapping algorithm's main interaction with VOG-Map is to update VOG-Map 3.4.1 upon loop closures while planning algorithm queries VOG-Map 3.4.2 to plan paths for the robot to follow.

1.3 Contributions and Organization

In this thesis, we present an approach that enables the robot to autonomously explore and map in an underwater unstructured scene where robot's uncertain state estimates must be considered. This thesis makes the following main contributions:

- The state-of-the-art mapping approach requires the robot to either be manually controlled or follow a predetermined trajectory while the state-of-the-art planning approach assumes accurate state estimates. We propose an approach to unify state-of-the-art smoothing-based pose graph SLAM mapping approach and state-of-the-art sampling-based planning approach by constructing a globally deformable *virtual* occupancy grid map (VOG-Map). This space representation allows pose graph SLAM systems to correct globally accumulated drift via loop closures while maintaining free space information for the purpose of path planning.
- The quality of VOG-Map depends on both the *pairwise submap registration* and the paths returned by the planner. To avoid bad *pairwise submap registration* in degenerate scene, we customize our ICP method to be degeneracy-aware, and add a custom pairwise submap registration constraint on only well-conditioned direction based on principal component analysis. In addition, we add constraints to sampled way-points and modify how their respective information gains are computed. These result in an overall better trajectory that works well specifically with the sonar-based submap SLAM mapping approach using HAUV.
- A full-fledged autonomous system based on VOG-Map is implemented for reconstruction tasks in underwater unstructured scenes. The system is evaluated in both simulation and real-world experiments in terms of reconstruction quality.

The thesis is structured as follows. Chapter 2 discusses background and related work. Chapter 3 presents details on the *virtual* occupnacy grid map or VOG-Map. Chapter 4 demonstrates the use of VOG-Map in an underwater unstructured scene reconstruction task. Finally, Chapter 5 summarizes the thesis' contributions and presents thoughts for future work.

Chapter 2

Background and Related Work

We will begin with the discussion of submaps used in SLAM literature, with a focus on related works that use submaps also for exploration and navigation tasks that involve planning since our proposed VOG-Map can be considered as a collection of submaps that are meant for simultaneous mapping and planning. Then, we will summarize state-of-the-art planners for unknown-space exploration tasks and their underlying assumptions. Finally, we will conclude with the challenges of making state-of-the-art mapping and planning algorithms work together at the same time.

2.1 Submaps

2.1.1 Submaps and Topological Graph

SLAM or simultaneous localization and mapping is the problem of constructing a map while at the same time localizng against the map. SLAM systems operating in large-scale environments widely use submap-based methods that represent the world as a collection of local maps or *submaps* each with their own local coordinate frames. A topological graph can be constructed from these submaps by associating each submap with a node and connecting adjacent submaps using edges. These submaps decompose environment into smaller manageable pieces such that the robot only need to operate on one of these fixed-size submaps at any given time so as to bound computational complexity, as used by systems in [Yamauchi and Langley, 1996] [Schultz et al., 1998] [Lisien et al., 2003] [Jefferies et al., 2004] [Bosse et al., 2004] Estrada et al. [2005] Fairfield et al. [2010] [Fairfield and Wettergreen, 2010]. In these systems, the robot is localized within the local coordinate system of a submap and gets re-localized when switching submaps.

2.1.2 Submaps and Pose Graph

A natural extension to this idea is to replace the topological graph with the pose graph of smoothing-based pose graph SLAM where each node is associated with a submap and each edge is either a dead-reckoning or a scan-matching constraint between two submaps, as used in Ni et al. [2007] Konolige et al. [2011] Wagner et al. [2014] VanMiddlesworth et al. [2015] Teixeira et al. [2016] [Hsiao et al., 2017]. This construction has the advantage that the global map represented by the pose graph is metrically consistent. In addition, it can be updated efficiently upon loop closures since only the poses of submaps' reference frames have to be updated, and not the submaps themselves. As will be discussed in more detail, this is exactly the same reason that our VOG-Map can be easily deformed upon loop closures.

2.1.3 Exploration and Navigation using Submaps

Of the related works mentioned above, the approaches used in Fairfield et al. [2010], Fairfield and Wettergreen [2010] and Konolige et al. [2011] look especially similar to our approach at first glance because of their use of local occupancy grid maps with either filtering-based or smoothing-based SLAM approach. Fairfield, et al. in Fairfield et al. [2010] use submap-based RaoBlackwellized particle filter SLAM (a.k.a. SegSLAM) for long-term exploration task in unstructured environments. While they also use octree-based occupancy grid to represent submap, they associate submaps with particles, which is different from our approach where each submap is associated with a pose node. As discussed in their paper, this renders their approach not suitable for reconstruction task that requires a globally accurate metric map. In addition, in this thesis, we present ways to use VOG-Map for purpose of path planning whereas how the map produced by SegSLAM can be adopted to work with state-of-the-art planners is left undiscussed.

In Fairfield and Wettergreen [2010], Fairfield, et al. show how SegSLAM can be used with a RRT-based planner in an exploration task. In order to work with the RRT-based planner, their approach first merges submaps into a single occupancy grid map by traversing each octree-based submap and transforming each grid cell into a common coordinate frame. The RRT-based planner then generates plans based on the resulting occupancy grid map. Although not discussed in their paper, merging of all local occupancy grid maps is an expensive operation, and due to their use of particle filter based SLAM method, their approach requires frequent re-merging whenever particles are re-sampled and updated with new measurements. While our approach based on VOG-Map also requires merging of all local occupancy grid maps for planner efficiency, re-merging is only performed upon loop closures, and VOG-Map can be updated incrementally whenever there is new local occupancy grid map and no new loop closure is detected, which happens most of the time. Not only is merging more efficient using our approach, our merged map is also a globally consistent metric map, whereas the merged map using their approach is not. This is the reason why our approach is more suitable for reconstruction task.

On the other hand, Konoliege, et al. in Konolige et al. [2011] show an approach that constructs a map based on local occupancy grid maps and pose graph SLAM. This map is then used for

generating near-optimal path between two arbitrary locations in the map for navigation. Unlike our VOG-Map where there is one-to-one correspondence between local occupancy grid maps and pose nodes, each of their local occupancy grid maps can contain multiple pose nodes. This is the reason that it is much more costly to update their map upon loop closures, as compared to our approach which will be discussed later. Nonetheless, for the purpose of navigation where a global metric map is not required, only some of their local occupancy grid maps are needed to be re-computed immediately after loop closures. Due to the difference in application, their usage of local occupancy grid maps is meant for local shortest path planning, whereas, in this thesis, local occupancy grid maps together with the pose estimates of their respective pose nodes form a global occupancy grid map that is used for global path planning. Since their global path planning is based on the pose graph, their approach requires the environment to be mapped using a pose graph SLAM method as a preprocessing step. This sequential dependence of planning on mapping means their approach is not suitable for autonomous unknown space exploration and mapping, which demands concurrent mapping and planning. Contrary to their approach, our approach based on VOG-Map is capable of mapping and planning at the same time.

2.1.4 Summary

In summary, the use of submaps is fairly common in SLAM literature. Submap or fusion of sensor data in a local region has been used to bound the computational cost in large-scale applications and/or to accumulate enough information for purpose of loop closure detection. In this thesis, as will be discussed later, the reason for using submap is more towards the latter. It is also worth noting that occupancy grid map is just one way to represent a submap, and is commonly used with sonar data [Thrun, 2003] Fairfield et al. [2007]. Occupancy grid map is also useful for planning purpose as will be discussed more in next section. Construction of a global structure based either on topological graph [Fairfield and Wettergreen, 2010] or on pose graph [Konolige et al., 2011] has shown to be useful for both mapping and planning.

2.2 Planning for Unknown-Space Exploration

Occupancy grid map is a space representation that can distinguish between free-space, occupied-space, and unknown-space, therefore making it a popular map representation method in both grid-based [Likhachev et al., 2005] and sampling-based planning [LaValle, 1998] literature. Contrary to the navigational path planner [Konolige et al., 2011] that is based on multiple local occupancy grid maps and a pose graph discussed above, planning approaches Bircher et al. [2016] Papachristos et al. [2017] Vidal et al. [2017] for exploration and mapping of unknown 3D environments, plan paths based on a single global occupancy grid map of the world and the rapidly-exploring random tree or RRT [LaValle, 1998]. The major stages of these planning algorithms involve sampling of potential next-best future viewpoints, followed by computing an information gain value for each potential viewpoint, and finally returning path to the viewpoint that provides maximum information

gain. Of these approaches, the approach implementing "next-best-view" strategy as in Bircher et al. [2016] is shown to be advantageous in terms of exploration efficiency over other "frontier-based" approachesYamauchi [1997]. As detailed later, our planning algorithm is based on the approach proposed in Bircher et al. [2016].

In addition to sampling most informative future viewpoints, an equally important criteria for high fidelity 3D scene reconstruction is the ability to accurately localize the robot within the map. However, all the aforementioned planning methods for unknown scene exploration and reconstruction work well under the assumption that the robot state estimate is accurate at least during the duration of operation. This assumption is justified for aerial robot with access to GPS. However, it is usually not the case for underwater robots since GPS and other global positioning systems are generally not available underwater. Direct application of these approaches would produce poor reconstruction due to drift or accumulated noise.

None of the methods in Bircher et al. [2016] and Vidal et al. [2017] addresses the problem of simultaneously taking into account localization and mapping uncertainties during planning. The key observation is that their use of global occupancy grid maps which cannot be corrected or deformed for drift or accumulated noise, makes the problem of addressing mapping and localization during planning difficult. Papachristos, et al. in Papachristos et al. [2017] extend the approach proposed in Bircher et al. [2016] to also account for minimizing localization and mapping uncertainties. However, they do this minimization as a post-processing stage on an already selected "next-best-view" way-point. A more coherent approach towards minimizing these uncertainties would be leveraging methods from SLAM literature.

2.3 Mapping and Planning

State-of-the-art SLAM methods increasingly utilize smoothing approaches to SLAM as they have proven to be more accurate and efficient as compared to original approaches for SLAM based on nonlinear filtering Cadena et al. [2016]. A popular variation of the smoothing framework is based on pose graph wherein the variables to be optimized are poses along the robot trajectory subject to either dead-reckoning or scan-matching constraint between pose nodes. For doing dense scene reconstruction using a sparse pose graph representation, a typical approach is to attach a submap to each pose node Ni et al. [2007] Konolige et al. [2011] Wagner et al. [2014] VanMiddlesworth et al. [2015] Teixeira et al. [2016] [Hsiao et al., 2017].

The challenge of using the planning approaches in Bircher et al. [2016] Papachristos et al. [2017] Vidal et al. [2017] within a submap-based pose graph SLAM formulation is that these planning methods work based on a single occupancy grid map while pose graph SLAM methods rely on a collection of local submaps for space representation. Another complicating factor is that the map maintained by pose graph SLAM usually does not contain free-space information which is required by planners.

To address these challenges in a non-trivial manner, this thesis proposes a novel approach that

makes use of local occupancy grids within the pose graph SLAM formulation and creates a *virtual* global occupancy grid map or VOG-Map for purposes of robot path planning. VOG-Map maintains a submap-based deformable global map structure but at the same time can be accessed like any standard global occupancy grid map by path planning systems such as Bircher et al. [2016] Papachristos et al. [2017] Vidal et al. [2017]. In fact, VOG-Map is fairly general that it can be used as a drop-in replacement for the global occupancy grid map used by any grid-based [Likhachev et al., 2005] and sampling-based [LaValle, 1998] planning methods.

Chapter 3

Virtual Occupancy Grid Map

This chapter presents details on the *virtual* occupancy grid map or VOG-Map. VOG-Map is at the core of our overall system as highlighted in Fig. 3.1.

State-of-the-art pose graph SLAM approach Kaess et al. [2007] has been shown capable of producing great reconstruction results Teixeira et al. [2016] even in underwater environments where drift or accumulated noise in robot state estimates is significant. State-of-the-art sampling-base planner, on the other hand, enables fully autonomous exploration and mapping in priori unknown unstructured scenes, assuming accurate robot state estimates. VOG-Map provides an unifying map representation that enables our overall system leverages the advantages of these state-of-the-art mapping and planning approaches for autonomous underwater unstructured scene reconstruction.

Unlike a standard global occupancy grid map, the VOG-map representation of the world can be deformed and corrected for globally accumulated drift. It achieves this by design, as VOG-Map in a nutshell is just a set of *local occupancy grid maps* whose base poses correspond to the pose nodes of the smoothing-based pose graph SLAM framework. Whenever the pose graph SLAM optimization happens after adding new constraints, VOG-Map is corrected or deformed accordingly as the base poses of its local occupancy grid maps get updated. The following sections first provide formal definition of VOG-Map, and then suggest ways to construct and implement VOG-Map. Finally, standard and advanced operations that can be performed on the VOG-Map are discussed.

3.1 VOG-Map Definition

To represent VOG-map $\mathcal{M}_{vog}$, consider a set of N *local occupancy grid maps* $\{m_i\}_{i=1}^{N}$ and a set of N base poses in global frame $\{\mathbf{x}_i\}_{i=1}^{N}$. Coordinates of grid cells in each local occupancy grid map m_i are expressed locally with respect to a reference frame placed at the corresponding base pose $\mathbf{x}_i$ of the local occupancy grid map. Here, the term base pose is borrowed from [Teixeira et al., 2016], and refers to the pose with respect to which all sensor data in a local region is fused or merged.

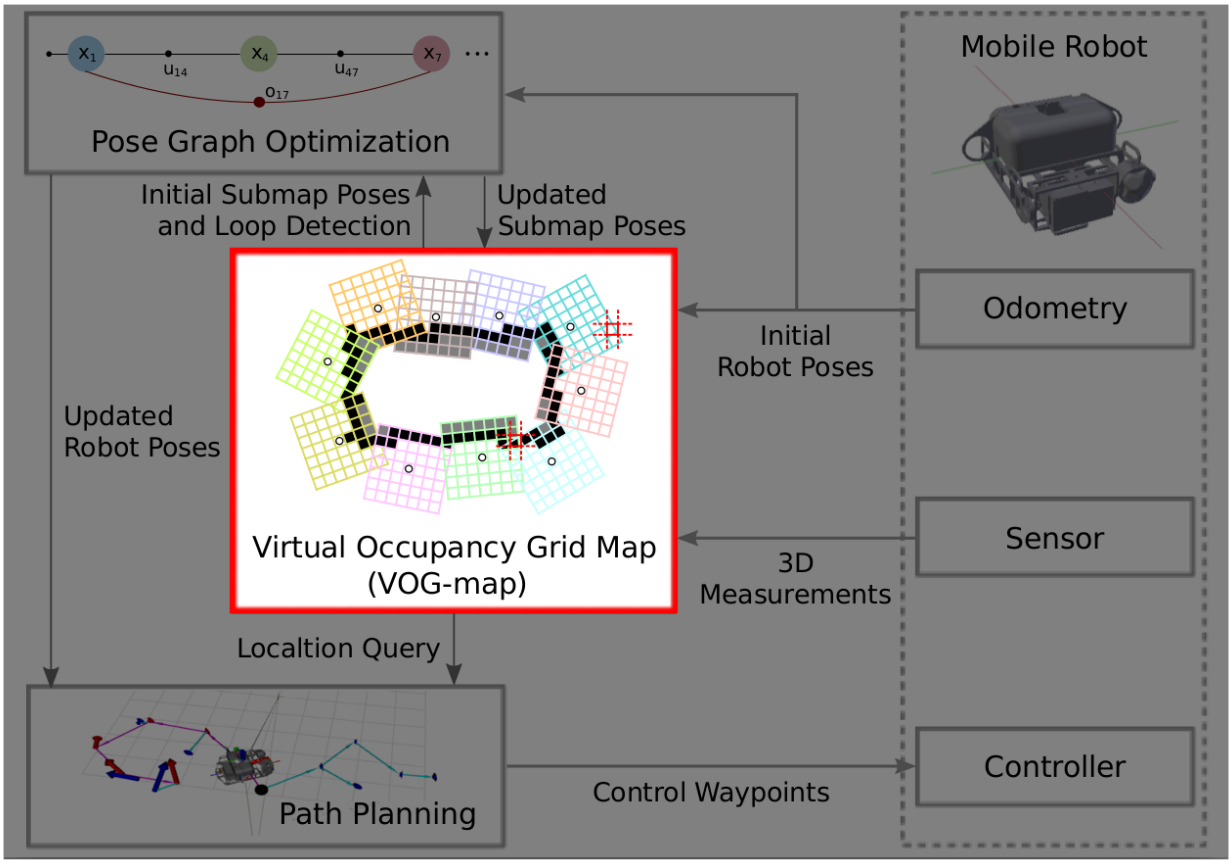

Figure 3.1: Overall system for autonomous unknown underwater unstructured scene reconstruction. This chapter details the *virtual* occupancy grid map or VOG-Map which is highlighted in red. VOG-Map is a collection of local occupancy grid maps, each anchored at a pose node of a pose graph SLAM system.

