Online and Consistent Occupancy Grid Mapping for Planning in Unknown Environments

Paloma Sodhi[1], Bing-Jui Ho[2], and Michael Kaess[1]

***Abstract*— Actively exploring and mapping an unknown environment requires integration of both simultaneous localization and mapping (SLAM) and path planning methods. Path planning relies on a map that contains free and occupied space information and is efficient to query, while the role of SLAM is to keep the map consistent as new measurements are continuously added. A key challenge, however, lies in ensuring a map representation compatible with both these objectives: that is, a map that maintains free space information for planning but can also adapt efficiently to dynamically changing pose estimates from a graph-based SLAM system.**

In this paper, we propose an online global occupancy map that can be corrected for accumulated drift efficiently based on incremental solutions from a sparse graph-based SLAM optimization. Our map maintains free space information for real-time path planning while undergoing a bounded number of updates in each loop closure iteration. We evaluate performance for both simulated and real-world datasets for an application involving underwater exploration and mapping.

I. Introduction

We address the problem of maintaining a 3D map representation by a robot that can be used for autonomous exploration and mapping. The task of exploring an unknown 3D environment requires both simultaneous localization and mapping (SLAM) and path planning to work together. A key challenge, however, lies in ensuring a map representation that can be used by a path planner in real-time and can also adapt efficiently to dynamically changing pose estimates from a graph-based SLAM system.

In order for a robot to autonomously explore and map an unknown environment, its map representation must maintain free, occupied and unknown space information. Of the different 3D map representations, volumetric representations like occupancy grid maps [1] have the advantage of maintaining this information by dividing space into voxels that can then be classified as free, occupied or unknown. Occupancy grid maps, however, are restricted to a filtering paradigm wherein each map voxel is updated as a binary Bayes filter [1–3]. The Bayes filter makes use of the Markovian assumption to update a map voxel based on the most recent posterior state estimate and discards historical sensor measurements. This comes at the disadvantage that recovering from incorrect map updates due to uncertain poses becomes challenging. The drawback is particularly evident in event of loop closures from a graph-based SLAM system, which may require large adjustments to correct past inaccurate poses in the robot's trajectory.

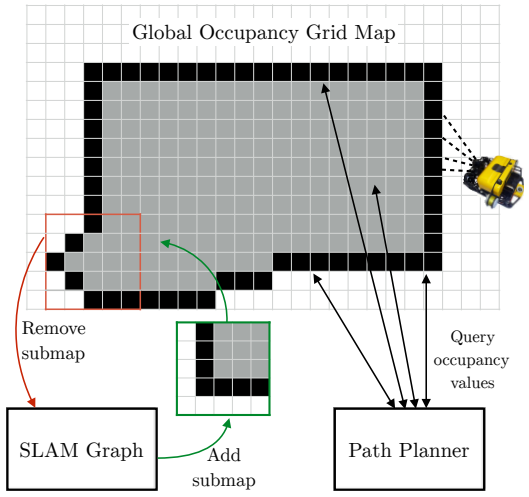

Fig. 1. Proposed framework for an online consistent occupancy map. SLAM maintains a pose graph and associated set of submaps. Loop closures result in pose updates that are applied incrementally to the global map by undoing updates from older submaps and adding back submaps with corrected poses. Planner works in parallel and queries the global map directly.

One solution [4, 5] to the problem is to have an explicit merge or rasterization step to generate a global occupancy map from optimized poses. However, this can become intractable for larger environments as the merged global map would have to be recomputed using all submaps every time there is an updated solution for the past poses. On the other hand, recent work [6] addressed this problem by maintaining multiple local occupancy submaps whose reference poses can undergo loop closure updates efficiently. These local occupancy submaps are directly used by the path planning algorithm, eliminating the need for a single global occupancy map. However, this puts the burden on the planner to query voxels from multiple submaps, leading to linearly rising query times, as opposed to a sublinear rise in the case of a single global map.

In this paper, we propose an online global occupancy map representation that falls between these two previously stated extremes. It retains submaps in the SLAM pose graph but also maintains a global occupancy map for planning that doesn't have to be fully remerged every loop closure iteration. Our key goal is that incremental solutions from a sparse

[1]Paloma Sodhi and Michael Kaess are with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {psodhi, kaess}@andrew.cmu.edu

[2]Bing-Jui Ho is with the AMoD Group at Aptiv, Pittsburgh, PA 15238, USA. bing-jui.ho@aptiv.com

This work was partially supported by the Office of Naval Research under award N00014-16-1-2103 and a University Presidential Fellowship.

graph-based SLAM optimization must reflect as incremental updates to the online global occupancy map. For this, we leverage two important factors—first, only a small number of poses in the SLAM graph are updated each loop closure iteration, and second, since probabilistic updates for each submap are independent of each other, we can incrementally undo updates based on a pose uncertainty criteria. This results in us being able to maintain a single global occupancy map that is: globally consistent every iteration, has sublinear query times for real-time path planning, and undergoes on average a bounded number of updates in each loop closure iteration (Fig. 1). In particular our main contributions are,

(1) An online global occupancy map representation that balances planning times and is compatible with graph-based SLAM systems.
(2) Evaluation and analysis of our proposed approach on both simulated and real-world datasets.
(3) Implementing the proposed approach for an autonomous underwater exploration and mapping application

II. Related Work

State-of-the-art SLAM algorithms are predominantly formulated as maximum a posteriori (MAP) estimation over a graph [6–11]. The MAP estimation framework has proven to be more accurate and efficient than previous approaches for SLAM based on nonlinear filtering [12]. The pose graph, which stores poses of the robot and spatial constraints between them, is a central data structure in graph-based SLAM. The graph representation that we make use of is a submap-based pose graph SLAM [6, 9–11]. This is a popular representation for large-scale SLAM systems wherein the SLAM frontend divides the world into a collection of locally consistent maps or *submaps* each with their own local coordinate frames [5, 6, 9–11, 13, 14] that can then be deformed with respect to each other for drift correction. The relative constraints between different submap coordinate frames are maintained either as a pose graph in the case of graph-based SLAM systems [9–11] or as a posterior belief in the case of filtering-based SLAM systems [5, 13].

