2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Virtual Occupancy Grid Map for Submap-based Pose Graph SLAM and Planning in 3D Environments

Bing-Jui Ho[1], Paloma Sodhi[1], Pedro Teixeira[2], Ming Hsiao[1], Tushar Kusnur[3], and Michael Kaess[1]

***Abstract*— In this paper, we propose a mapping approach that constructs a globally deformable virtual occupancy grid map (VOG-map) based on local submaps. Such a 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. We demonstrate use of such a representation for implementing an underwater SLAM system in which the robot actively plans paths to generate accurate 3D scene reconstructions. We evaluate performance on simulated as well as real-world experiments. Our work furthers capabilities of mobile robots actively mapping and exploring unstructured, three dimensional environments.**

I. Introduction

Active mapping (or exploration) involves controlling robot motion and sensor configurations so as to gather information for purposes of geometric or semantic scene reconstruction. As robotic systems become increasingly autonomous, the problem of active mapping has received considerable attention in recent years [1, 14, 16] and has found its way into a variety of applications including inspection of bridges and underwater structures, search and rescue, reef mapping, and crop monitoring.

In order to actively map a 3D environment with high fidelity, a robot needs to plan paths to informative viewpoints as well as be able to accurately localize itself within the map. State-of-the-art planning algorithms [1, 14, 16] for actively mapping unknown 3D environments require an input map that can offer both free and occupied space information. The most commonly employed world representation for such a purpose is an occupancy grid that divides the entire space into *voxels* that can then be classified as free, occupied or unknown. Voxels in an occupancy grid, however, are rigidly aligned with respect to each other and can hence not be updated efficiently for correcting global map drift based on loop closure constraints from the simultaneous localization and mapping (SLAM) system.

On the other hand, state-of-the-art SLAM systems like submap-based pose graph SLAM use a world representation involving a graph of robot poses with each pose associated with a local 3D point cloud [4, 8, 15]. Such representations can correct their global map drift by incorporating loop closure constraints and optimizing the SLAM graph. They, however, do not preserve direct free space information as required by path planning algorithms.

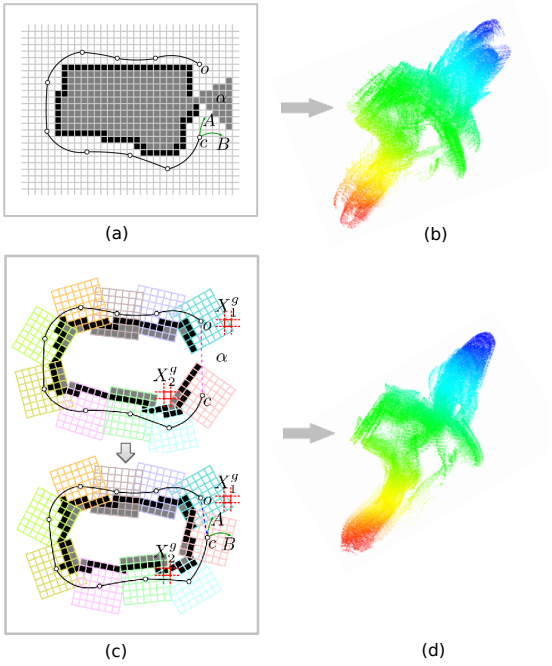


Fig. 1. Comparison of scene reconstruction quality for active mapping with standard global occupancy grid map and with our virtual occupancy grid map (VOG-map). **(a)** World as a standard global occupancy map as used in many current path planning algorithms. **(c)** The same world as a VOG-map capable of undergoing corrections from global loop closures from SLAM. **(b)**, **(d)** Corresponding 3D reconstructions of a large ship propeller corresponding to using the world representations in (a) and (c), respectively.

In this paper, we propose a submap-based mapping approach that can correct global map drift based on loop closure constraints from the SLAM system while maintaining free space information for path planning. Our solution is directly applicable for active mapping and exploration tasks that robots must perform in the real-world. In particular, our main contributions are:

(1) We introduce the virtual occupancy grid map (VOG-map) that can model free and occupied space as well as correct global map drift based on SLAM loop closures.
(2) We use (1) to implement an underwater SLAM system that actively plans paths to generate accurate 3D scene reconstructions.
(3) We evaluate (2) in both simulation and real-world experiments.