Let T_i denotes the reference frame, represented in global frame, for local occupancy grid map m_i. By definition, T_i^{global} is represented by the equivalent homogeneous transformation matrix of the 6DOF base pose $\mathbf{x}_i$ of the local occupancy grid map m_i.

The VOG-map $\mathcal{M}_{vog}$ can now be defined as a set of local occupancy grid maps $\{m_i\}_{i=1}^{N}$ along with their corresponding global reference frames $\{T_i^{global}\}_{i=1}^{N}$, that is,

$$\mathcal{M}_{vog} = \left\{ \{m_1, T_1^{global}\}, \{m_2, T_2^{global}\}, \ldots \{m_N, T_N^{global}\} \right\} \tag{3.1}$$

3.2 VOG-Map Construction

The construction of VOG-Map comes down to the construction of each of the local occupancy grid maps $\{m_i\}_{i=1}^{N}$. How these local occupancy grid maps are constructed in turn depends on the

application and sensors used. We present below a way of constructing local occupancy grid maps specifically suitable for the sensor used in our application.

A local occupancy grid map m_i is created by accumulating a set of sequential sensor scans over a finite time period, wherein each scan in this set is registered into a coordinate frame placed at the pose of the first scan. The pose of this first scan is referred to as the base pose $\mathbf{x}_i$ of the resulting local occupancy grid map m_i. Here, we assume that sensor scans are produced by a range-finder sensor like a sonar or a lidar.

When determining the time period Δt for accumulating scans within a local occupancy grid map, the trade-off is that the local occupancy grid map should be large enough to have sufficient features for SLAM loop closure detection but at the same time short enough to bound the accumulated odometry error. For a local occupancy grid map m_i, this time period Δt is computed by keeping the base pose covariance Σ_i below a maximum value, where Σ_i is computed as $\Sigma_i = \Delta t \cdot \Sigma$. Here, Σ is the covariance matrix of measurement uncertainties in our robot's odometry pose estimate.

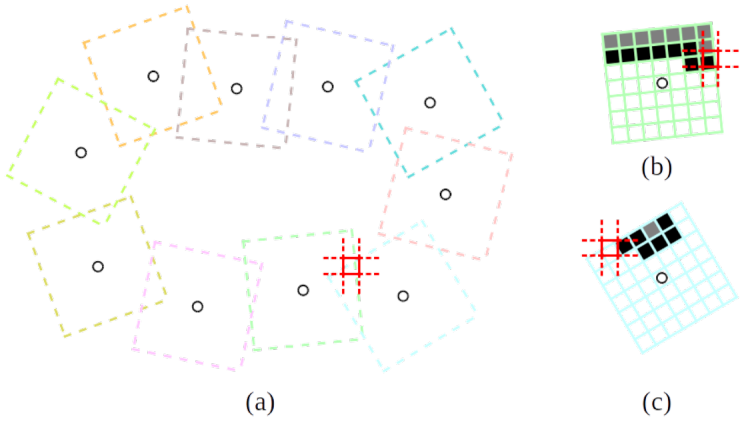

Figure 3.2: Structure of the *virtual* occupancy grid map (VOG-Map). **(a)** VOG-Map consists of a set of local occupancy grid maps m_i (dotted squares), each with a base pose T_i^{global} (black circle) in global frame. **(b)**, **(c)** For a queried global location X^g (red grid), its occupancy value is computed as a sum of log-odds occupancy values of local queries on corresponding local occupancy grid maps.

Incorporation of sensor measurements from a ranger-finder sensor into the VOG-map is done locally within the local occupancy grid map in which these readings are observed. Individual sensor scan readings $z^i_{t_{start}:t_{end}}$ observed in the local occupancy submap m_i during the time span from t_{start} to $t_{end} = t_{start} + \Delta t$ are integrated by performing ray casting operations from sensor scan origins $c^i_{t_{start}:t_{end}}$ to corresponding measurement endpoints in $z^i_{t_{start}:t_{end}}$. The occupancy probabilities $P(v|z^i_{t_{start}:t_{end}})$ for all voxels $v \in m_i$ along each beam are updated according to Moravec and Elfes [1985] and Hornung et al. [2013]

$$P(v|z^i_{t_{start}:t_{end}}) = \left[1 + \frac{1 - P(v|z^i_{t_{end}})}{P(v|z^i_{end})} \frac{1 - P(v|z^i_{t_{start}:t_{end}-1})}{P(v|z^i_{t_{start}:t_{end}-1})} \frac{P(v)}{1 - P(v)}\right]^{-1} \tag{3.2}$$

Using log-odds rule, $L(x) = \log\left[\frac{P(x)}{1-P(x)}\right]$ and assuming uniform prior probability $P(v) = 0.5$, Eq. 3.2 can be simplified to

$$L(v|z^i_{t_{start}:t_{end}}) = L(v|z^i_{t_{start}:t_{end}-1}) + L(v|z^i_{t_{end}}) \tag{3.3}$$

When inserting sensor measurements, it is useful to to use a clamping policy that defines an upper and a lower bound on the occupancy value, that is,

$$L(v|z^i_{t_{start}:t_{end}}) = \max\left(\min\left(L(v|z^i_{t_{start}:t_{end}}), l_{max}\right), l_{min}\right) \tag{3.4}$$

where, l_{min} and l_{max} denote the lower and the upper bound on the log-odds occupancy value. Clamping ensures that confidence in the map remains bounded and also improves runtime efficiency since more neighboring voxels can be compressed via pruning Hornung et al. [2013].

The rules for inserting sensor measurements in Eqs. (3.2)-(3.4) can be used with any kind of ranger finder sensor, as long as an inverse sensor model is available. For instance, for a beam-based inverse sensor model as used in our robotics application later, the ray casting operations from sensor scan origins $c^i_{t_{start}:t_{end}}$ to corresponding measurement endpoints in $z^i_{t_{start}:t_{end}}$ update the voxels at the endpoints based on the sensor's hit probability and all other voxels along the rays based on the sensor's miss probability.

Another factor that should be considered when constructing VOG-Map is the robot motion. Since the error in robot odometry pose estimate usually is dominated by the heading error, instead of using a fixed Δt, determining Δt for each local occupancy grid maps dynamically based on how the robot move in terms of translation and heading changes might be a better strategy. As will be discussed in next chapter, when using sampling-based planner, how the robot moves depends on the sampled way-points, so it is important to also consider how way-points are sampled when deciding the strategy used for the construction of local occupancy grid maps.

3.3 VOG-Map Implementation

Since the idea of VOG-Map is based on occupancy grid maps and pose graph SLAM, VOG-Map can be implemented using the OctoMap framework Hornung et al. [2013] and the iSAM optimization library Kaess et al. [2007]. OctoMap is an octree-based occupancy grid space representation that can model free as well as occupied volumes and also implicitly volumes that have not been measured. The octree data structure used underneath makes OctoMap an efficient representation for real-time robotics applications in 3D environments. iSAM, on the other hand, is an optimization library for sparse nonlinear problems as encountered in simultaneous localization and mapping (SLAM). iSAM provides algorithms that can efficiently recover the robot trajectory or robot pose estimates of the corresponding pose graph SLAM problem, making it suitable for real-time application.

By implementing our local occupancy grid maps based on OctoMap, VOG-Map can be stored efficiently, both in memory and on disk. Thanks to the use of OctoMap, standard map operations like occupancy query can be performed on each local occupancy grid map efficiently. However, as will be discussed in 3.4.2, the efficiency of standard operations performed on the VOG-Map generally depends on the number of local occupany grid maps. On the other hand, by using iSAM optimiza-

tion library, VOG-Map can be deformed or updated efficiently upon loop closures as discussed in 3.4.1.

3.4 VOG-Map Standard Operations

VOG-Map standard operations include VOG-Map deformation and VOG-Map occupancy query. VOG-Map deformation operation changes the base poses of local occupancy grid maps, thereby allowing VOG-Map to deform upon loop closures. VOG-Map occupancy query operation returns either the occupancy value or occupancy status of some queried location in space. We detail these two operations below.

3.4.1 VOG-Map Deformation

VOG-Map deformation operation allows the underlying spatial arrangement of local occupancy grid maps to be changed efficiently by replacing the base poses of local occupancy grid maps with other poses as input parameters. This operation is general but meant to be used to update the base poses of local occupancy grid maps upon loop closure using the estimated poses obtained from pose graph SLAM optimization. The runtime of this operation is linear in the number of local occupancy grid maps in VOG-Map.

Algorithm 1 VOG-Map Deformation

1: **Input:** poses arr_poses
2: $N \leftarrow$ total number of local occupancy grid maps
3: **for** $i = 1$ to N **do**
4: $\quad \mathbf{x}_i \leftarrow arr\_poses[i]$

Please note that synchronization mechanism is intentionally omitted in Algorithm 1 for clarity. Since VOG-Map occupancy query operation requires read access to local occupancy grid maps and their respective base poses, a mutex should be used when replacing the bases poses of local occupancy grid maps.

3.4.2 VOG-Map Occupancy Query

VOG-Map occupancy query operation receives as input coordinate values of a point in space, and output the occupancy value or the occupancy status of the queried point in space. Occupancy value refers to the probability of occupancy while occupancy status refers to the discrete states of being free, occupied, and unknown. Here, we discuss two strategies that can be used to perfom VOG-Map Occupancy Query.

Sequential VOG-Map Occupancy Query

Sequential VOG-Map occupancy query compute the occupancy value or occupancy status of queried point by iterating through all local occupancy grid maps one by one. An input query as a 3D position in global frame is first converted into corresponding queries with respect to local occupancy grid maps' reference frames $\{T_i^{global}\}_{i=1}^{N}$. These queries are then passed onto their respective local occupancy grid maps. Let X^g be the global 3D position of the queried point in space whose occupancy value needs to be looked up. We use the 4×4 homogeneous transformation matrix T_i^{global} to map X^g into a local coordinate X^i computed as $\bar{X}^i = (T_i^{global})^{-1}\bar{X}^g = T_g^i\bar{X}^g$ (in homogeneous coordinates) which is then passed as an occupancy query to local occupancy grid map m_i. This is done for all local occupancy grid maps $\{m_i\}_{i=1}^{N}$. The transformation matrix T_g^i is computed by taking the inverse of the homogeneous transformation matrix T_i^{global} for each local occupancy grid map m_i.

The occupancy probability values returned from all local occupancy grid maps $\{m_i\}_{i=1}^{N}$ can now be combined together using the same log-odds update rule as seen in Eq. 3.3. This is because combining measurements from multiple local occupancy grid maps is a similar operation as combining multiple measurements in a standard occupancy grid map Hornung et al. [2013]. Also, since every sensor measurement is incorporated only once (in any one of the local occupancy grid maps) by construction, we do not run the risk of double counting measurements. The log-odds occupancy probability for a global 3D location X^g is hence expressed as

$$\begin{aligned} L(X^g) &= \sum_{i=1}^{N} L_i(X^i) \\ L(X^g) &= \max\left(\min\left(L(X^g),\, l_{max}\right),\, l_{min}\right) \end{aligned} \tag{3.5}$$

$L_i(\cdot)$ implies that the log odds lookup is done in local occupancy grip map m_i. l_{min} and l_{max} are the same clamping thresholds as used in Eq. 3.4.

Algorithm 2 Sequential VOG-Map Occupancy Query

Input: coordinate values of queried point X^g
Output: log-odds occupancy probability $L(X^g)$
$L(X^g) \leftarrow 0$
$N \leftarrow$ total number of local occupancy grid maps
for $i = 1$ to N **do**
    $T_g^i \leftarrow \text{Inverse}(T_i^{global})$
    $\bar{X}^i \leftarrow T_g^i\bar{X}^g$
    $L(X^g) \leftarrow L(X^g) + L_i(X^i)$
$L(X^g) \leftarrow \max\left(\min\left(L(X^g), l_{max}\right), l_{min}\right)$
return $L(X^g)$

Please note that synchronization mechanism is intentionally omitted in Algorithm 2 for clarity. Since VOG-Map occupancy deformation operation requires write access to local occupancy grid

maps and their respective base poses, a mutex should be used when iterating through local occupancy grid maps.

Sequential VOG-Map occupancy query operation provides a way to retrieve the occupancy value without having to maintain a global representation of the underlying local occupancy grid maps. This has the advantage of avoiding storage overhead. However, its runtime is linear in the number of local occupancy grid maps in VOG-Map.

The term *virtual* in *virtual* occupancy grid map refers to the property of VOG-Map that VOG-Map query operation of a location in global frame can be done without having to merge all the local occupancy grid maps. This property is desirable if the number of queries is small. If many queries need to be performed on the VOG-Map, merged VOG-Map occupancy query operation as detailed in next section is a more efficient alternative whose amortized runtime is constant in the number of local occupancy grid maps, but at the cost of storage overhead.

Merged VOG-Map Occupancy Query

Merged VOG-Map occupancy query operation requires merging all the local occupancy grid maps into one global occupancy grid map. The key insight is this merging only needs to be done once for the first query. Therefore, its amortized runtime is constant with respect to the number of local occupancy grid maps, at the cost of having to cache the global occupancy grid map.

Algorithm 3 Merged VOG-Map Occupancy Query

Input: coordinate values of queried point X^g
Output: log-odds occupancy probability $L(X^g)$
$L(X^g) \leftarrow 0$
$G \leftarrow$ cached global occupancy grid map
if G not available **then**
    $G \leftarrow$ MergeLocalOccupancyGirdMaps()
$L(X^g) \leftarrow$ GetLogOddsOccupancy(G, X^g)
$L(X^g) \leftarrow \max\left(\min\left(L(X^g), l_{max}\right), l_{min}\right)$
return $L(X^g)$

GetLogOddsOccupancy function is a standard operation provided by OctoMap Hornung et al. [2013]. The next section will detail how to merge local occupancy grid maps and compare sequential VOG-Map occupancy query operation and merged VOG-Map occupancy query operation in terms of the occupancy values returned.

Merge VOG-Map

When the number of VOG-Map queries is large, sequential VOG-Map occupancy query operation can incur a lot of runtime overhead since every time it is called, it has to iterate through all the local

occupancy grid maps. Merged VOG-Map occupancy query operation can be a better alternative if storage space is not a concerned.

The merge function first discretizes the global space into smaller voxels based on the resolution used for local occupancy grid maps. Then, it looks up the occupancy of these voxels one by one using sequential VOG-Map occupancy query operation (Algorithm 2), and insert the result into a global occupancy grid map.