Only a few of these submap-based SLAM systems explicitly model freespace information in their submaps [5, 6, 11, 13] for path planning purposes. Ho et al. [6] maintain multiple local occupancy grid submaps that the path planner queries directly, eliminating the need for a global map. While efficient for loop closures, the burden is placed on the planner to query multiple submaps leading to linearly rising planner query times with number of submaps, as opposed to a sublinear rise with a global map. On the other hand, Fairfield et al. [5, 13] use a global occupancy map created by merging several local occupancy grid submaps. This ensures efficient planning times, but loop closure updates can become intractable for larger environments as *all* submaps would need to be remerged every time there is an updated pose solution. Konolige et al. [11] maintain local occupancy grid submaps but within a hybrid metric-topological map framework. The hybrid metric-topological class of methods, however, rely on local planning with metric maps and global planning with topological maps and hence cannot be easily substituted for use with path planners that rely on standard occupancy maps.

From the planning side, state-of-the-art path planning algorithms for exploration and mapping of prior unknown environments [15–17] make use of an occupancy grid map representation of the world that can model free, occupied and unknown space. These systems make use of a filtering framework for updating the occupancy grid map. In order to use occupancy grid maps with a graph SLAM framework, however, an explicit merge or rasterization step is needed to generate an occupancy grid map from the underlying graph. Naively doing this for large environments by iterating over all graph nodes, however, would take significantly longer than the graph optimization itself, making it unsuitable for real-time planning requirements.

III. Proposed Mapping Approach

Given an incoming sequence of measurements, our objective is to build a global occupancy map that can represent free, occupied and unknown space. If the set of sensor poses are known with certainty at the time they are recorded, existing occupancy grid mapping techniques can be used directly to construct a global occupancy map. Our focus, however, is on occupancy mapping with uncertain poses i.e. when sensor poses are not well localized globally at the time they are recorded due to accumulated drift. When mapping with uncertain poses, our objective is to maintain a occupancy map that can correct itself efficiently based on loop closure updates from a SLAM system.

We build our mapping system upon the OctoMap framework [1]. OctoMap is an octree-based 3D occupancy grid mapping system that models free and occupied volumes and implicitly volumes not yet measured. The octree data structure underneath makes OctoMap an efficient representation for real-time robotics applications in 3D environments.

A. Map Construction

Let $\mathcal{M}$ denote our proposed online occupancy grid map. At each step we consider a local occupancy grid submap m_i being updated in $\mathcal{M}$. The reference frame for each local submap m_i is parameterized as a rigid transformation with respect to a global reference frame as $T_i^g \in SE(3)$. Each submap addition can then be expressed as,

$$\mathcal{M} \leftarrow \mathcal{M} \cup T_i^g m_i \tag{1}$$

Since each local occupancy grid m_i is stored as an octree data structure, the union operation in Eq. 1 involves iterating over all leaf node voxels of m_i, transforming voxel centers to global coordinates using T_i^g, and updating corresponding occupancy values in the global octree representing $\mathcal{M}$.

The local occupancy grid submap m_i used here can be constructed by locally accumulating sensor scans using odometry or frame-to-frame alignment. Sensor scans $\{z_{1:t}^i, o_{1:t}^i\}$ accumulated in each submap are integrated as m_i by ray casting from sensor scan origins $o_{1:t}^i$ to each measurement endpoint in $z_{1:t}^i$. In our application involving

use of a sonar sensor on an underwater robot, m_i is created by accumulating a set of sequential sensor scans over a finite time period t based on robot odometry. Each scan in this set is registered into a coordinate frame T_i^g placed at the pose of the first scan, also referred to as the base (or keyframe) pose. The time period t is determined by balancing the trade-off that submaps are short enough for accumulated odometry drift to stay low, yet large enough that the submaps have sufficient features for identifying loop closures. For a submap m_i, this time period t is computed by keeping the pose covariance Σ_i below a maximum threshold value, i.e.,

$$\Sigma_i = t \times diag(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_h^2, \sigma_p^2, \sigma_r^2) \leq \Sigma_{max} \quad (2)$$

where, $diag(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_h^2, \sigma_p^2, \sigma_r^2)$ is the 6×6 covariance matrix representing uncertainties in {x, y, z, yaw, pitch, roll} directions respectively, and the $\leq$ operation is done in an element-wise manner. For datasets collected by our underwater robot, we only consider uncertainties in $\{x, y, h\}$ directions for the t computation. This is because, as we'll see later, the onboard state estimation on our robot gives us absolute measurements in the $\{z, p, r\}$ directions.

B. SLAM Pose Graph

Fig. 2 illustrates the SLAM pose graph represented as a factor graph. A factor graph is a bipartite graph with two types of nodes: variables $\mathbf{x}_i \in \mathcal{V}$ and factors $\Phi_j \in \mathcal{U}$ [18]. In our case, the variable nodes $\mathbf{x}_i$ are the base poses T_i^g of each local occupancy submap m_i. The factor nodes encode differences between actual and predicted measurements as soft constraints to the graph. Since we get absolute $\{z, p, r\}$ measurements for our underwater vehicle, the odometry factor has been split as a relative pose-to-pose xyh factor and a unary zpr factor [8, 9]. The loop closure factor is obtained by applying ICP-based scan matching to submap point cloud pairs $\{m_i, m_k\}$ so as to get transformation l_{ik} as the actual measurement.

To estimate values of variable nodes, we need to perform *maximum a posteriori* (MAP) inference on the factor graph in Fig. 2. For Gaussian noise models, MAP inference is equivalent to solving a nonlinear least-squares optimization problem of the form [9, 18],

$$\hat{\mathcal{X}} = \underset{\mathcal{X}}{\operatorname{argmin}} \{ \underbrace{(||\mathbf{x}_0 \ominus \mathbf{x}_{prior}||^2_{\Sigma_0})}_{\text{pose prior}} + \sum_{i=1}^{N} (\underbrace{||f(\mathbf{x}_{i-1}, \mathbf{x}_i) - u_i||^2_{\Lambda_i}}_{\text{xyh factor}} + \underbrace{||g(\mathbf{x}_i) - v_i||^2_{\Gamma_i}}_{\text{zpr factor}}) + \sum_{(i,k) \in \mathcal{L}} \underbrace{(||h(\mathbf{x}_i, \mathbf{x}_k) - l_{ik}||^2_{\Xi_{ik}})}_{\text{loop closure factor}} \} \quad (3)$$

where, $||e||^2_\Sigma = e^T \Sigma^{-1} e$ is the Mahanalobis distance and $\ominus$ the difference between two manifold elements. $\mathcal{X}$ is the state comprising of all variables: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n\}$ where $\mathbf{x}_i \in SE(3)$ represents base pose transform T_i^g of submap m_i. The measurement prediction functions and covariances associated with the xyh, zpr, and loop closure factors are

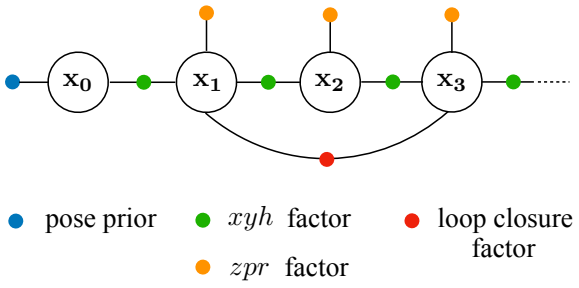

Fig. 2. SLAM pose graph represented as a factor graph. Variable nodes are shown as larger uncolored circles with corresponding variable names x_i. Factor nodes are shown as smaller color filled circles with color legend showing type of factor.

$\{f(\cdot), g(\cdot), h(\cdot)\}$ and $\{\Lambda_i, \Gamma_i, \Xi_{ik}\}$, respectively. $\mathcal{L}$ is set of all tuples (i, k) for which a pairwise registration constraint l_{ik} exists between poses $\mathbf{x}_i$ and $\mathbf{x}_k$.

To solve the least squares optimization in Eq. 3, we make use of the iSAM optimization library [19]. iSAM provides efficient algorithms for solving Eq. 3 incrementally. It does so by linearizing measurement prediction functions $f(\cdot), g(\cdot), h(\cdot)$ and collecting all components into one large system yielding a standard linear least squares of the form $\hat{\mathcal{X}} = \operatorname{argmin}_{\mathcal{X}} ||A\mathcal{X} - b||^2$. Since A is a sparse matrix, the linear least squares objective is solved for by computing a matrix factorization R of A. As new factors and variables are added to the graph, solution $\hat{\mathcal{X}}$ is incrementally computed by updating the previous factorization R directly instead of refactoring A. This leads to an efficient incremental algorithm (with periodic batch update steps) for computing $\hat{\mathcal{X}}$. A more thorough treatment is provided in [19].

C. Global Map Loop Closure Update

In the previous subsection, we looked at how incremental optimization for sparse graph-based SLAM provided a way to incrementally compute optimized posteriors for base poses $\hat{\mathcal{X}} = \{\hat{T}_i^g\}_{i=1}^N$ as new loop closures are added to the graph. Our objective now is to use these optimized pose posteriors to efficiently update our online global occupancy grid $\mathcal{M}$.

A naive approach would be iterate over all nodes of the graph and add all local submap measurements into a single global map. However, when applied to larger environments, this step would take significantly longer than the optimization itself, making it unsuitable for real-time planning requirements. Instead, we'd like to propagate the incremental pose updates from iSAM efficiently as incremental updates to our global occupancy grid map $\mathcal{M}$. Consider, after a loop closure update, the base pose values $\{T_i^g\}_{i=1}^N$ get updated to $\{\hat{T}_i^g\}_{i=1}^N$. We compute a subset $\mathcal{S}_1$ of local maps whose base pose updates exceed a threshold ϵ_1, that is,

$$m_i \in \mathcal{S}_1 \quad \forall \quad ||\hat{T}_i^g \ominus T_i^g|| > \epsilon_1. \quad (4)$$

For each local submap m_i, $i \in \mathcal{S}_1$, we further compute the marginal uncertainty $\Sigma_{i,i}$ of each base pose node. If a local submap m_i is well localized globally, it would have a low

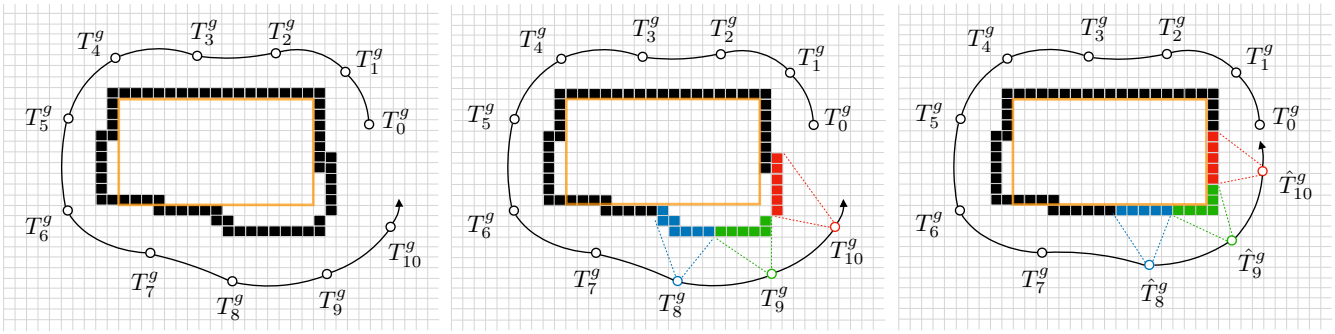

Fig. 3. **(a)-(c)** illustrates a loop closure update to the proposed online global occupancy grid map $\mathcal{M}$. The rectangular structure being mapped is shown in orange, occupied grid cells are shown in black. **(a)** shows the grid map constructed using base poses $\{T_i^g\}_{i=0}^{10}$ **(b)** After a loop closure, the incremental SLAM optimization returns updated base pose solutions $\{\hat{T}_i^g\}_{i=0}^{10}$. Based on this SLAM solution, submap subset $m_j \in \mathcal{S}_1 \cap \mathcal{S}_2$ is identified that must be updated in $\mathcal{M}$. Here, $j = 8, 9, 10$ with corresponding voxels shown in blue, green, red respectively. **(c)** shows incremental updates from the SLAM optimization reflected as corresponding updates to the global map $\mathcal{M}$.