Algorithm 4 Merge Local Occupancy Grid Maps

1: **Output:** global occupancy grid map O
2: $V \leftarrow$ array containting coordinate values of the voxels
3: $O \leftarrow$ initialized global occupancy grid map
4: **for** $v \in V$ **do**
5: $\quad l \leftarrow$SequentialVOGMapQuery(v)
6: $\quad$SetOccupancyLogOdds(O, v, l)
7: **return** O

SetOccupancyLogOdds function is a standard operation provided by OctoMap Hornung et al. [2013]. Since the merge function relies on the sequential VOG-Map occupancy query operation, it should be intuitive to see that the occupancy values returned by sequential VOG-Map occupancy query operation and merged VOG-Map occupancy query operation should be very similar, if not the same. Fig. 3.3 shows the comparison of queried results using these two VOG-Map occupancy query operations in terms of misclassification percentage computed with respect to the standard global occupancy grid map. As can be seen, the two misclassification percentages are indeed very similar, and the small discrepancy comes from using different queried points for comparison from those used (the voxels in Algorithm 4) for merging.

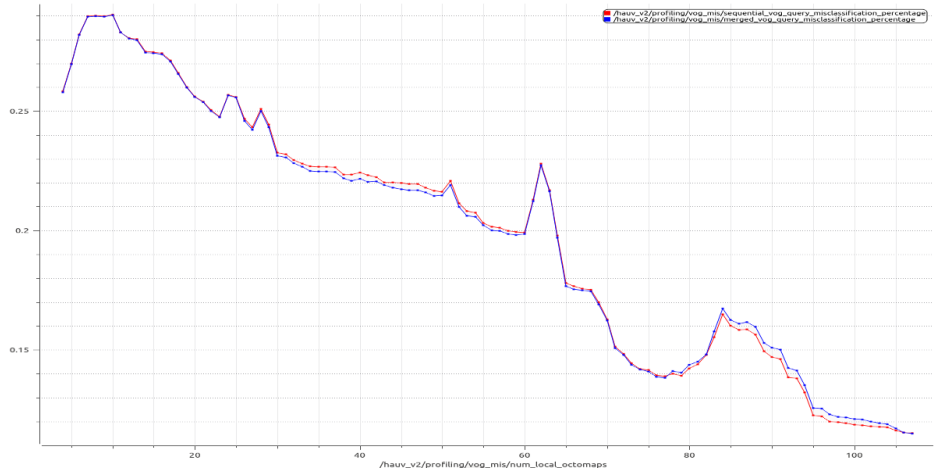

Figure 3.3: The number on x-axis is the total number of local occupancy grid maps in the VOG-Map. The number on y-axis is computed by dividing the number of mis-classified queried points by the total number of queried points. The red line corresponds to the results using sequential VOG-Map query operation while the blue line corresponds to the results using merged VOG-Map query operation.

3.5 VOG-Map Advanced Operations

We have shown how VOG-Map can work with pose graph SLAM through standard operation, i.e. VOG-Map deformation operation 3.4.1. Also, we have shown how to retrieve occupancy value of some location in global frame using standard operation 3.4.2. These operations are fairly application-agnostic. Here, we will discuss a variant of query operation called VOG-Map line query operation. VOG-Map line query operation takes as input two points in global frame and returns the occupancy values for all voxels along the line connecting the two points.

3.5.1 VOG-Map Line Query

Occupancy queries to the VOG-Map are typically made by the planner in form of ray casting queries, i.e. casting a ray into space along a given direction and returning occupancy values for voxels along that ray. Planning algorithms require such ray casting queries for computing collision-free paths and view utility gains.

The process of performing ray casting query or line query on VOG-map is illustrated in Algorithm 5. It begins by finding a subset $\mathcal{S}$ of the local occupancy grid maps $\{m_i\}_{i=1}^{N}$ that intersect with the line formed by p_1 and p_2 in global frame. It then steps along the line and computes occupancy value for each voxel using the sum of log odds rule as expressed in Eq. 3.5. As a result, the summation in Eq. 3.5 is taken only over a subset $\mathcal{S}$ of local occupancy grid maps that intersect with the line instead of all the local occupancy grid maps.

Algorithm 5 VOG-Map Line Query Operation

Input: two points in global frame p_1, p_2
Output: occupancy values $\mathcal{O}$ along the line formed by p_1 and p_2
$\mathcal{O} \leftarrow \emptyset$,
$\Delta s \leftarrow$ step size (usually just the resolution of each local occupancy grid map)
$n \leftarrow$ number of steps computed by dividing the length of the line by the step size
$v \leftarrow$ direction vector from p_1 to p_2
$\mathcal{S} \leftarrow$ GetIntersectedLocalOccupancyGridMap(p_1, p_2)
for $k \leftarrow 1$ to n **do**
    $X^g = p_1 + (k\Delta s)v$
    $L(X^g) \leftarrow 0$
    for $i \in \mathcal{S}$ **do**
        $T_g^i \leftarrow$ Inverse(T_i^{global})
        $\bar{X}^i \leftarrow T_g^i \bar{X}^g$
        $L(X^g) \leftarrow L(X^g) + L_i(X_i)$
    $L(X^g) \leftarrow \max(\min(L(X^g),\ l_{max}),\ l_{min})$
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{L(X^g)\}$
return $\mathcal{O}$

GetIntersectedLocalOccupancyGridMap function can be implemented efficiently based on the line and axis-aligned bounding box test. VOG-Map line query operation exploits the structure of VOG-Map and the queried line to reduce the number of local occupancy grid maps that need to be looked up for each voxel along the line. Fig. 3.4 shows a comparison in terms of the time taken for each planning iteration between a planner using only sequential VOG-Map occupancy query operation and a planner using VOG-Map line query operation whenever possible. As can be seen from the figure, planner using VOG-Map line query operation is significantly more efficient. However, as will be discussed in more detail in next chapter, even with the use of VOG-Map line query operation, planning based on the VOG-Map can be very slow as each planning iteration requires way more than a few seconds. As will be discussed later, we can make planning based on VOG-Map always run in real-time by using merged VOG-Map occupancy query.

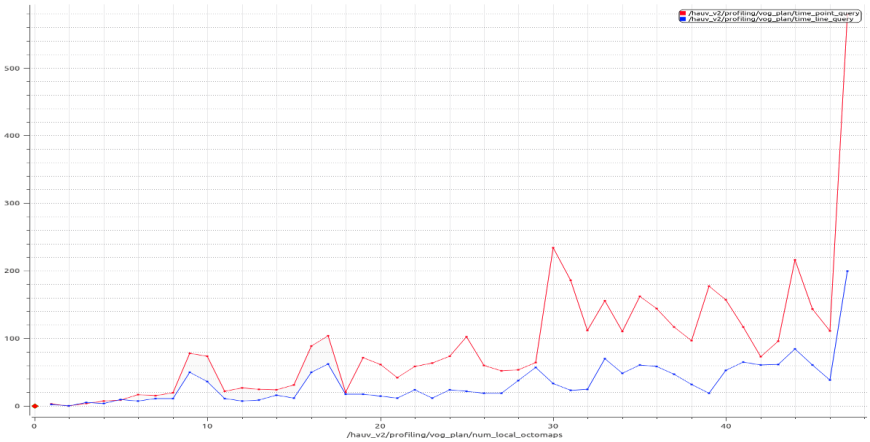

Figure 3.4: The number on x-axis is the total number of local occupancy grid maps in the VOG-Map. The number on y-axis is the time taken to plan in seconds. The red line corresponds to the results using only VOG-Map point query operation while the blue line corresponds to the results using VOG-Map line query operation whenever possible. Sequential VOG-Map occupancy query operation is used one query at a time so an alternative name for it is VOG-Map point query operation.

Chapter 4

Robotics Application

This chapter shows how VOG-Map can be used to implement an approach that enables a marine robot to autonomously explore and map in underwater unstructured scenes. The advantage of using VOG-Map over a standard global occupancy map is that it can be easily deformed and corrected for drift or accumulated noise. This property of VOG-Map is ideal especially in underwater environment where robot state estimates are uncertain and global position systems are unavailable.

This chapter is organized as follows. We will first introduce the marine robot we use for our reconstruction task. Then, we discuss the algorithms we used and the changes we made. Finally, we evaluate our system both in simulation and real-world experiments in terms of reconstruction quality.

4.1 Platform

A schematic view of the marine robot we used and its sensors payloads is shown by Fig. 4.1. *Hovering Autonomous Underwater Vehicle* or HAUV is a marine robot developed by MIT/Bluefin. It was also used for ship reconstruction task in both Hover et al. [2012] and Teixeira et al. [2016]. Since our mapping algorithm is based on the sonar-based volumetric submaps approach presented in Teixeira et al. [2016], we use the same sensor payloads as well as the same sensor configurations. The following detailed description of HAUV is directly taken from Teixeira et al. [2016] with minor modifications:

> HAUV is equipped with five rim-driven thrusters that make it directly controllable in all axes but roll and pitch. Its navigation payload comprises a Honeywell HG1700 inertial measurement unit (IMU), a 1.2MHz Teledyne/RDI Workhorse Navigator Doppler Velocity Log (DVL), and a Paroscientific Digiquartz depth sensor. The relevant performance characteristics are summarized in table 4.1. The DVL can rotate parallel to the vehicles pitch axis, allowing for both bottom- and hull-relative motions. The HAUV's primary payload is a dual-frequency identification sonar (DIDSON) [Belcher et al.,

Table 4.1: Navigation Sensor Performance

Sensor	Axis	Accuracy	Unit
IMU	Roll, Pitch	0.02	°
	Yaw	0.05	°
DVL	X, Y, Z	0.003	m/s
Depth	Z	0.01	%

> 2002], which we use primarily in its high frequency (1.8MHz) mode, as it provides greater detail. It has a total of 96 beams, each with 0.3° of horizontal beam width. Since we are interested in complex geometries, we make use of a concentrator lens, reducing its vertical field of view from the standard 14° to 1° (-3dB values). While its range resolution depends on the minimum and maximum range configuration, a typical window of 9.5 meters will yield a resolution of better than 2 centimeters.

The takeaway from the excerpt is that the depth sensor provides absolute measurement along z-axis and the IMU provides absolute roll and pitch measurements. Since magnetometer is not used, measurement of yaw is not drift-free. And, since the heading estimate drifts over time, the position of HAUV along x-axis and y-axis is bound to drift over time too even though DVL is used. The reason is although DVL provides accurate velocity estimates with zero-mean bias, integration of these velocity estimates into position estimates relies on heading estimates. So, the 6 degrees of freedom (DOF) pose estimation is reduced to only 3 DOF, i.e. x, y, and yaw. Another important takeaway from the excerpt is the sensor used for perception. The DIDSON sonar comprises an array of transducers that produce one or more narrow beams. Each beam is a cone-like structure with 0.3° horizontal beam width and 14° vertical beam width. But, with concentrator lens, each beam now approximates a single line into space with 0.3° horizontal beam width and 1° vertical beam width. The usage of concentrator lens removes much of the ambiguity along beam's vertical field of view and allows us to treat the range measurement from each beam as corresponding more or less to some unique point in space. With 96 beams, each sonar scan swipes everything in an arc in the horizontal plane. After denoising, filtering, enhancement, and range extraction as presented in [Teixeira et al., 2016], each sonar scan is represented by a set of points in the horizontal plane.

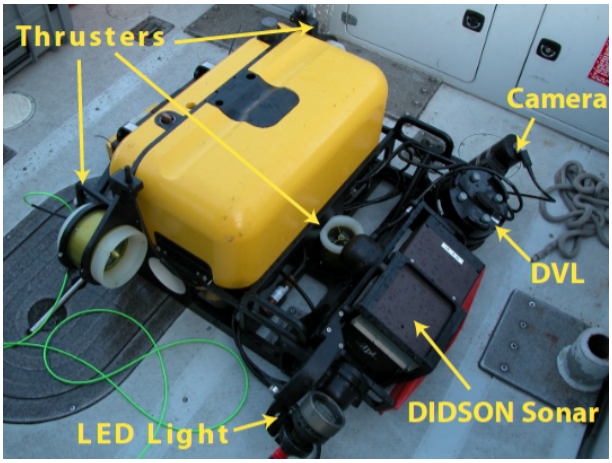

Figure 4.1: Hovering Autonomous Underwater Vehicle

Since HAUV cannot roll or pitch, the Didson sonar is rotated 90° along x axis as illustrated by Fig. 4.2. In this setup, HAUV can rotate in place while gathering 3D volumetric data of its surrounding, which is important for unknown-space exploration.

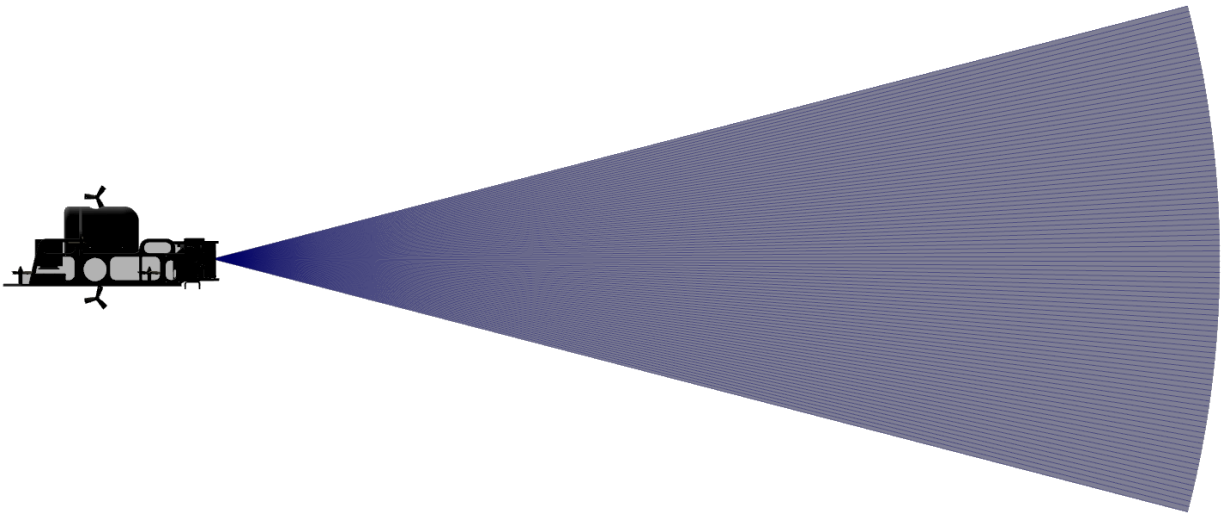

Figure 4.2: Simulated HAUV and simulated sonar scan as viewed from the side. The blue cone comprised of 96 beams each is modeled as a line. While a line is not an accurate representation of how the actual sonar beam filtered by a concentrator lens looks like, the blue cone should give an idea of how the actual sonar scan looks like using our sonar setup.

4.2 VOG-Map and Mapping

In this section, we will describe the main components of the sonar-based volumetric submaps approach [Teixeira et al., 2016] which we use as the basis of our mapping algorithm. In particular, we will focus on how VOG-Map can be constructed upon Pedro's mapping approach, or more generally upon the smoothing-base pose graph SLAM framework.

As discussed in 3.2, we assemble consecutive sonar scans to form local occupancy grid maps. Fig. 4.3 shows an example of local occupancy grid map assembled from real sonar scans. To understand how the local occupancy grid map is constructed, it is important to note that the HAUV initially sits at the bottom-left corner of Fig. 4.3. Then, HAUV rotates in place for about 90° while constructing the local occupancy grid map. Once the local occupancy grid map is created, we say it is anchored at a pose node of the pose graph SLAM [Kaess et al., 2007] when we create the said pose node using the base pose of the local occupancy grid map. With multiple local occupancy grid maps and their respective bases poses, a pose graph as illustrated by Fig. 4.4 is constructed. The binary constraint between consecutive pose node is obtained by computing the relative transformation between two consecutive base poses along x, y, and yaw directions. This is the dead-reckoning constraint based on HAUV's odometry. The unary constraint associated with each pose node is ob-

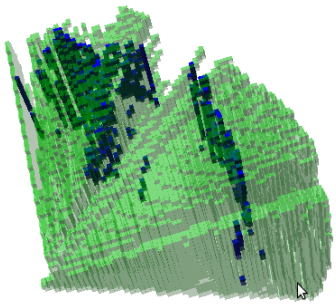

Figure 4.3: Local occupancy grid map assembled from consecutive real sonar scans. Blue voxels represent occupied-space and green voxels represent free-space.