marginal uncertainty, implying that measurements integrated from m_i lie within an acceptable uncertainty threshold. This can be expressed as,

$$m_i \in \mathcal{S}_2 \quad \forall \quad ||\Sigma_{i,i}|| > \epsilon_2 \tag{5}$$

where, $||\cdot||$ is the l^2-norm proportional to the volume of an ellipsoid of constant probability density defined by $\Sigma_{i,i}$.

The set $\mathcal{S}_2$ essentially identifies *stable* ($||\Sigma_{i,i}|| \leq \epsilon_2$) and *unstable* ($||\Sigma_{i,i}|| > \epsilon_2$) regions in the map, with pose uncertainty being the metric for map stability. For computing the marginal pose covariances $\Sigma_{i,i}$, we utilize the dynamic programming based approach included in the iSAM library [19] and detailed in [20] to efficiently recover arbitrary entries from the covariance matrix. This avoids inverting the entire information matrix.

Now the submap subset $m_j \in \{\mathcal{S}_1 \cap \mathcal{S}_2\}$ essentially denotes submaps that lie in unstable regions of the global map and whose base pose updates after a loop closure exceeds a threshold. This is the submap subset that must be updated within our online global occupancy map $\mathcal{M}$. To compute this update, consider first the standard recursive Bayes filter update rule for an occupancy map voxel n [1, 3],

$$\frac{p(n|z_{1:t})}{1-p(n|z_{1:t})} = \underbrace{\frac{p(n|z_t)}{1-p(n|z_t)}}_{\text{inverse sensor model}} \underbrace{\frac{p(n|z_{1:t-1})}{1-p(n|z_{1:t-1})}}_{\text{recursive term}} \underbrace{\frac{1-p(n)}{p(n)}}_{\text{prior}} \tag{6}$$

where, z_t is the measurement added at the most recent time step t. Expressing Eq. 6 in log-odds ratio form,

$$l(n|z_{1:t}) = l(n|z_t) + l(n|z_{1:t-1}) - l(n) \tag{7}$$

where, $l(n) = \log\left(p(n)/(1-p(n))\right)$ is the log-odds ratio.

We are interested in formulating a probabilistic measurement update as in Eqs. 6, 7 but now for updating online global map $\mathcal{M}$ based on measurements from the submap subset $m_j \in \{\mathcal{S}_1 \cap \mathcal{S}_2\}$. For this, we must perform a new measurement update based on optimized poses and undo previous updates from unoptimized poses. Consider $\{n_v^j\}_{v=1}^V$ to be the leaf node voxels of submap m_j, $\{\hat{T}_j^g,\ T_j^g\}$ the optimized and unoptimized base poses of m_j and $n \in \mathcal{M}$ the voxels in online global occupancy map $\mathcal{M}$. The recursive probabilistic update to $\mathcal{M}$ (with an additional term for undoing an update) for each submap m_j can then be written out as,

$$\begin{aligned}\frac{p(n|n_{1:v}^j, \hat{T}_j^g)}{1-p(n|n_{1:v}^j, \hat{T}_j^g)} = &\underbrace{\frac{p(n|n_v^j, \hat{T}_j^g)}{1-p(n|n_v^j, \hat{T}_j^g)}}_{\text{new update}} \underbrace{\frac{1-p(n|n_v^j, T_j^g)}{p(n|n_v^j, T_j^g)}}_{\text{undo old update}} \\ &\underbrace{\frac{p(n|n_{1:v-1}^j, \hat{T}_j^g)}{1-p(n|n_{1:v-1}^j, \hat{T}_j^g)}}_{\text{recursive term}} \underbrace{\frac{1-p(n)}{p(n)}}_{\text{prior}}\end{aligned}$$

$$\begin{aligned}\Rightarrow \quad l_{\mathcal{M}}(n|n_{1:v}^j, \hat{T}_j^g) = &\ l_{\mathcal{M}}(n|n_v^j, \hat{T}_j^g) - l_{\mathcal{M}}(n|n_v^j, T_j^g) + \\ &l_{\mathcal{M}}(n|n_{1:v-1}^j, \hat{T}_j^g) - l_{\mathcal{M}}(n)\end{aligned} \tag{8}$$

where, the subscript $\mathcal{M}$ to the log-odds function $l_{\mathcal{M}}(\cdot)$ denotes the map in which the log-odds lookup is being done.

Note that the *new update* term in Eq. 8 is equivalent to the *inverse sensor model* term in Eq. 6. The log-odds form $l_{\mathcal{M}}(n|n_v^j, \hat{T}_j^g)$ of this term denotes how voxels $n \in \mathcal{M}$ must be updated on addition of measurements $\{n_v^j, \hat{T}_j^g\}$ coming from local submap m_j. Conversely, the log odds form of the *undo old update* term in Eq. 8 denotes how voxels $n \in \mathcal{M}$ must be updated on removal of measurements $\{n_v^j, T_j^g\}$ coming from local submap m_j. This leads to a simple but efficient algorithm, as summarized in Algorithm 1, for doing loop closure updates in the online global occupancy map $\mathcal{M}$. Additionally, Fig. 3 too illustrates the process of updating voxels in $\mathcal{M}$ based on loop closure updates.

We make two underlying assumptions when defining the *undo update* term in Eq. 8: firstly updates to each voxel

Algorithm 1 Global Map Loop Closure Update

Input $\mathcal{M}$: proposed online global occupancy map,
$\{m_i\}_{i=1}^N$: local occupancy submaps,
$\{T_i^g\}_{i=1}^N$: submap poses before loop closure,
$\{\hat{T}_i^g\}_{i=1}^N$: optimized submap poses after loop closure,
$\{\Sigma_{i,i}\}_{i=1}^N$: marginal pose covariances before loop closure
Output $\mathcal{M}$
Parameters ϵ_1, ϵ_2

for i = 1 to N **do**
  updateFlag = $(||\hat{T}_i^g \ominus T_i^g|| > \epsilon_1)$ & $(||\Sigma_{i,i}|| > \epsilon_2)$
  if updateFlag **then**
    $n_{1:V}^i \leftarrow$ getLeafNodeVoxels(m_i)
    $X_{1:V}^i \leftarrow$ getVoxelCenters$(n_{1:V}^i)$
    for v = 1 to V **do**
      $l_{\mathcal{M}}(T_i^g X_v^i) \leftarrow l_{\mathcal{M}}(T_i^g X_v^i) - l_{m_i}(X_v^i)$
      $l_{\mathcal{M}}(\hat{T}_i^g X_v^i) \leftarrow l_{\mathcal{M}}(\hat{T}_i^g X_v^i) + l_{m_i}(X_v^i)$
    end for
  end if
end for

$n \in \mathcal{M}$ are independent, and secondly probabilistic update is an invertible function. The first assumption holds true for occupancy grid mapping in general. For the second assumption to hold true, clamping thresholds (as also used by Octomap [1]) must be set to their minimum and maximum possible values, that is $p_{min} = 0,\ p_{max} = 1$ for all,

$$l_{\mathcal{M}}(n|n_{1:v}^j, \hat{T}_j^g) = \max\left(\min\left(l_{\mathcal{M}}(n|n_{1:v}^j, \hat{T}_j^g), l_{max}\right), l_{min}\right) \tag{9}$$

where, $l_{min},\ l_{max}$ are the log-odds values corresponding to $p_{min},\ p_{max}$ respectively. This would ensure that information close to $p = 0,\ p = 1$ is not lost, and the probabilistic map update function is invertible.

D. Map Occupancy Queries

Occupancy queries to the map are typically made by the path planning module. We make use of a sampling-based exploration planner [17] that queries the map in the form of ray cast queries. Ray cast queries are needed by the planner for computing collision-free paths and for computing view utility gains of sensor rays casted from next-best-viewpoints.

Map queries to our online global occupancy map $\mathcal{M}$ are performed the same way as in Octomap [1]. Ray cast queries are performed by stepping along a ray from sensor origin to an end point, and returning the occupancy values for all voxels along that ray. Each voxel query on an octree data structure of tree depth d_{max} can be performed sub-linearly with a complexity of $O(d_{max}) = O(\log n)$, where n is the number of nodes in the tree. Like Octomap, the use of octrees also enables multi-resolution queries, where tree traversal is stopped at a fixed depth.

IV. Results and Evaluation

We evaluate our proposed mapping approach on metrics like map quality, accuracies, query and update time complexities. We perform this evaluation for two simulated datasets and one real-world dataset. Since our application of interest is underwater exploration, we collect these datasets by making our underwater robot explore simulated and real-world underwater environments. All evaluation is done on a laptop with an Intel Core i7-7820HQ 2.9GHz processor.

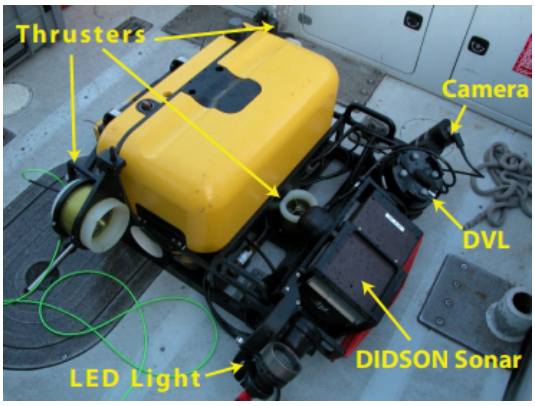

Fig. 4. Bluefin hovering autonomous underwater vehicle (HAUV)

A. Experimental Setup

For generating the simulated datasets, we adapt the UUV Simulator [21], a gazebo-based underwater simulator, and customize it for our underwater robot model. The underwater robot that we use is the Bluefin hovering autonomous underwater vehicle (HAUV) as seen in Fig. 4. The Bluefin HAUV is equipped with several sensors for onboard navigation: a Doppler velocity log (DVL), an attitude and heading reference system (AHRS), and a depth sensor. The depth sensor gives direct measurements of the vehicle's z position. Additionally, the AHRS is also capable of providing direct, drift-free estimates of vehicle's roll r and pitch p angles. The vehicle's x,y positions and yaw h are not directly observable and instead estimated by accumulating dead reckoning of the DVL and IMU odometry measurements. As a result, the vehicle's onboard navigation system gives us a drifting pose estimate in the $\{x, y, h\}$ directions, and an absolute pose estimate in the $\{z, p, r\}$ directions [8].

To replicate this navigation payload on the simulated HAUV, we add gaussian noise to *absolute* ground truth estimates in the $\{z, p, r\}$ directions, and gaussian noise to the *relative* ground truth estimates in the $\{x, y, h\}$ directions.

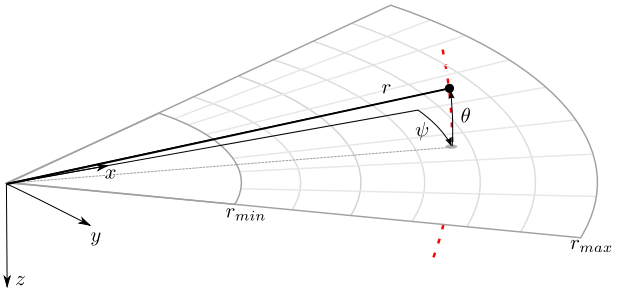

Fig. 5. Geometry of a single sonar scan.