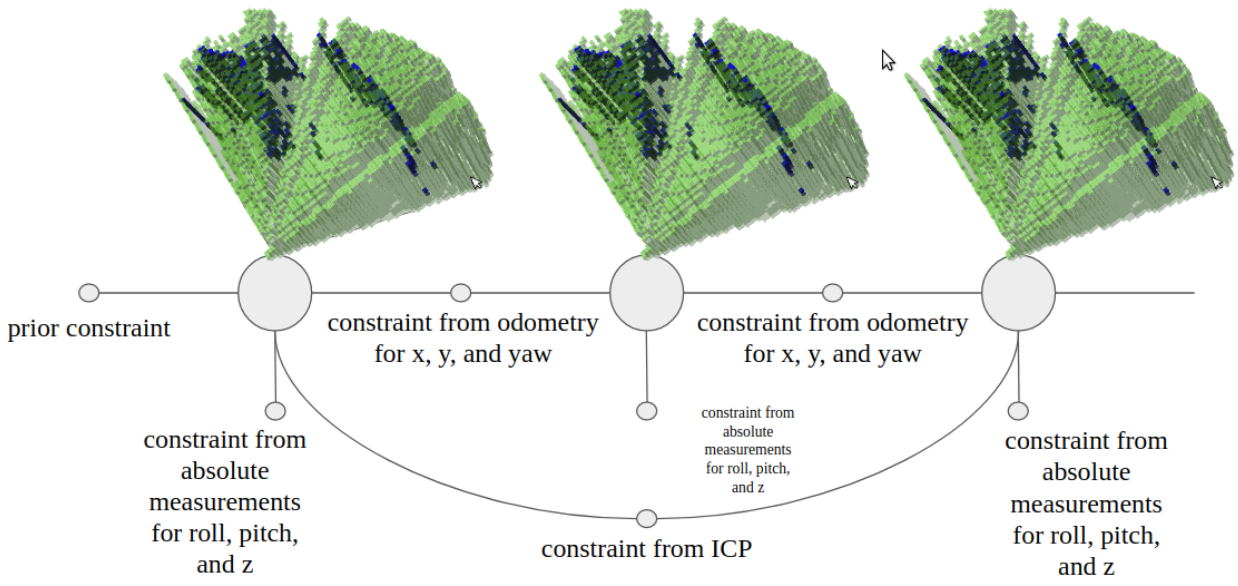

Figure 4.4: The pose graph as formulated in [Teixeira et al., 2016] upon which VOG-Map is built. When the SLAM optimization updates the pose estimates of the pose nodes, base poses of the local occupancy grid maps are also updated using VOG-Map deformation operation as discussed in 3.4.1 which effectively and efficiently corrects the VOG-Map for drift or accumulated noise.

tained by using the absolute measurements along roll, pitch, and z. Another type of binary constraint is obtained by computing the relative transformation between two local occupancy grid maps via iterative closest point (ICP).

ICP is a method that tries to bring two point-clouds together by minimizing some error metric. Therefore, to make ICP more efficient, we cache a point-cloud representation for each local occupancy grid map. Since local occupancy grid maps are never modified and only their base poses are changed, their point-cloud representations once created will not need to be changed either. As an example, Fig. 4.5 shows the corresponding point-cloud representation of the local occupancy grid map shown in Fig. 4.3.

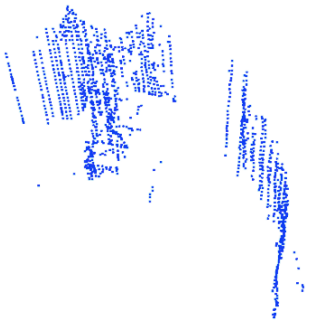

Figure 4.5: Point-cloud representation of the local occupancy grid map shown in Fig. 4.3. It is obtained by down-sampling the raw point-cloud data collected from a set of consecutive sonar scans.

So far, the formulation of the pose graph SLAM problem is exactly the same as presented in [Teixeira et al., 2016]. We have shown how VOG-Map can be constructed upon the pose graph of the SLAM problem. By building VOG-Map upon the pose graph SLAM framework, VOG-Map can be deformed efficiently as the bases poses of the underlying local occupancy grid maps get updated based on the optimized pose estimates from SLAM optimization. Therefore, the quality or accuracy of VOG-Map depends on the pose graph SLAM optimization, which in turn depends on the ICP method used which will be discussed in next section.

4.2.1 Point-to-Point Iterative Closest Point

In this section, we will discuss how to find potential relative pose constraint between any two local occupancy grids via the ICP method presented in Teixeira et al. [2016]. It is important to note that the ICP method is performed on the corresponding point-cloud representations of the local occupancy grid maps.

Since the pose estimation is a 3 DOF problem along x, y, and yaw, as described in 4.1, the 2D Point-to-Point ICP method provided by Point Cloud Library [Rusu and Cousins, 2011] is used. This method iteratively find the rotation (yaw only) and translation (x and y only) that aligns two set of corresponding points. A detailed derivation of the math behind can be founded in [Sorkine-Hornung and Rabinovich, 2017].

The key insight is if perfect correspondences between points in two point-clouds are known, the formula derived in [Sorkine-Hornung and Rabinovich, 2017] can find the best-fitting rigid transformation that aligns two point-clouds in one step. When data association is not available, ICP methods are able to converge to the best-fitting rigid transformation only if good initial relative transformation is available. The strategy used in Teixeira et al. [2016] for deciding whether or not a good initial relative transformation is available is to first transform the source point-cloud into the reference frame of the target point-cloud using the initial relative transformation from odometry. Then, centroids of the two point-clouds are computed. If either of the two centroids is inside the bounding box of the other point-cloud, ICP is performed on these two point-clouds. Otherwise, ICP is skipped for the two point-clouds. Fig. 4.6 illustrates the strategy used for determining good initial relative transformation.

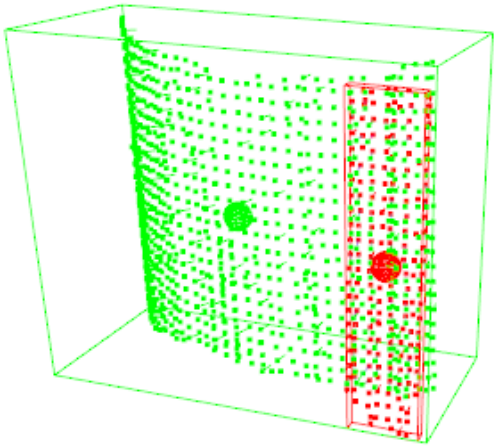

Figure 4.6: Source point-cloud is in red, with the red sphere denotes its corresponding centroid. Target point-cloud is in green, with the green sphere denotes its corresponding centroid. The source point-cloud is first transformed into the coordinate frame of the target point-cloud. Then, the centroids and bounding boxes of the two point-clouds are computed. Since the centroid of the source point-cloud is inside the bounding box of the target point-cloud, the initial relative transformation between these two point-clouds is considered good.

This precondition avoids unnecessary computation overhead of performing ICP on two point-clouds that do not have good initial relative transformation. To make the result of ICP more robust, even if ICP converges, some criteria need to be met for the ICP result to be accepted. Algorithm 6 summarizes the important components of the ICP pipeline as used in [Teixeira et al., 2016].

2D-Point-to-Point-ICP is considered converged - criterion **(1)** if a maximum number of iterations has been reached, or the difference between consecutive estimated transformation is smaller than a threshold, or the mean squared error (MSE) between the current set of correspondences and the previous one is smaller than some threshold. If 2D-Point-to-Point-ICP converges, we first

Algorithm 6 ICP Pipeline

Input: source point-cloud $PC1$, target point-cloud $PC2$, initial relative transformation T_0
Output: relative transformation T
$T \leftarrow null$
if $PC1$ dose not have enough points **or** $PC2$ does not have enough points **then**
    return T
if centroid of $PC1$ not in bounding box of $PC2$ **and**
  centroid of $PC2$ not in bounding box of $PC1$ **then**
    return T
converged $\leftarrow$ 2D-Point-to-Point-ICP($PC1$, $PC2$, T_0, T)
if (1) converged **and (2)** averaged sum of squared distance
  between corresponding points after ICP trasformation is less than some threshold **and**
  (3) the difference between T and the relative transformation obtained from odometry
  is smaller than the maximum possible drift computed based on the covariance of
  odometry measurements and the time difference betwee the two point-clounds **then**
    return T
else
    return $null$

transform the source point-cloud into the coordinate frame of the target point-cloud using the relative transformation obtained from ICP. Then, we find the correspondences between source and target point-clouds. Finally, we add up the squared distance between each pair of points and take average. This number is also called score and can be retrieved using PCL API. This score check corresponds to the criterion **(2)** in the pseudocode above. Criterion **(3)** is a bit more complicated and is implemented as follows: Let T(x) be the estimated translation along x-axis from ICP. Let O(x) be the translation along x-axis from odometry. Let T(y) be the estimated translation along y-axis from ICP. Let O(y) be the translation along y-axis from odometry. Then, we compute dx = T(x) - O(x) and dy = T(y) - O(y). The difference between T and the relative transformation obtained from odometry is $\delta = \sqrt{dx^2 + dy^2}$. Let dt be the time elapsed since the construction of source point-cloud till the construction of target point-cloud. Let Cov(x) be the assumed odometry measurement covariance along x-axis per second. Let Cov(y) be the assumed odometry measurement covariance along y-axis per second. Then the maximum possible drift is $\sigma = \sqrt{\text{Cov(x)} * \text{dt} + \text{Cov(y)} * \text{dt}}$. If δ is greater than σ, then the ICP result is rejected. Otherwise, the criterion is met. Once the ICP converges and all the criteria are met, a relative pose constraint based on the relative transformation from ICP is created and added to the pose graph.

4.2.2 Degeneracy

The criteria in Algorithm 6 are used in order to avoid wrong ICP registration as shown in Fig. 4.7. Wrong registration can happen even when good initial relative transformation is available if the underlying optimization problem solved by ICP is degenerate. As defined in [Zhang et al., 2016],

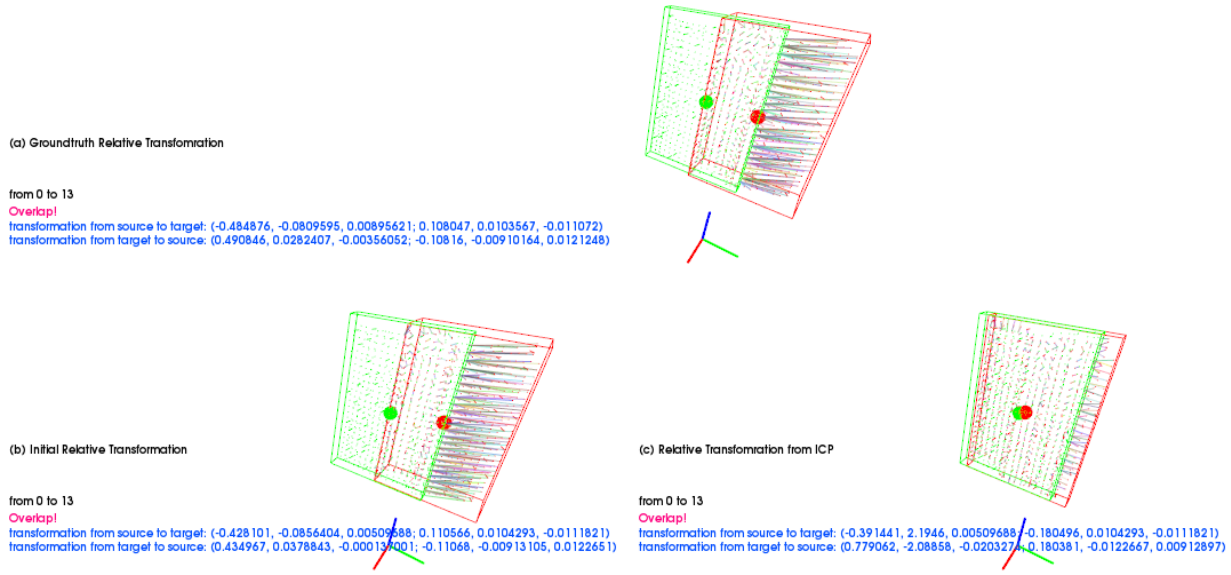

Figure 4.7: Illutration of wrong relative transformation obtained from ICP due to degeneracy in the underlying optimization problem. Since these submaps are obtained in simulation, the groundtruth relative transformation between source point-cloud (red) and target point-cloud (green) is available and is used to visualize the two point-clouds in target-cloud coordinate frame as shown in **(a)**. **(b)** shows the result of using initial relative transformation from odometry, and **(c)** shows the result of using relative transformation from wrong ICP registration.

an optimization problem is degenerate if some direction in the state space of the optimization problem is not well-conditioned. A direction is not well-conditioned if there are no enough constraints along that direction. In Fig. 4.7, since some direction in the horizontal plane formed by x-axis and y-axis is not well-conditioned, the optimization problem is degenerate, resulting in wrong relative transformation from ICP.

Degeneracy is a common problem especially in scenes that lack geometrical structures. When ICP is performed on submaps assembled from range sensor data (when using sonar or lidar) in such scenes, the underlying optimization problem often contains one or more ill-conditioned direction in the state space of the variables for which ICP tries to optimize. It is important to avoid wrong ICP registrations because incorrect relative pose constraints obtained from wrong ICP registrations will in turn result in erroneous pose estimates from SLAM optimization problem, causing both the VOG-Map and the scene reconstruction to deform incorrectly.

While the ICP pipleine in Algorithm 6 determines and discards bad ICP registrations using heuristics, a more robust way is to separate well-conditioned directions from ill-conditioned directions in the state space of the optimization problem and update the solution only along the well-conditioned directions as presented in [Zhang et al., 2016]. As will discussed in 4.5.1, we improve the ICP pipeline of Teixeira et al. [2016] and make it more robust to degeneracy in the scene by incorporating ideas from [Zhang et al., 2016] into our custom ICP methods.

4.3 VOG-Map and Planning

As mentioned in 1.2, we base our planning algorithm on Bircher's sampling-based exploration approach detailed in [Bircher et al., 2016]. In this section, we discuss the key components of the planning algorithm with a focus on the changes we made.

The key compoents of Bircher's sampling-based exploration approach include **(1)** a space representation that can distinguish between free-space, occupied-space, and unknown-space, **(2)** a sampling technique for generating collision-free paths, and **(3)** a strategy for computing the information gain of each waypoint.

4.3.1 Space Representation, Way-point Sampling, and Gain Computation

The space representation maintained by Bircher's planning algorithm is an octree-based occupancy grid map implemented using OctoMap Hornung et al. [2013]. This occupancy grid map represents the entire exploration bounding box within which the robot operates. If the robot state estimate is accurate, using one occuapncy grid map to represent the entire area of interest works well as shown in [Bircher et al., 2016]. However, with uncertain robot state estimates, the occupancy grid map will not be an accurate space representation over time due to drift or accumulated noise. While our VOG-Map also suffers from drift or accumulated noise, it can be deformed or corrected efficiently upon loop closure detection via ICP method as described in the previous section. Therefore, we replace the occupancy grid map used by Bircher's planning algorithm with our VOG-Map.