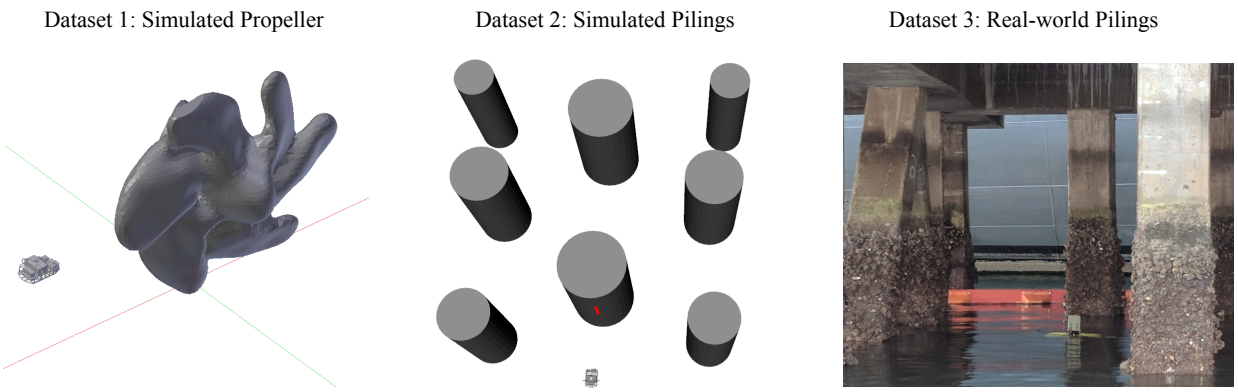

Fig. 6. Simulated and real-world underwater environments used for generating the datasets

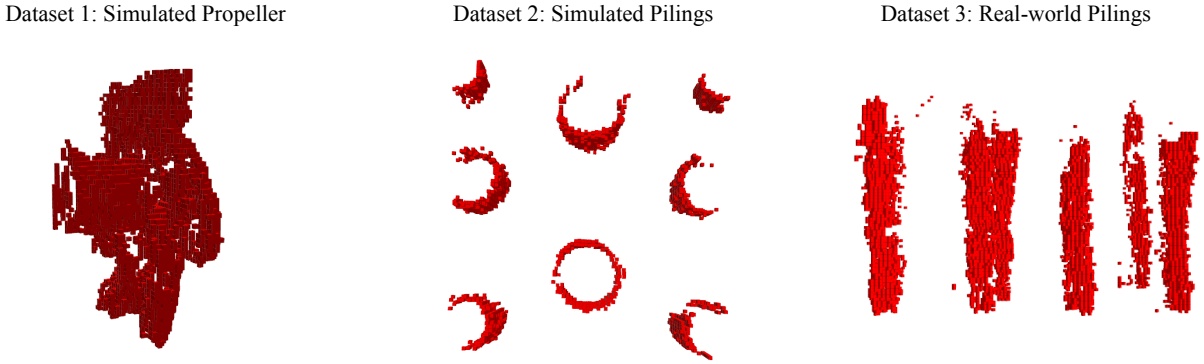

Fig. 7. Qualitative reconstructions of proposed online occupancy map

Consider $\mathbf{x}_t, \mathbf{x}_{t+1}$ as the uncorrupted ground truth state estimates, $\tilde{\mathbf{x}}_t$ the current corrupted state estimate, the corrupted next time step estimate $\tilde{\mathbf{x}}_{t+1}$ can then be computed as,

$$\begin{aligned} \Delta\tilde{\mathbf{x}} &= (\mathbf{x}_{t+1} \ominus \mathbf{x}_t) \oplus \mathcal{N}(0, \Sigma_{xyh}) \\ \tilde{\mathbf{x}}_{t+1} &= \tilde{\mathbf{x}}_t \oplus \Delta\tilde{\mathbf{x}} \\ \tilde{\mathbf{x}}_{t+1} &= \tilde{\mathbf{x}}_{t+1} \oplus \mathcal{N}(0, \Sigma_{zpr}) \end{aligned} \tag{10}$$

where, $\Sigma_{xyh} = diag(\sigma_x^2, \sigma_y^2, 0, \sigma_h^2, 0, 0)$ are the covariance values in $\{x, y, h\}$ directions and $\Sigma_{zpr} = diag(0, 0, \sigma_z^2, 0, \sigma_p^2, \sigma_r^2)$ are covariances in the $\{z, p, r\}$ directions. The variance values that we use for all simulated experiments are $\sigma_x^2 = 0.00138\ m^2/s$, $\sigma_y^2 = 0.00138\ m^2/s$, $\sigma_h^2 = 10^{-7}\ rad^2/s$, $\sigma_z^2 = 0.0001\ m^2/s$, $\sigma_p^2 = 10^{-7}\ rad^2/s$, $\sigma_r^2 = 10^{-7}\ rad^2/s$.

For sensing, we equip the simulated HAUV with a 1D laser line sensor in order to replicate the profiling sonar sensor on the real vehicle. A schematic of the profiling sonar sensor geometry is shown in Fig. 5. Each scan consists of 96 beams evenly spaced within sonar's $\psi = 29^o$ horizontal field-of-view. Sensor readings are retuned as $\{r, \psi\}$ values which are converted to cartesian coordinates. In the case of the real sonar sensor, the elevation angle opening is a non-zero value $\theta \approx 1^o$, which we don't explicitly model in our simulated line sensor. As a result, there is greater vertical ambiguity in the real sonar returns compared to the simulated returns.