Bircher's planning algorithm builds upon rapidly-exploring random tree or RRT LaValle [1998] to generate paths for exploration and mapping. RRT is a sampling-based algorithm designed for efficient space exploration by building a space-filling tree. During each iteration of RRT, a new way-point w_{new} is sampled, and the closest way-point $w_{closest}$ in the tree usually in terms of euclidean distance is found. w_{new} is then updated so that the distance between w_{new} and $w_{closest}$ is within some user-defined maximum extension range between adjacent way-points. w_{new} is added to the tree only if the path segment from $w_{closest}$ to w_{new} is collision-free. To check whether or not it is collision free, multiple ray-tracing or line queries need to be performed on the VOG-Map. This can be done by using VOG-Map line query operation as described in 5.

To determine where the robot should move once the way-points are sampled, we need to compute for each way-point the expected reward or the information gain. For the computation of a way-point's information gain, sensor measurements as viewed from the way-point need to be simulated based on the sensor model. We simulate sonar measurements in sonar's local frame using some simple trigonometry based on the horizontal field of view, vertical field of view, minimum range, and maximum range of our sonar. These simulated measurements in local frame are then converted into global frame using the way-point's global pose. Once these simulated measurements are in global frame, we query each simulated measurment for its corresponding occupancy status using VOG-Map sequential or point query operation as described in 2, and increment the information

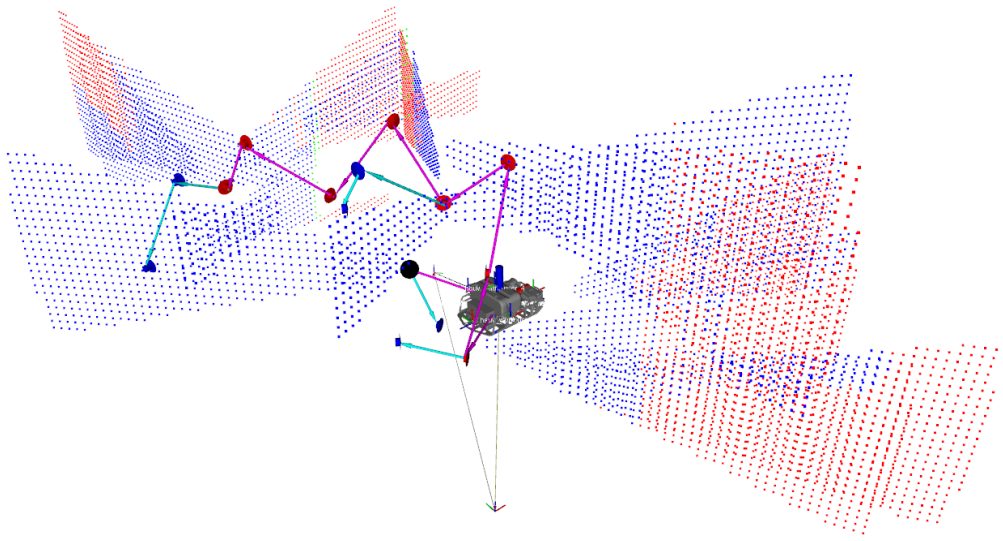

Figure 4.8: Illutration of sampled paths using RRT and the simulated measurements used for information gain computation. Cyan and magenta arrows show the path segments between way-points. Blue and red arrows/disks show the way-points. For each way-point, there is a corresponding cone-like structure that is made up of points with different colors. Blue point means the measurement corresponds to free-space in the VOG-Map. Green point means the measurement corresponds to occupied-space in the VOG-Map. Red point means the measurement corresponds to unknown-space in the VOG-Map. The information gain for a way-point is then computed by adding up the value associated with each point in the corresponding cone. Some cones are incomplete because when computing information gain for a way-point, invisible simulated measurement points as viewed from the way-point are discarded. Since a sensor measurement is considered invisible when it is behind occupied-space, these cones terminate at green points or occupied-space. Some points can still appear after green points since the visibility check depends on the resolution of VOG-Map.

gain by some user-defined value corresponding to the type of occupancy status. In practice, we assoicate most gain with unknown occupancy status. This is done for all simulated measurements, and the resulting sum is basically the way-point's information gain. Fig. 4.8 shows an example of simulated sonar measurements used when computing information gain for a sampled way-point.

4.3.2 Unconstrained Sampling

As can be seen in Fig. 4.8, the sampled way-points are very arbitrary. The difference in heading between consecutive way-points can be larger than 90 °, and the robot can virtually move in any direction between adjacent way-waypoints as indicated by the path segments. This works great when using Micro Aerial Vehicle or MAV equipped with camera as in [Bircher et al., 2016]. However,

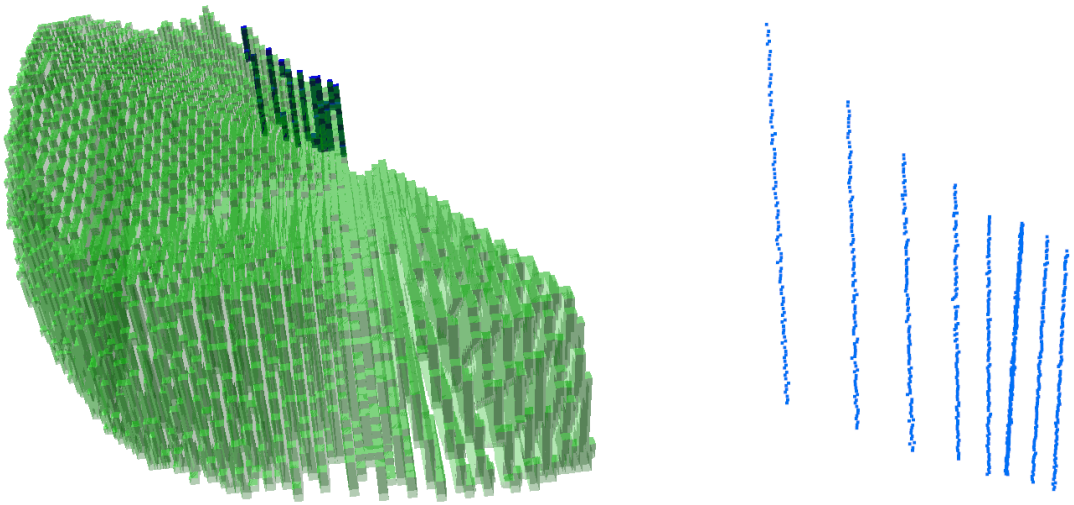

Figure 4.9: Illustration of how unconstrained sampling can result in bad submap for both loop closure detection and path planning. The local occupancy grid map shown on the left contains unknown-space in-between known-space. This is bad because when checking collision, unknown-space is considered in-traversable. Hence, even though a lot space is free as shown in green, the planner will still not be able to return a collision-free path. The point-cloud representation of the local occupancy grid map as shown on the right is very sparse and therefore not suitable for robust loop closure detection.

since our sonar perceives space slice by slice, if we don't constrain the yaw difference between consecutive way-points and allow the robot to move arbitrarily along x-axis, y-axis, and z-axis, the submap we construct would not be very suitable for both loop closure detection and path planning as shown by Fig. 4.9. In addition, since the error in robot state estimate is dominated by the heading error, adding constraint to the change in heading between consecutive way-points can avoid unnecessary drift or accumulated error due to frequent and large heading change. As will be discussed in 4.5.3, adding constraints to how the way-points are sampled can improve the quality of both VOG-Map and scene reconstruction when working with HAUV and using our sensor setup.

4.4 Simulated Experiments in Structured Scene

So far, we have shown show to use VOG-Map as the core component in bringing together smoothing-based pose graph SLAM mapping approach [Teixeira et al., 2016] and sampling-based planner Bircher et al. [2016]. Our overall system is shown again in Fig. 4.10 for convenience. So far, we have only made necessary changes to the mapping and planning algorithms. To be more specific, we built VOG-Map upon the mapping algorithm and added a function to update the base poses of local occupancy grid maps upon loop closure detection. For the planning algorithm, we replaced the global occupancy grid map used with VOG-Map and modified how information gain is computed based on our sensor model.

In this section, we evaluate our system in a 3D simulated structured underwater environment,

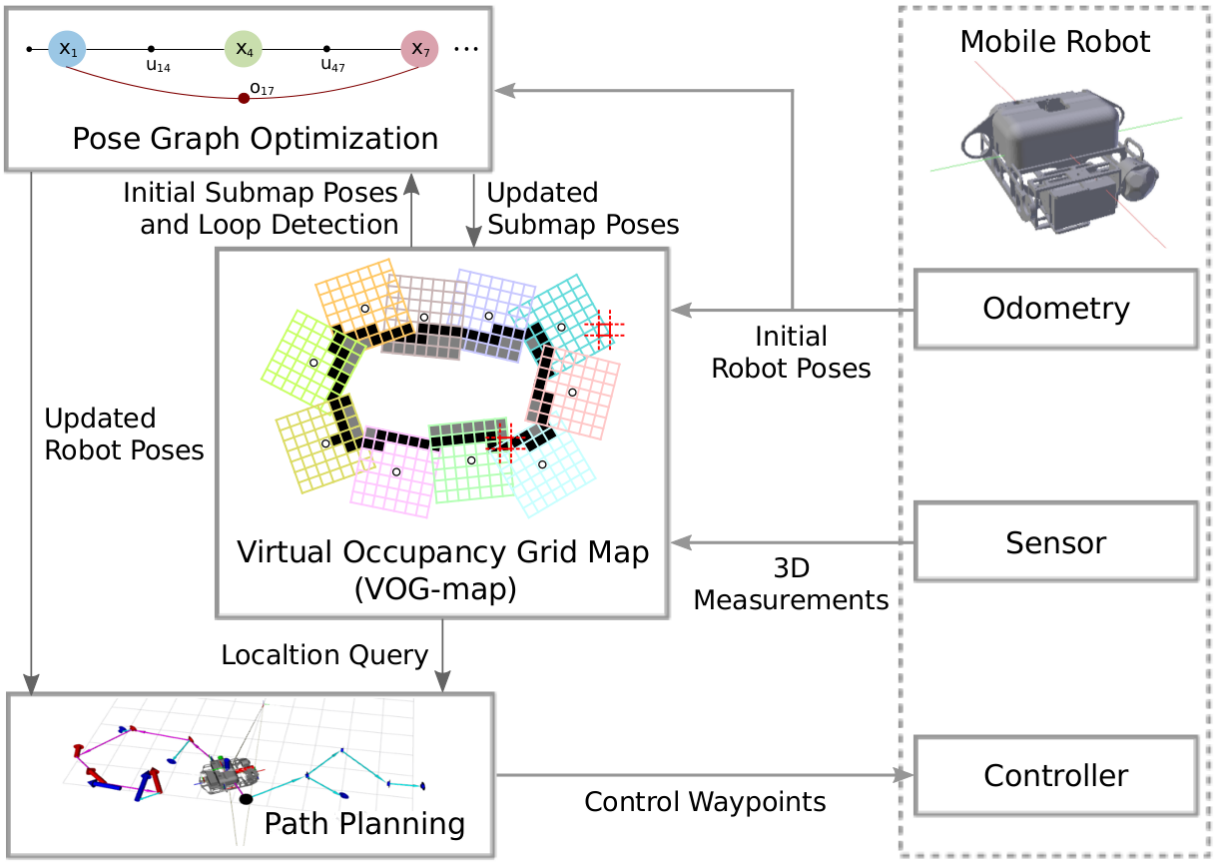

Figure 4.10: Overall system using *virtual* occupancy grid map or VOG-Map for concurrent mapping and planning

being explored by the simulated HAUV. We compare the results of our VOG-Map-based approach against the exploration and mapping approach by Bircher et al. [2016] that uses a single global occupancy map for planning.

4.4.1 Simulation Setup

For performing simulation-based evaluation, we require a closed-loop setup interfacing robot state estimator and low-level control with mapping and planning algorithms. We hence adopt the *UUV Simulator* Manhães et al. [2016], a gazebo-based simulation environment, and customize it based on the model of our underwater vehicle, the Hovering Autonomous Underwater Vehicle (HAUV) from MIT/Bluefin as seen in Fig. 4.1. For sensing, the simulated vehicle is equipped with a profiling sonar sensor that produces real-time 1D scan of its environment. Each sonar scan is composed of 96 beams evenly spaced within the sonar's $\psi = 29^o$ field of view. The vertical field-of-view of the profiling sonar sensor is small with a value of $\theta = 1^o$. However, for simplicity, the profiling sonar sensor is simulated using a laser plugin, and therefore each beam is basically just a ray or line into space. As a result, this vertical field-of-view is not reflected by the simulated

profiling sonar.

For state estimation, in the roll, pitch and z directions, it is possible to obtain absolute measurements at each robot pose using the navigation payload as described in 4.1. We hence simply simulate added gaussian noise for state estimates obtained in these three directions. In the x, y and yaw directions, the sensor measurements from IMU/DVL are not that drift-free and the error grows unbounded over time. In order to simulate such a drifting state estimate in the x, y and yaw directions, we compute corrupted state estimates $\widehat{P}$ from ground truth state estimates P. The corrupted state estimate $\widehat{P}_t$ at time step t is expressed as

$$\widehat{P}_{t+1} = \widehat{P}_t \oplus \left(P_{t+1} \ominus P_t \oplus \Delta t \mathcal{N}(0, \Sigma)\right), \tag{4.1}$$

where $\widehat{P}_1 = P_1$. In Eq. 4.1, each state estimate $\widehat{P}_t$ is computed by corrupting the difference of current and previous ground truth state estimates with additive white Gaussian noise (AWGN) with mean 0 and covariance Σ, and then adding the result to the previous estimated state $\widehat{P}_t$. Σ is the covariance matrix expressing measurement uncertainties and expressed as $\Sigma = diag(\sigma_x^2, \sigma_y^2, \sigma_\psi^2)$. The variance in x-direction is taken as $\sigma_x^2 = 0.00138\ m^2/s$, the variance in y-direction is taken as $\sigma_y^2 = 0.00138\ m^2/s$, and the variance in yaw direction is taken as $\sigma_\psi^2 = 10^{-7}\ rad^2/s$.

4.4.2 Dataset: Ship Propeller of SS Curtiss

The simulated environment consists of the propeller model of *SS Curtiss* as shown in Fig. 4.11. The size of the propeller is approximately 7m $\times$ 4m $\times$ 6m.

The vehicle starts from the starboard side and navigates around the propeller for exploration and mapping within a bounding box of size 12m $\times$ 7m $\times$ 3.5m. Table 4.2 lists the dataset parameters that have been used. When evaluating our approach against the next-best-view planner in Bircher et al. [2016], results are obtained based on multiple runs since the stochasticity in the path planning may lead to selection of different paths in each experimental run. We also run each experiment for roughly the same number of submaps so as to keep the evaluation fair.

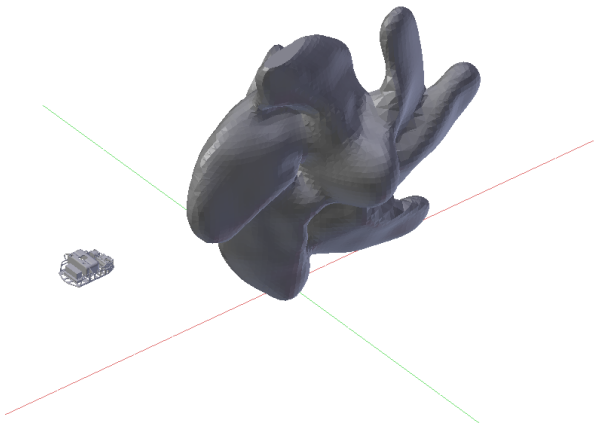

Figure 4.11: Propeller of SS Curtiss and HAUV

Fig. 4.12 shows the obtained 3D reconstructions registered using ICP against the ground truth 3D reconstruction for three different scenarios. Fig. 4.12(a) shows the 3D reconstructions obtained from the baseline next-best-view algorithm Bircher et al. [2016] operating using a standard single global occupancy map, Fig. 4.12(b) shows reconstructions using the VOG-map without loop closures and finally Fig. 4.12(c) shows reconstructions

Table 4.2: Propeller dataset parameters

Parameter	Value	Parameter	Value
Area	10x6x9m	Octomap resolution r	0.095m
v_{max}	0.25m/s	$\dot{\psi}$	0.15rad/s
FoV	[1°,29°]	Scans per submap	100
$d_{max}^{planner}$	5m	d_{max}^{sensor}	11m
λ	0.5	RRT max edge length	2m
N_{max}	15	Collision box	1.0x1.0x0.5m
FPS_{sensor}	5	Maximum submaps	156

using the VOG-map with loop closures. It should be noted that the results in Fig. 4.12(b), Fig. 4.12(c) are from the same experimental run with the loop closure information not being used for mapping for Fig. 4.12(b). Fig. 4.12(a) is from a different experimental run. Table 4.3 shows the root-mean-square error (RMSE) for each of these three 3D reconstructions when registered using ICP against the ground truth point cloud. It can be seen that the 3D reconstruction in Fig. 4.12(c) has the lowest RMSE errors and looks least noisy due to fixing of globally accumulated drift by loop closures. Such globally accumulated drift can be corrected for our system as a result of using the VOG-map representation underneath.

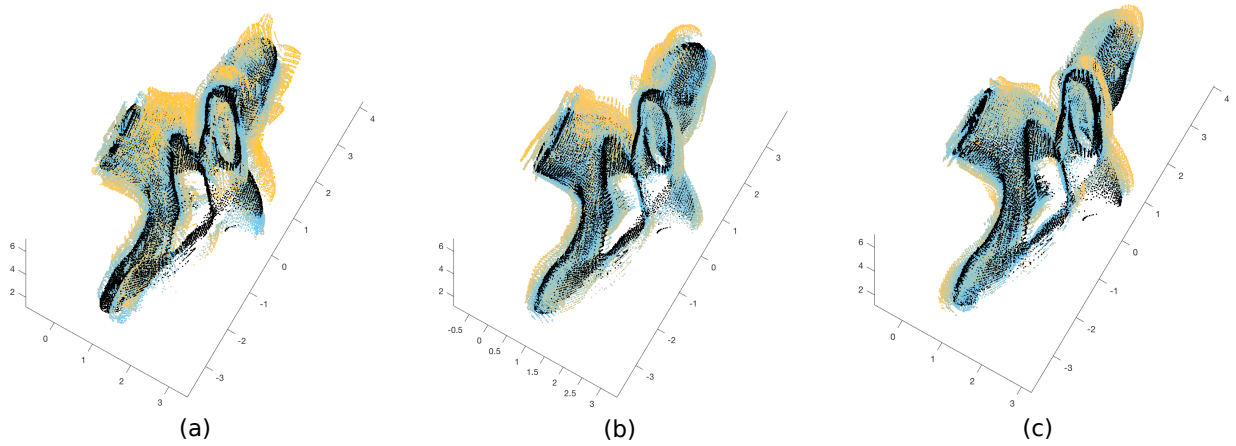

Figure 4.12: 3D Reconstruction results registered using ICP against 3D reconstruction from ground truth odometry for three different scenarios. (a) shows reconstruction from baseline next-best-view algorithm Bircher et al. [2016] operating using a single global occupancy map, (b) shows reconstruction using VOG-map without loop closures and (c) shows reconstruction using VOG-map with loop closures. 3D reconstruction generated using ground truth odometry is in black for each figure. Registered 3D points change color from blue to yellow as distance from ground truth point cloud increases. Scale of the plots is in meters.

Table 4.3: RMSE Errors in 3D Reconstructions

Run No.	Standard Global Map	Virtual Global Map without loop closure	**Virtual Global Map with loop closures**
1	0.19792 m	0.14902 m	**0.13345 m**
2	0.20491 m	0.14176 m	**0.13357 m**

4.5 Improved Mapping and Planning

While the result from the simulated experiments is very promising, we observe several problems that can cause our VOG-Map-based system to perform even more poorly at times as compared to the base-line planning algorithm. These problems can be broadly classified into **(1)** incorrect loop closure and **(2)** slow planning speed. In this section, we present the improvements we made on the mapping and planning algorithm as we try to address these problems

4.5.1 Degeneracy-aware ICP

As was discussed in 4.2.2, wrong ICP registration or wrong loop closure can happen even with informative initial relative transformation from odometry. While there are criteria that we can use to reject the relative transformation obtained from ICP heuristically, it is better if we can tackle the problem at the source, i.e. detect the degenerate direction in the state space of underlying optimization problem. Zhang, et al. in Zhang et al. [2016] prove this can be done by performing principal component analysis, and present a technique called solution remapping that updates the optimization problem along only well-conditioned directions. Algorithm 7 shows the pseudo-code of our degeneracy-aware ICP method based on the solution remapping technique presented in Zhang et al. [2016].

Algorithm 7 Degeneracy-aware ICP

Input: initial relative transformation from odometry T_{odom}
Output: relative transformation from ICP T_{icp}
$T_{icp} \leftarrow T_{odom}$
while nonlinear iterations **do**
    Linearize the optimization problem at T_{icp} to get A^TA and A^Tb
    Compute eigenvalue λ_i and eigenvector v_i of A^TA for $i = 1 \dots 6$
    Determine an eigenvalue threshold λ_{min}
    Construct matrix V_f containing all the eigenvectors
    Construct matrix V_u containing only well-conditioned directions based on λ_{min}
    $\Delta x_u \leftarrow (A^TA)^{-1}A^Tb$
    $T_{icp} \leftarrow T_{icp} + V_f^{-1}V_u\Delta x_u$
return T_{icp}

We implement degeneracy-aware ICP by extending PCL's [Rusu and Cousins, 2011] Point-to-Plane ICP method. The derivation and construction of A^TA and A^Tb based on Point-to-Plane error metric can be found in [Low, 2004]. The key difference of degeneracy-aware ICP from solution remapping technique presented in [Zhang et al., 2016] is that we re-compute the well-conditioned directions based on a different covariance matrix (A^TA) at each non-linear iteration. Theoretically,

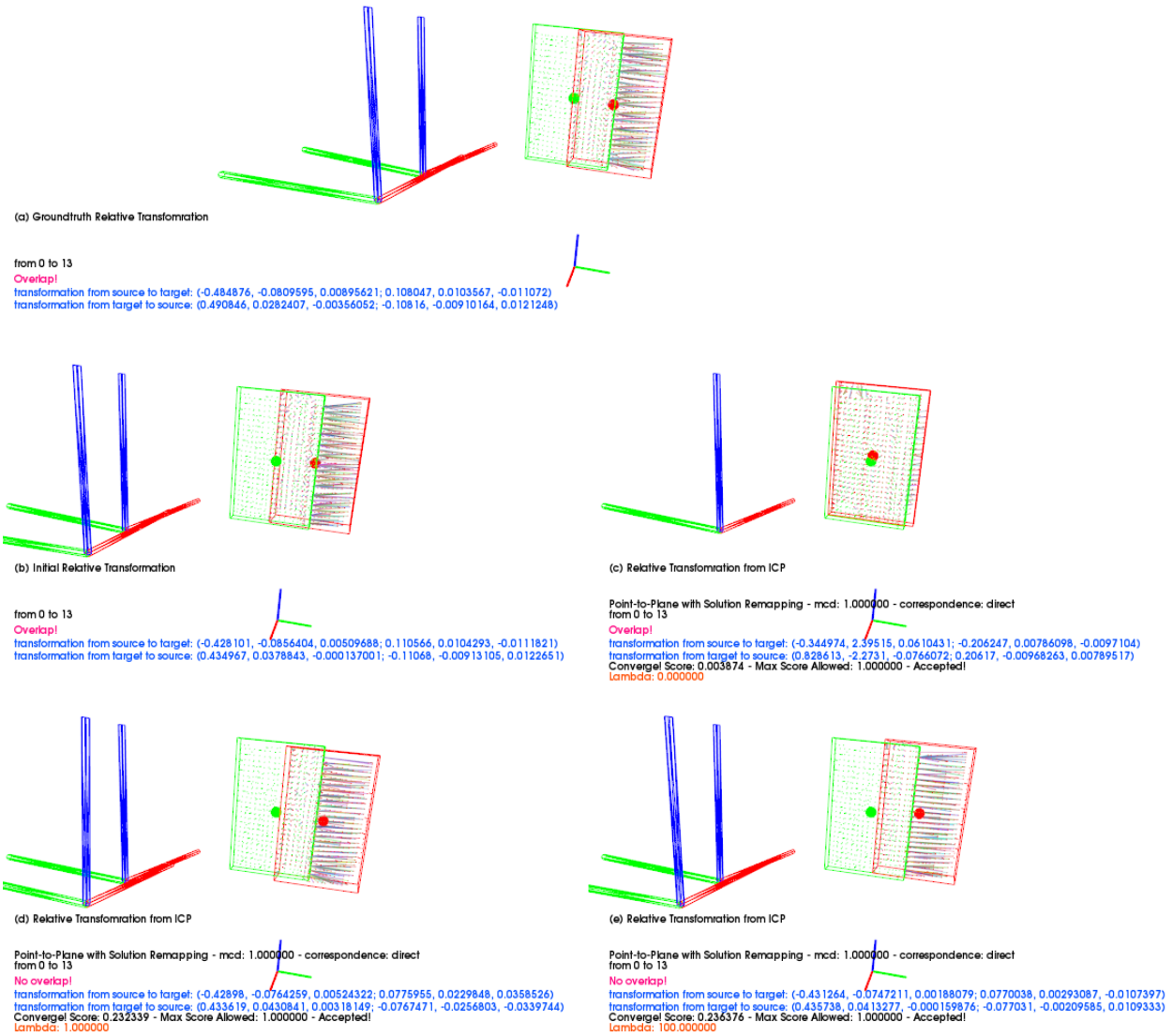

Figure 4.13: Illutration of relative transformation obtained from degenerate-aware ICP. Since these submaps are obtained in simulation, the groundtruth relative transformation between source point-cloud (red) and target point-cloud (green) is available and is used to visualize the two point-clouds in target-cloud coordinate frame as shown in **(a)**. **(b)** shows the result of using initial relative transformation from odometry. **(c)** shows the result of using relative transformation from degeneate-aware ICP with $\lambda_{min} = 0$, which essentially becomes the original Point-to-Plane ICP. **(d)** and **(e)** show the results of using relative transformation from degeneate-aware ICP with $\lambda_{min} > 0$. While the results are way better than that shown in Fig. 4.7 based on 2D-Point-to-Point-ICP, the relative tranformation obtained from degenerate-aware ICP can still be worse than odometry transformation as compared to the groundtruth transformation.

this re-computation captures the fact that at each iteration when we linearize at an updated T_{icp}, the degenerate directions might change. It is also important to note that how to determine the λ_{min} for separating well-conditioned directions from degenerate directions is still an open question. For now, we choose λ_{min} heuristically and for each non-linear iteration, we use the same λ_{min}.

Fig. 4.13 shows the result obtained using degenerate-aware ICP. We use the same point-clouds as those used in Fig. 4.7 for comparison. It is obvious that degenerate-aware ICP coverges to a better solution as seen in Fig. 4.13 **(d)** and **(e)**, but the relative transformation obtained can still be worse than the relative transformation from odometry. It is especially obvious along the horizontal axis where the two point-clouds can slide freely as a result of using the Point-to-Plane error metric. If we simply construct a relative pose constraint from the resulting relative transformation without considering the degenerate directions along horizontal axis, and add the relative pose constraint to the pose graph SLAM framework, the SLAM optimization will converge to erroneous pose estimates. We therefore want to construct a relative pose constraint that only constrains well-conditioned directions. The relative pose constraint that only constrains well-conditioned direction is also called custom factor, as will be detailed in next section.

4.5.2 Custom Factor

As mentioned above, custom factor provides constraints only along well-conditioned directions. Algorithm 8 details the construction of custom factor.

Algorithm 8 Custom Factor

Input: relative transformation from ICP as 6 DOF vector X_{icp} **and**
    6×6 measurement covariance matrix Cov
Output: custom factor F
Linearize the ICP optimization problem at X_{icp} to get A^TA and A^Tb
Compute eigenvalue λ_i and eigenvector v_i of A^TA for $i = 1 \dots 6$
Determine an eigenvalue threshold λ_{min}
Construct matrix V_u containing only well-conditioned directions based on λ_{min}
$F.measurement \leftarrow V_u X_{icp}$
$F.covariance \leftarrow V_u Cov V_u^T$
return F

We first use the similiar approach as in Algorithm 7 to compute A^TA and A^Tb. Then, measurement or relative transformation from degenerate-aware ICP is projected onto only well-conditioned directions. Finally, the heuristically chosen measurement covariance Cov is transformed based on the well-conditioned directions as well.

Once the custom factor is constructed and added to the pose graph SLAM framework, we can perform *maximum a posteriori* (MAP) inference on resulting pose graph so as to determine value of unknown base poses $\mathbf{x}_j$ that maximally agree with the information present in uncertain measurements. Performing MAP inference for SLAM problems with Gaussian noise models is equivalent

to solving a nonlinear least-squares problem Dellaert and Kaess [2017]. For doing MAP estimation over sequence of poses $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N\}$, the associated nonlinear least-squares problem is written as

$$\mathcal{X}^{MAP} = \underset{\mathcal{X}}{\operatorname{argmin}} \left(\sum_{i=1}^{N} || f(\mathbf{x}_{i-1}, u_{i-1}) - \mathbf{x}_i ||^2_{\Lambda_i} + \sum_{(i,k)\in O'} ||\mathbf{V}_u h(\mathbf{x}_i, \mathbf{x}_k) - \mathbf{V}_u o_{ik}||^2_{\mathbf{V}_u \Gamma_{ik} \mathbf{V}_u^T} \right), \tag{4.2}$$

where $f(\mathbf{x}_{i-1}, u_{i-1})$, $h(\mathbf{x}_i, \mathbf{x}_k)$ are the motion and sensor models respectively subject to additive white gaussian noise. Λ_i is the covariance matrix associated with motion model noise, and O' is the set of all tuples (i, k) for which we consider partial pairwise registration constraints (in non-degenerate directions). $\mathbf{V}_u$ is the matrix projecting the relative transformation from degenerate-aware ICP to only the well-conditioned directions. We now solve this least squares optimization for sequence of unknown poses $\mathcal{X}$ using the iSAM optimization library [Kaess et al., 2007].

The motion and sensor models in Eq. 4.2 can be linearized using Taylor expansion as shown in Dellaert and Kaess [2017], and Eq. 4.2 becomes

$$\begin{aligned} \Delta^* &= \underset{\Delta}{\operatorname{argmin}} \left(\sum_{i=1}^{N} || f(\mathbf{x}_i^0) + F_i \Delta_i - \mathbf{x}_i ||^2_{\Lambda_i} + \sum_{(i,k)\in O'} ||\mathbf{V}_u h(\mathbf{x}_i^0, \mathbf{x}_k^0) + \mathbf{V}_u H_{(i,k)} \Delta_{(i,k)} - \mathbf{V}_u o_{ik}||^2_{\mathbf{V}_u \Gamma_{ik} \mathbf{V}_u^T} \right) \\ &= \underset{\Delta}{\operatorname{argmin}} \left(\sum_{i=1}^{N} || F_i \Delta_i - \{\mathbf{x}_i - f(\mathbf{x}_i^0)\} ||^2_{\Lambda_i} + \sum_{(i,k)\in O'} ||\mathbf{V}_u H_{(i,k)} \Delta_{(i,k)} - \mathbf{V}_u \{o_{ik} - h(\mathbf{x}_i^0, \mathbf{x}_k^0)\}||^2_{\mathbf{V}_u \Gamma_{ik} \mathbf{V}_u^T} \right), \end{aligned} \tag{4.3}$$

where F_i and $H_{(i,k)}$ are the measurement Jacobians for motion model and sensor model respectively. Δ_* is the state update vector. $\mathbf{x}_i - f(\mathbf{x}_i^0)$ and $o_{ik} - h(\mathbf{x}_i^0, \mathbf{x}_k^0)$ are the prediction errors, i.e. the difference between actual and predicted measurement. Δ^* denotes the solution to the locally linearized problem.