B. Datasets and Baselines

We collect datasets for the following underwater environments: simulated propeller, simulated pilings, and real-world pilings. These environments are visualized in Fig. 6. To collect the simulated datasets, a next-best-view based exploration path planner [17] was used for planning collision-free trajectories through complex structures in the environments. The real-world dataset was collected using the real HAUV during field trials in San Diego in Nov 2018. During the field trials, the HAUV was controlled using manually set waypoints due to constraints like vehicle tether that are currently not handled by our exploration planner.

We compare our proposed mapping approach against: (a) a global occupancy grid map remerged using all submaps with SLAM optimized poses and (b) *VOG Map* [6] that only maintains local occupancy grid submaps and doesn't do any global merging. To generate (a), the remerge can be done using either local occupancy grids in each submap m_i or raw measurements $\{z_{1:t}^i, o_{1:t}^i\}$ in each submap m_i. Reintegrating raw measurements would involve iterating over all measurements and ray casting these again into a new map, which is clearly a computationally expensive process not suitable for real-time SLAM and planning. Hence, when comparing map query and map update times, we'll compare against a global occupancy grid remerged using the local

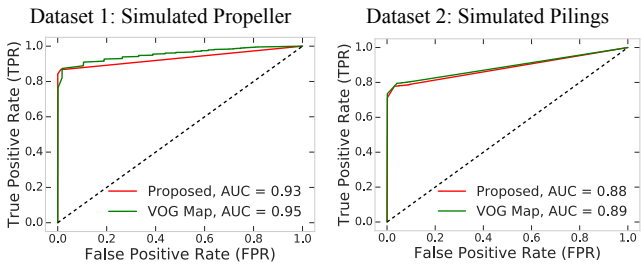

Fig. 10. ROC curves computed against ground truth for simulated datasets

occupancy grids directly in each submap m_i. We refer to this as a *Full Merge* map, and is the equivalent of doing a batch update on a global occupancy grid map.

C. Map Accuracies

Fig. 7 shows qualitative results for the proposed online occupancy grid maps corresponding to the underwater environments in Fig. 6. Occupied voxels for the environment explored up till 100 submaps have been visualized. In all the experiments, number of sonar scans in each submap were kept at 200, octree resolution at 10cm, probability hit as 0.75, probability miss as 0.20, probability occupied threshold as 0.7, probability free threshold as 0.3 and clamping thresholds as $p_{min} = 0,\ p_{max} = 1$.

For the two simulated datasets, we also compare map accuracies quantitatively against ground truth occupancy maps. Fig. 10 plots the Receiver Operating Characteristic (ROC) curves with corresponding Area under Curve (AUC) values for proposed and VOG map. VOG map was generated for comparison by querying global coordinates in multiple local submaps and summing their log-odds. Since VOG map uses the most recent SLAM poses, it has an accurate reconstruction. Fig. 10 shows the proposed map having similar AUC values as VOG map despite sparse updates. A primary reason for AUC values not being closer to 1 for both VOG and proposed maps are the quality of SLAM loop closures—which seem to be better for the simulated propeller dataset over the simulated pilings.

D. Map Query Complexities

Occupancy queries to the map are made by the path planner. The queries from the exploration path planner [17] that we use are primarily ray cast queries. Ray cast queries on the map are performed by stepping along a ray from ray origin to ray end point, returning occupancy voxels of all voxels along that ray. We profile ray cast query times for 1000 rays of varying lengths ($\leq$ 4m) sampled randomly in the explored map regions. Fig. 8 shows the median query times for a single ray against number of submaps for proposed map, VOG map and the Full Merge map. VOG map has a linearly rising map query complexity since it queries multiple local overlapping submaps. Our proposed approach, however, has a sublinear query time (similar to Full Merge) since it maintains a single global map.

E. Map Update Complexities

Fig. 9 shows the cumulative average over map update times against number of submaps. It can be seen that the Full Merge map has a rising map update time, with sizable jumps at iterations where a loop closure update happens. VOG-map