We can eliminate the covariances Λ_i and $\mathbf{V}_u \Gamma_{ik} \mathbf{V}_u^T$ in Eq. 4.3 by pre-multiplying the Jacobians and prediction errors with $\Lambda_i^{-\frac{1}{2}}$ and $\{\mathbf{V}_u \Gamma_{ik} \mathbf{V}_u^T\}^{-\frac{1}{2}}$ respectively.

$$\begin{aligned} A_i &= \Lambda_i^{-\frac{1}{2}} F_i & A_{i,k} &= (\mathbf{V}_u \Gamma_{ik} \mathbf{V}_u^T)^{-\frac{1}{2}} H_{(i,k)} \\ b_i &= \Lambda_i^{-\frac{1}{2}} (\mathbf{x}_i - f(\mathbf{x}_i^0)) & b_{i,k} &= (\mathbf{V}_u \Gamma_{ik} \mathbf{V}_u^T)^{-\frac{1}{2}} \mathbf{V}_u \{o_{ik} - h(\mathbf{x}_i^0, \mathbf{x}_k^0)\} \end{aligned}$$

This whitening process eliminates the units of measurements. Therefore, custom factor that constrains along well-conditioned directions with mixed units can be used with other factors that

provide constraints in the original bases, i.e. x, y, z, yaw, pitch, roll. For completeness, Eq. 4.3 becomes a standard least-squares problem as shown by Eq. 4.4.

$$\Delta^* = \underset{\Delta}{\operatorname{argmin}} \left(\sum_{i=1}^{N} || A_i \Delta_i - b_i ||_2^2 + \sum_{(i,k) \in O'} || A_{i,k} \Delta_{(i,k)} - b_{i,k} ||_2^2 \right), \tag{4.4}$$

4.5.3 Constrained Sampling

As was discussed in 4.3.2, sampled way-points have a large impact on the submaps we obtained and the amount of drift. To address these issues, we add constraints during our sampling process such that the heading difference between consecutive way-points is bounded by some threshold. In addition, we separate the sampling process into two steps. The first step generates way-points on the x-y plane with a fixed z or depth. The second step generates way-points along the z-axis with fixed x and y coordinate values. We use the current robot position for the fixed values during the sampling process. Fig. 4.14 illustrates the resulting sampled paths. Fig. 4.15 shows the typical submap we obtained using the constrained sampling strategy.

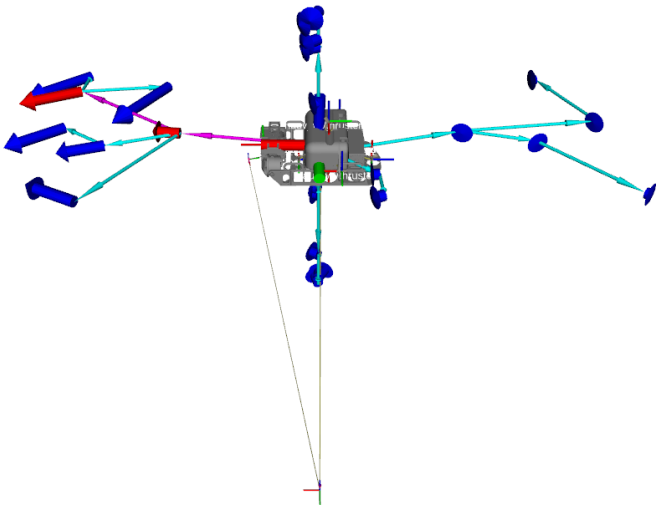

Figure 4.14: Illustration of sampled paths generated using the constrained sampling strategy after one planning iteration. The heading difference threshold used is 90°

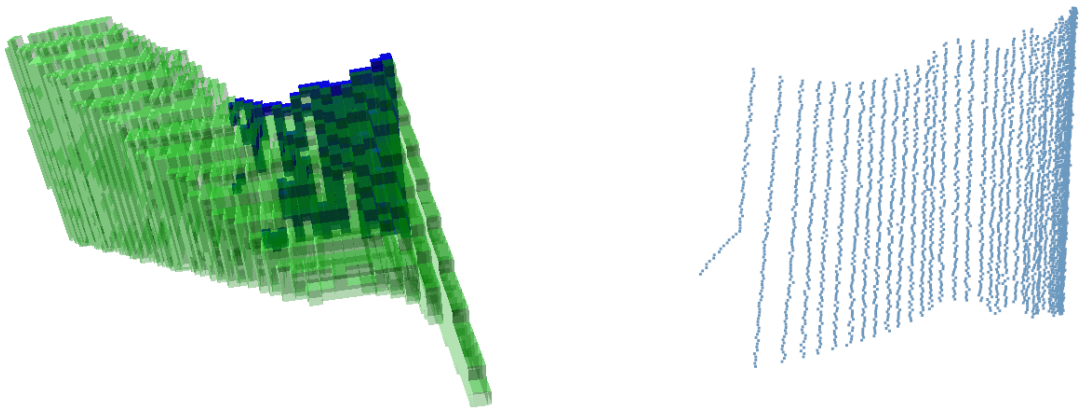

Figure 4.15: Illustration of typical submap obtained using constrained sampling strategy. The local occupancy grid map shown on the left is more compact, and the point-cloud shown on the right is more dense as compared to those in Fig. 4.9.

4.5.4 Gain Computation

After way-points are sampled and paths generated, the way-point robot chooses to move to depends on the information gain associated with the way-point. In 4.3.2, we compute the gain for each way-point only at the way-point's location using our sensor model. However, due to our sensor model, computing gain this way won't be too informative. A better strategy is to compute the gain along path segment that connects consecutive way-points as illustrated by Fig. 4.16.

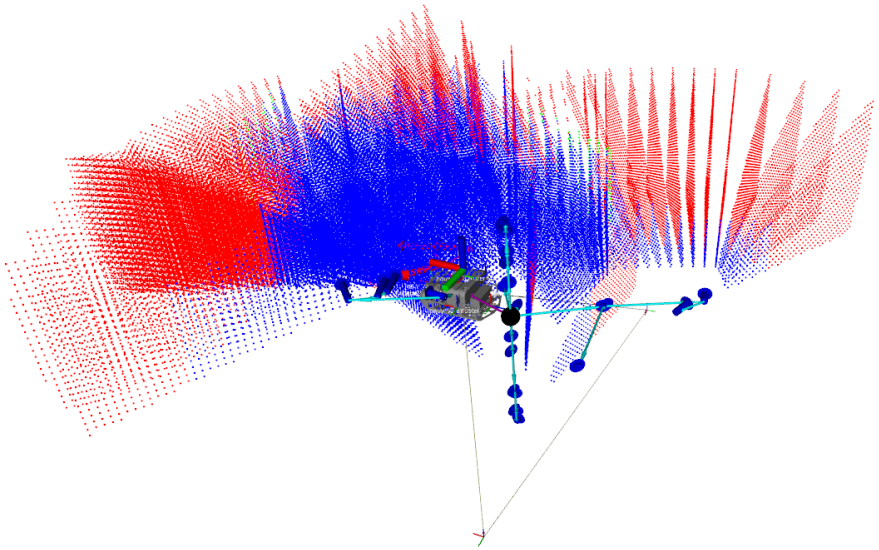

Figure 4.16: Illustration of the new gain computation strategy. We first interpolate consecutive way-points based on the resolution used to build local occupancy grid maps. Then, we compute the gain for each intermediate way-points obtained via interpolation. Finally, the information gain for the destination way-point is set to be the sum of these intermediate gains.

4.5.5 Cached VOG-Map

As was discussed in 3.5.1, the planning time for each planning iteration quickly exceeds beyond just a few seconds even when using VOG-Map line query operation 3.5.1, as shown by Fig. 3.4. The reason is that planner made too many queries to the VOG-Map each planning iteration as illustrated in Fig. 4.17. This slow planning time can be problematic because the robot will stay still for more than a few seconds each time the planning algorithm re-plans, which happens very frequently for exploration task.

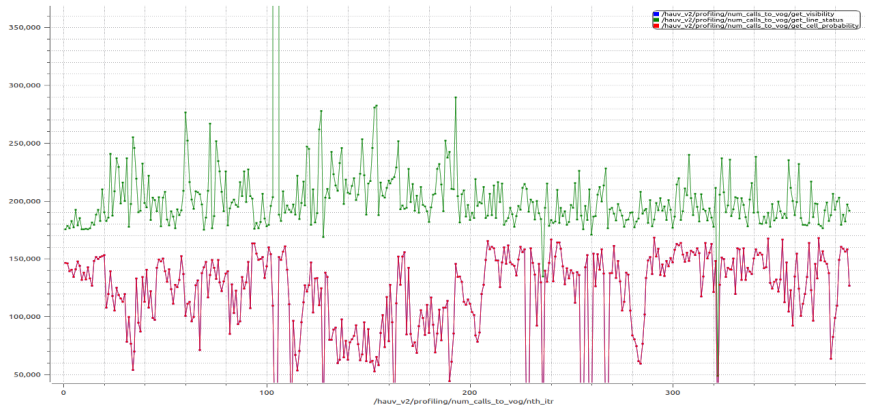

Figure 4.17: The number on x-axis shows the current planning iteration number (1st itr, 2nd itr, etc.). The number on y-axis shows the total number of calls to VOG-Map made by each planner function call. The magenta line is due to the overlap of red and blue lines. The red line corresponds to the number of times get_cell_probability is called. This function is called when computing gain for each way-point. The blue line corresponds to the number of times get_visibility is called. This function is called when computing gain for each way-point. The green line corresponds to the number of time get_line_status is called. This function is called when checking collision between consecutive way-points. get_cell_probability corresponds to sequential VOG-Map query operation 3.4.2 while get_visibility and get_line_status correspond to VOG-Map line query operation 3.5.1.

A potential fix to this problem is to use merged VOG-Map occupancy query operation as discussed in 3.4.2. However, it won't be much better if we have to merge all local occupancy grid maps from scratch each time new local occupancy grid map is available. The key insight is that we don't have to merge all local occupancy grid maps from scratch each time new local occupancy grid map is available. This is because when new odometry constraint and new pose node is added to the pose graph SLAM framework, the SLAM optimized pose estimates for all the previous pose nodes won't be changed. Therefore, most of the time, we only need to merge the new local occupancy grid map into the cached VOG-Map. This incremental update to the cached VOG-Map can be done efficiently as shown by Fig. 4.18. The spikes in the figure indicate that we occasionally have to merge all local occupancy grid maps from scratch when new relative pose constraint from ICP is added.

By using merged VOG-Map occupancy query operation and more efficient merging strategy, the planning time taken for each planning iteration is consistently under one second. This is possible because the operation to merge all the local occupancy grid maps from scratch runs on a different

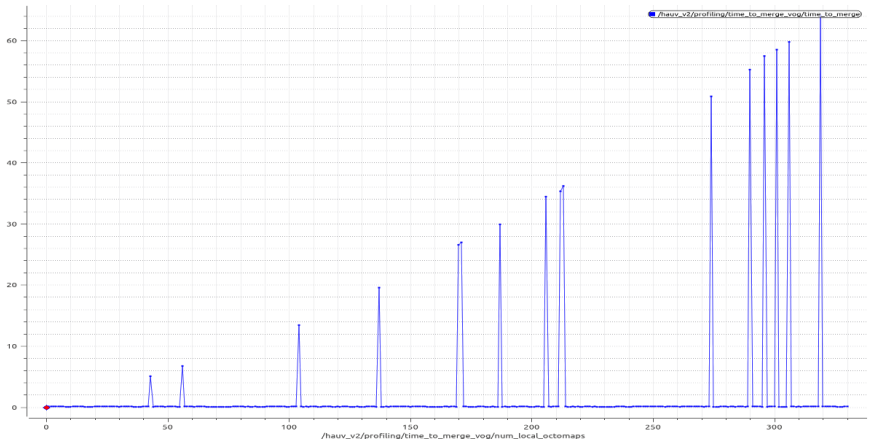

Figure 4.18: Illustration of our merging strategy. The number on x-axis is the total number of local occupancy grid maps in the VOG-Map. The number on y-axis is the time taken to merge in seconds. As can be seen, most of time, we update the cached VOG-Map incrementally with the new local occupancy grid map. We only have to re-merge from scratch upon loop closure.

thread from the thread that the planner runs on.

4.6 Real-world Experiments

We conduct real-world experiments with the HAUV shown in Fig. 4.1 and compare the results of our VOG-Map-based approach against the approach based on a single global occupancy grid map as presented in Bircher et al. [2016].

4.6.1 Real-world Experiment Setup

The HAUV used in our real-world experiments is equipped with the same navigation payload and sonar sensor as described in 4.1. We design a scenario as shown in Fig. 4.19 in which the HAUV starts in a position in a tank where it can only see a limited portion of its environment. It has

Table 4.4: Real-world experiment parameters

Parameter	**Value**	**Parameter**	**Value**
Area	12x12x2.5m	Octomap resolution r	0.095m
FoV	[1°,29°]	Scans per submap	100
$d_{max}^{planner}$	5.63m	d_{max}^{sensor}	5.63m
λ	0	RRT max edge length	1m
N_{max}	15	Collision box	1.0x1.0x0.5m
FPS_{sensor}	10	Maximum submaps	105

to autonomously map the tank based on VOG-Map to create a reconstruction of the scene that includes the walls of the cylindrical tank, the rectangular aluminum box, and the ladder. Table 4.4 lists the experiment parameters that have been used. In particular, the size of the tank is 7m $\times$ 7m $\times$ 3m while the bounding box we use for exploration is much larger since the HAUV's starting position is not necessarily at the center of the tank.

4.6.2 Real-world Experiment Results

The navigation payload of the HAUV is quite accurate in this rather small contained environment. To more readily see the effect of drift, we artificially corrupt values obtained from the state estimates given by the IMU/DVL in the same way in Eq. 4.1. Since there is no ground truth model, we qualitatively compare the resulting occupancy maps and the reconstructed models. To keep the comparison fair, the HAUV while operating based on the VOG-map, also simultaneously constructs a standard global occupancy grid using raw odometry without loop closures.

Fig. 4.20 shows the standard global occupancy map and the VOG-map reconstructed from local submaps. We reconstruct the VOG-map from local submaps only for a visual comparison. Fig. 4.21 shows 3D scene reconstructions generated without and with loop closures. It can be seen that, as a result of loop closures, the reconstruction in Fig. 4.21(b) has corrected drift. Since the tank environment is small, the drift accumulation is not as significant as for the simulated propeller dataset. Such globally accumulated drift can be corrected for our system as a result of using the VOG-map representation underneath.

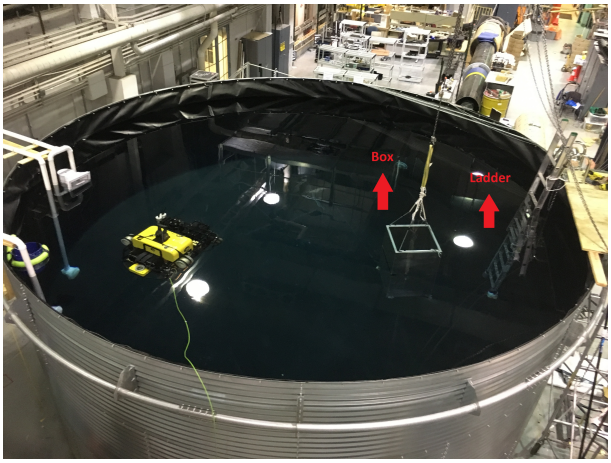

Figure 4.19: HAUV in the tank with an aluminum box and a ladder

The effect of improvement in scene reconstruction quality is more easily seen than the effect of VOG-map on planning. Since this is a small environment, it is difficult to conclude that the planner based on VOG-map returns better overall plan in terms of collision avoidance and information gathering. However, based on the occupancy grid maps in Fig. 4.20, it is expected that a planner operating using the VOG-map would be able to generate waypoints that account for drift better than a planner using a standard global occupancy map.

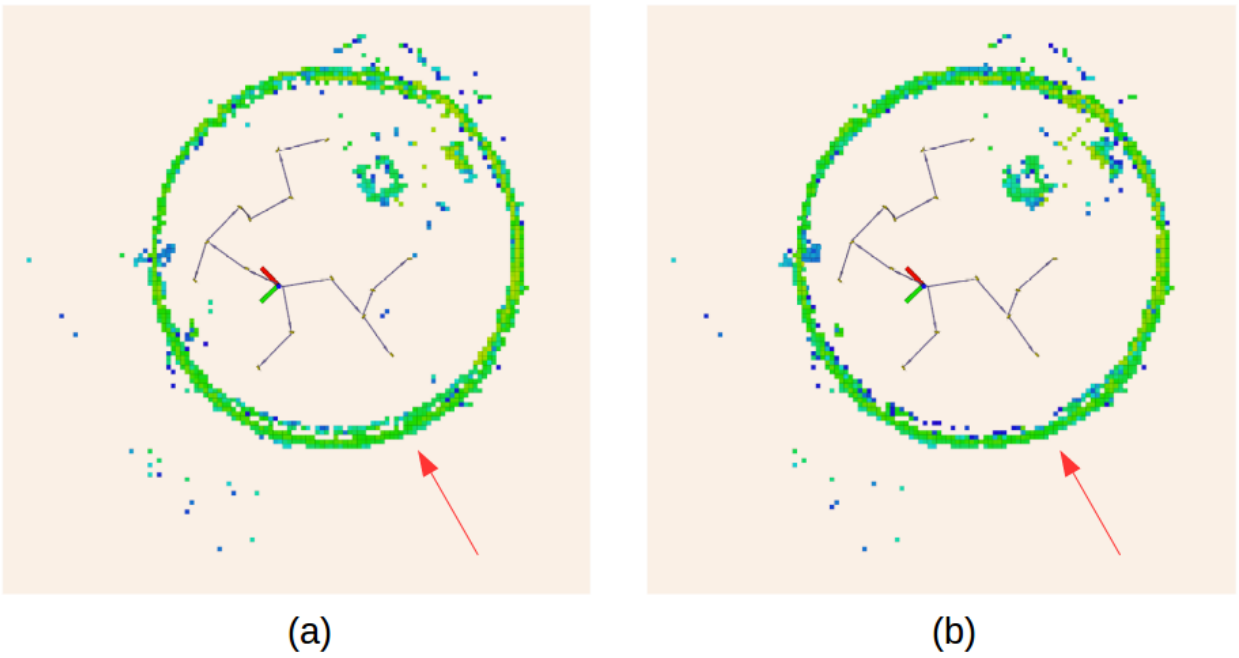

Figure 4.20: **(a)** Top view of standard global occupancy grid. **(b)** Top view of VOG-map, reconstructed from local submaps only for comparison. The yellow background shows the extent of the exploration bounding box. Regions indicated with red arrows show the areas where loop closures using the VOG-map are able to correct accumulated drift.

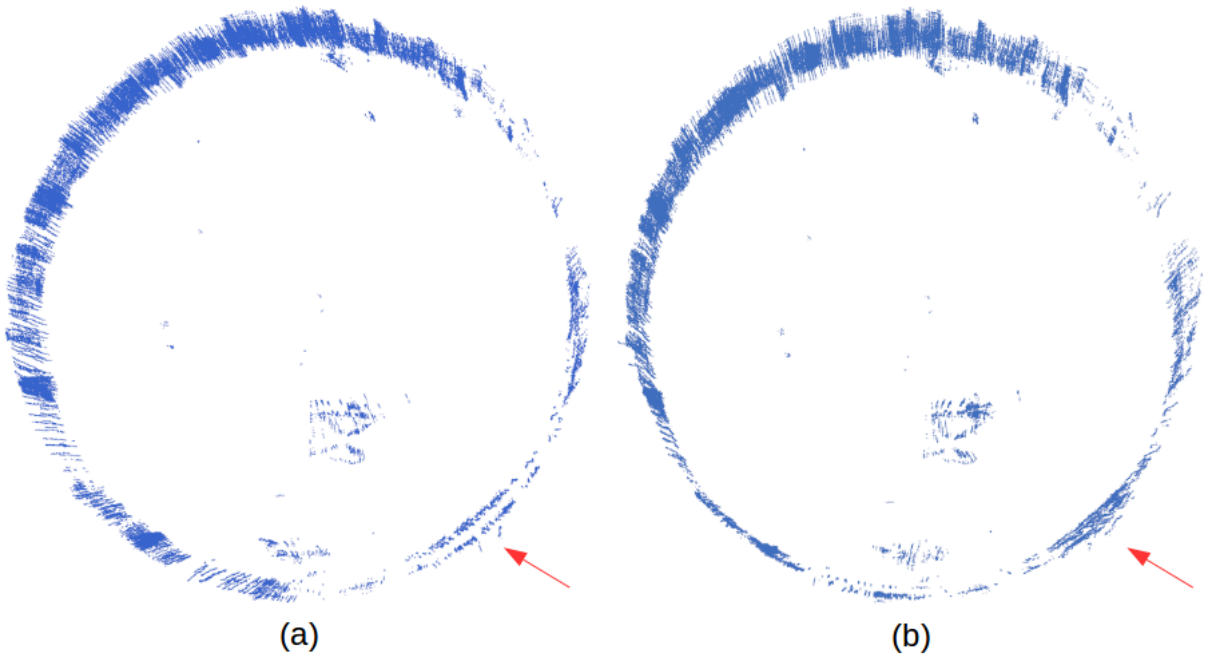

Figure 4.21: 3D scene reconstructions: **(a)** Without loop closures. **(b)** With loop closures. Regions indicated with red arrows show the areas where loop closures using the VOG-map are able to correct accumulated drift. Since the tank environment is small, the drift accumulation is not as significant as for the simulated propeller dataset.

Chapter 5

Conclusions

5.1 Contributions

This thesis presents the following contributions towards autonomous reconstruction task in underwater unstructed scenes,

- It presents a general space representation called *virtual* occupancy grid map or VOG-Map that can work with any smoothing-based pose graph SLAM approaches and any planning approaches that are based on occupancy grid map.

- It shows VOG-Map is a better space representation than the standard occupancy grid map especially when robot state estimates are uncertain since VOG-Map can be deformed efficiently to correct globally accumulated drift via loop closures while maintaining free space information for the purpose of path planning.

- It presents degenerate-aware ICP which is shown to be more robust than normal ICP method especially in degenerate scenes that lack geometric details. In addition, it shows how to construct a custom factor that constrains arbitrary directions, and proves the correctness of using custom factor in the SLAM optimization problem.

- It presents strategies that can be used to sample way-points that work better with our platform and sensor used. It also shows a different way that information gain associated with each way-point can be computed.

- It shows how VOG-Map can be a unifying space representation that combines state-of-the-art mapping and planning approaches by implementing a full-fledged autonomous system. The system is evaluated in both simulation and real-world experiments in terms of reconstruction quality.

5.2 Observations and Future Work

As future work, we would like to look into the following problems which can be broadly categorized as **(1)** correspondences for ICP and **(2)** planning time based on VOG-Map.

Currently, even with degeneracy-aware ICP and custom factor, our ICP method is still prone to error, and wrong relative pose constraint could still be added to the pose graph SLAM framework, resulting in worse VOG-Map and scene reconstruction than those based on odometry. The obvious reason is that we use the same heuristically chosen λ_{min} at each non-linear iteration of the ICP algorithm. The same λ_{min} is also used for constructing custom factor. The better way would be to determine λ_{min} in a systematic way based on statistics. And, instead of using a fixed λ_{min} at each non-linear iteration, we could use a dynamically chosen λ_{min} at each non-linear iteration, as well as for the construction of custom factor. However, λ_{min} is not the main reason. The main cause of our problem is wrong data association. The performance of ICP algorithms depends on correct correspondences between points. Currently, we simply use the nearest point heuristics for the computation of correspondences. But, often times, the correspondences are not correct, and the best-case scenario when using incorrect correspondences is that the ICP algorithms converge to local minima. We would like to examine more advanced techniques for determining correct correspondences and for removing correspondences that are likely to be wrong. We hope to make our degenerate-aware ICP more robust by using dynamically chosen λ_{min} and better correspondences.

As shown by Fig. 4.18, merging all the local occupancy grid maps can take more than a minute when the number of local occupancy grid maps approaches 300. Although we only need to merge from scratch occasionally upon loop closure, and merging happens on a separate thread, merging from scratch is not an ideal solution. The first reason is some mobile platform might not have the storage space to cache the additional merged VOG-Map. The second reason is that during the time to merge from scratch, planner would have to plan based an outdated VOG-Map. This means increased chance for the robot to run into obstacles. One strategy that we can use to avoid keeping a cached VOG-Map is to bound the number of local occupancy grid maps in the VOG-Map below some number so that planner can use VOG-Map line query operation to plan efficiently. This can be done by selectively merging some local occupancy grid maps based on their corresponding base pose uncertainty. However, the resulting VOG-Map would become just an approximation to the original VOG-Map. This might not be ideal depending on application and tolerance. If we have to cache a merged VOG-Map, we probably don't have to merge from scratch upon loop closure. This is because even when loop closure is detected, the resulting relative pose constraint most likely only affects pose nodes in a local region. Therefore, we can speed up the merging time by only re-merge from scratch only those local occupancy grid maps whose base poses are changed.

Bibliography

Edward Belcher, William Hanot, and Joe Burch. Dual-frequency identification sonar (didson). In *Underwater Technology, 2002. Proceedings of the 2002 International Symposium on*, pages 187–192. IEEE, 2002. 4.1

Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1462–1468. IEEE, 2016. 1.4, 1.2, 1.2, 2.2, 2.3, 4.3, 4.3.1, 4.3.2, 4.4, 4.4.2, 4.12, 4.6

Michael Bosse, Paul Newman, John Leonard, and Seth Teller. Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. *The International Journal of Robotics Research*, 23(12):1113–1139, 2004. 2.1.1

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. 2.3

Frank Dellaert and Michael Kaess. Factor graphs for robot perception. *Foundations and Trends in Robotics*, 6(1-2):1–139, 2017. 4.5.2, 4.5.2

Carlos Estrada, José Neira, and Juan D Tardós. Hierarchical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005. 2.1.1

Nathaniel Fairfield and David Wettergreen. Active slam and loop prediction with the segmented map using simplified models. In *Field and Service Robotics*, pages 173–182. Springer, 2010. 2.1.1, 2.1.3, 2.1.4

Nathaniel Fairfield, George Kantor, and David Wettergreen. Real-time slam with octree evidence grids for exploration in underwater tunnels. *Journal of Field Robotics*, 24(1-2):03–21, 2007. 2.1.4

Nathaniel Fairfield, David Wettergreen, and George Kantor. Segmented slam in three-dimensional environments. *Journal of Field Robotics*, 27(1):85–103, 2010. 2.1.1, 2.1.3

Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013. 1.2, 3.2, 3.2, 3.3, 3.4.2, 3.4.2, 3.4.2, 4.3.1

Franz S Hover, Ryan M Eustice, Ayoung Kim, Brendan Englot, Hordur Johannsson, Michael Kaess, and John J Leonard. Advanced perception, navigation and planning for autonomous in-

water ship hull inspection. *The International Journal of Robotics Research*, 31(12):1445–1464, 2012. 1.1, 1.2, 4.1

Ming Hsiao, Eric Westman, Guofeng Zhang, and Michael Kaess. Keyframe-based dense planar slam. In *IEEE International Conference on Robotics and Automation, ICRA, Singapore*, 2017. 2.1.2, 2.3

Margaret E Jefferies, Michael C Cosgrove, Jesse T Baker, and Wai-Kiang Yeap. The correspondence problem in topological metric mapping–using absolute metric maps to close cycles. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 232–239. Springer, 2004. 2.1.1

Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Fast incremental smoothing and mapping with efficient data association. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1670–1677. IEEE, 2007. 1.2, 3, 3.3, 4.2, 4.5.2

Kurt Konolige, Eitan Marder-Eppstein, and Bhaskara Marthi. Navigation in hybrid metric-topological maps. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3041–3047. IEEE, 2011. 2.1.2, 2.1.3, 2.1.4, 2.2, 2.3

Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998. 1.2, 2.2, 2.3, 4.3.1

Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *ICAPS*, pages 262–271, 2005. 2.2, 2.3

Brad Lisien, Deryck Morales, David Silver, George Kantor, Ioannis M Rekleitis, and Howie Choset. Hierarchical simultaneous localization and mapping. In *IROS*, pages 448–453, 2003. 2.1.1

Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4:1, 2004. 4.5.1

Musa Morena Marcusso Manhães, Sebastian A Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. Uuv simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–8. IEEE, 2016. 4.4.1

Hans Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 116–121. IEEE, 1985. 3.2

Kai Ni, Drew Steedly, and Frank Dellaert. Tectonic sam: Exact, out-of-core, submap-based slam. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1678–1685. IEEE, 2007. 2.1.2, 2.3

Christos Papachristos, Shehryar Khattak, and Kostas Alexis. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4568–4575. IEEE, 2017. 2.2, 2.3

Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *Robotics and*

automation (ICRA), 2011 IEEE International Conference on, pages 1–4. IEEE, 2011. 4.2.1, 4.5.1

Alan C Schultz, William Adams, and JJ Grefenstette. Continuous localization using evidence grids. In *ICRA*, pages 2833–2839, 1998. 2.1.1

Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. *no*, 3:1–5, 2017. 4.2.1, 4.2.1

Pedro V Teixeira, Michael Kaess, Franz S Hover, and John J Leonard. Underwater inspection using sonar-based volumetric submaps. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4288–4295. IEEE, 2016. 1.1, 1.2, 1.2, 1.3, 1.2, 2.1.2, 2.3, 3, 3.1, 4.1, 4.1, 4.2, 4.4, 4.2, 4.2.1, 4.2.1, 4.2.1, 4.2.2, 4.4

Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous robots*, 15(2):111–127, 2003. 2.1.4

Mark VanMiddlesworth, Michael Kaess, Franz Hover, and John J Leonard. Mapping 3d underwater environments with smoothed submaps. In *Field and Service Robotics*, pages 17–30. Springer, 2015. 2.1.2, 2.3

Eduard Vidal, Juan David Hernández, Klemen Istenic, and Marc Carreras. Online view planning for inspecting unexplored underwater structures. *IEEE Robotics and Automation Letters*, 2(3): 1436–1443, 2017. 2.2, 2.3

René Wagner, Udo Frese, and Berthold Bäuml. Graph slam with signed distance function maps on a humanoid robot. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2691–2698. IEEE, 2014. 2.1.2, 2.3

Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE, 1997. 2.2

Brian Yamauchi and Pat Langley. Place learning in dynamic real-world environments. *Proceedings of RoboLearn*, 96:123–129, 1996. 2.1.1

Ji Zhang, Michael Kaess, and Sanjiv Singh. On degeneracy of optimization-based state estimation problems. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 809–816. IEEE, 2016. 4.2.2, 4.2.2, 4.5.1, 4.5.1