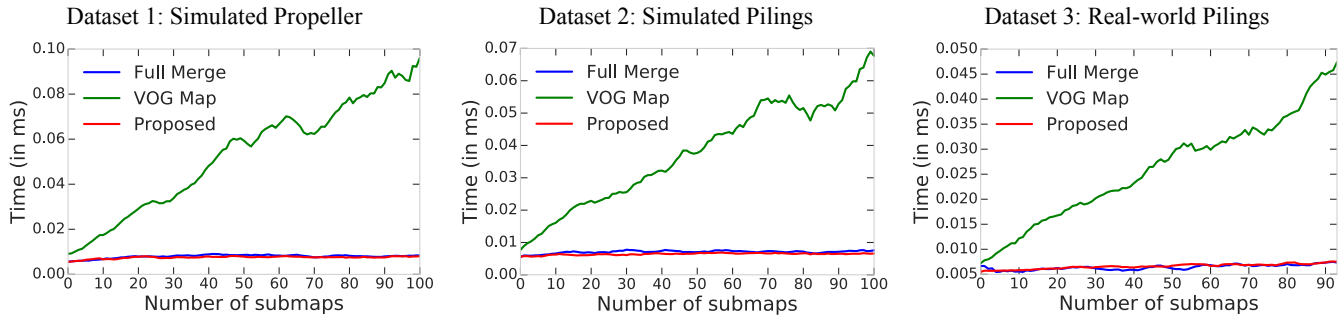

Fig. 8. Median ray cast query times

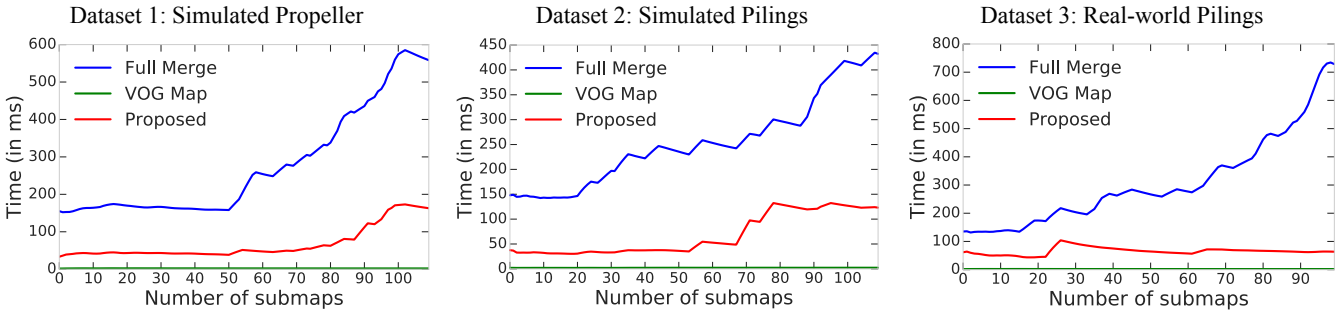

Fig. 9. Cumulative average over the map update times

has almost a constant time update since each map update step only involves correcting the base pose values of local submaps. Our proposed online occupancy map maintains a global map without rising map update times. This is because our map undergoes on average a bounded number of updates each loop closure iteration proportional to the impact of the loop closures.

Thresholds ϵ_1, ϵ_2 affect the submap subset $m_j \in \{\mathcal{S}_1 \cap \mathcal{S}_2\}$ that must be incrementally updated in our online map. ϵ_1 was chosen to be 10^{-2} or $\sim 0.6^o$ for differences in yaw, pitch, roll axes and 2×10^{-3} or 0.2cm for differences in x, y, z axes. $||\Sigma_{i,i}||$ is taken to be the volume of the 95% confidence ellipse of the uncertainty values in the x, y, h directions. We set ϵ_2 as the maximum uncertainty volume reached in the initialization phase (chosen here as $j \leq 10$).

F. Overall System Performance

From the planning side, the exploration planner that we use [17] extensively queries the occupancy map to compute information gain. On average, in each planning iteration, the planner performs $\sim 100,000$ ray cast queries. In the case of VOG map, the median query times in Fig. 8 would cause each planning iteration to be in the order of seconds within 100 submaps. Having sublinear query times, as is the case for the proposed map or the Full Merge map, is essential for real-time planning.

From the SLAM side when there is a loop closure, Full Merge map recreates the global map by iterating over all nodes of the pose graph. Our proposed map bounds rising map update times by selecting a small subset of submaps and incrementally updating the global map. While VOG map has a more efficient update, however, update time for proposed map is easily amortized over the $\sim 100,000$ planner queries that happen each planning iteration in sublinear times.

V. Discussions

We present an online global occupancy map representation that can be corrected for accumulated drift efficiently based on incremental solutions from SLAM. A key challenge is ensuring that the map can be queried by a path planner in real-time as well as be updated efficiently by dynamically changing pose estimates from the SLAM system. Our solution is to maintain both a global occupancy map and a set of submaps in the SLAM pose graph. Every time a pose is updated, a submap is removed and a submap with the corrected pose is added to the global map. We leverage two properties—sparsity of updates and independence of submaps and demonstrate that this results in both bounded query time and bounded update time.

One issue that remains, however, is that the memory footprint of the system still continues to rise linearly. In current SLAM formulations, this is inevitable as submaps are used for determining loop closure. One way to overcome this could be to formalize a criteria for rejecting a new measurement if the information content is negligible. Since entropy can only decrease, this can potentially guarantee a bound on the number of submaps.

Acknowledgements

We would like to thank E. Westman for insightful discussions and help with real-world data collection. We would also like to thank P.V. Teixeira for making his underwater SLAM code available.

References

[1] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[3] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 2, pp. 116–121, 1985.

[4] N. Fairfield and D. Wettergreen, "Active SLAM and loop prediction with the segmented map using simplified models," in *Field and Service Robotics*, pp. 173–182, Springer, 2010.

[5] N. Fairfield, *Localization, mapping, and planning in 3D environments*. PhD thesis, 2009.

[6] B.-J. Ho, P. Sodhi, P. Teixeira, M. Hsiao, T. Kusnur, and M. Kaess, "Virtual occupancy grid map for submap-based pose graph SLAM and planning in 3D environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2175–2182, 2018.

[7] J. Li, M. Kaess, R. M. Eustice, and M. Johnson-Roberson, "Pose-graph SLAM using forward-looking sonar," vol. 3, no. 3, pp. 2330–2337, 2018.

[8] E. Westman and M. Kaess, "Underwater AprilTag SLAM and calibration for high precision robot localization," Tech. Rep. CMU-RI-TR-18-43, Robotics Institute, Carnegie Mellon University, 2018.

[9] P. V. Teixeira, M. Kaess, F. S. Hover, and J. J. Leonard, "Underwater inspection using sonar-based volumetric submaps," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4288–4295, 2016.

[10] R. Wagner, U. Frese, and B. Bäuml, "Graph SLAM with signed distance function maps on a humanoid robot," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2691–2698, 2014.

[11] K. Konolige, E. Marder-Eppstein, and B. Marthi, "Navigation in hybrid metric-topological maps," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3041–3047, 2011.

[12] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[13] N. Fairfield, D. Wettergreen, and G. Kantor, "Segmented SLAM in three-dimensional environments," *J. of Field Robotics*, vol. 27, no. 1, pp. 85–103, 2010.

[14] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous localization and map building in large-scale cyclic environments using the atlas framework," *Intl. J. of Robotics Research*, vol. 23, no. 12, pp. 1113–1139, 2004.

[15] C. Papachristos, S. Khattak, and K. Alexis, "Uncertainty-aware receding horizon exploration and mapping using aerial robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 4568–4575, 2017.

[16] E. Vidal, J. D. Hernández, K. Istenič, and M. Carreras, "Online view planning for inspecting unexplored underwater structures," vol. 2, no. 3, pp. 1436–1443, 2017.

[17] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3D exploration," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1462–1468, 2016.

[18] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[19] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[20] M. Kaess and F. Dellaert, "Covariance recovery from a square root information matrix for data association," *J. of Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1198–1210, 2009.

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