[1]Bing-Jui Ho, Paloma Sodhi, Ming Hsiao, and Michael Kaess are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {bingjuih, psodhi, mhsiao, kaess}@andrew.cmu.edu

[2]Pedro Teixeira is with the Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA 02139, USA. pvt@mit.edu

[3]Tushar Kusnur is with the Department of Electrical & Electronics Engineering, BITS Pilani K. K. Birla Goa Campus, India kusnur.tushar@gmail.com

This work was partially supported by the Office of Naval Research under award N00014-16-1-2103 and by the National Science Foundation under award IIS-1426703.

II. Related Work

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 [2, 6, 8, 10, 15, 17]. In some of these systems [2, 6] the robot is localized within the local coordinate system of a submap and gets re-localized when switching submaps. In other more recent systems [8, 10, 15, 17] each submap still maintains its own local coordinate system, but globally there exists a pose graph that constrains submaps with respect to each other. The pose graph here is essentially a graph with nodes being local coordinate frames of each submap and edges being odometry or loop closure constraints between submaps.

Of these various SLAM systems, only a few [6, 10] explicitly model free space information in their submaps for path planning. Fairfield et al. [6] do so for 3D environments by maintaining an occupancy grid at each submap and using a particle filter based approach to maintain a global structure between submaps. Konolige et al. [10] do so for 2D environments by maintaining an occupancy grid at each submap, but instead of particle states, they use a pose graph to maintain a global structure. Both these systems, however, require different path planning methods for short-term planning within local metric maps and for long-term planning on the global topological map. Moreover, since the global map structure is topological, they need further processing for generating metrically consistent global 3D scene reconstructions. Subsequent work by Fairfield et al. [5] uses an RRT planner within their particle filter based SegSLAM system [6]. In order to use this RRT planner in the full global map, they create a new merged occupancy grid by transforming each local submap voxel into a common global coordinate frame. This can become an expensive operation when there are frequent loop closures, since the merged grid would have to be recomputed every time there is an updated solution for the local submap transforms.

On the other hand, path planning systems for actively mapping unknown 3D environments [1, 14, 16] plan paths using a single global occupancy grid of the world that can offer both free and occupied space information. However, the main drawback of such a representation is that the global occupancy map that they use is unable to correct globally accumulated drift via loop closures from a SLAM system.

This motivates us to come up with a representation that can be compatible with both large-scale SLAM and path planning systems. In this paper, we propose a virtual occupancy grid map (VOG-map) representation that maintains a submap-based deformable global map structure but at the same time can be accessed like any standard global occupancy map by path planning systems such as [1, 14, 16].

III. Virtual Occupancy Grid Map

The proposed virtual occupancy grid map (VOG-map) is implemented based on the OctoMap framework [7]. OctoMap is an octree-based 3D occupancy mapping system that can model free and 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.

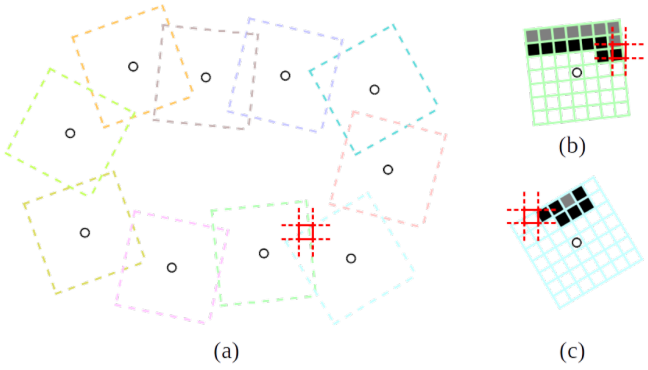


Fig. 2. Structure of the virtual occupancy grid map (VOG-map). **(a)** VOG-map consists of a set of local occupancy submaps m_i (dotted squares), each with a global pose t_i^{global} (black circle). **(b)**, **(c)** For a queried global location X^g (red grid), its occupancy value is computed as a sum of log-odds occupancy of local queries in each submap m_i.

Unlike a standard global 3D occupancy grid map, however, VOG-map representation of the world can be deformed and corrected for globally accumulated drift whenever the SLAM graph optimization computes an updated solution. It achieves this by representing the VOG-map as a set of *local occupancy submaps* whose base poses form nodes in the SLAM pose graph. Fig. 2 illustrates this VOG-map structure in 2D. The following subsections describe the VOG-map along with details on standard map operations like measurement updates and occupancy queries performed on VOG-map.

A. Map Representation

To represent VOG-map $\mathcal{M}_{vog}$, consider a set of N *local occupancy submaps* $\{m_i\}_{i=1}^N$. Point coordinates in each submap m_i are expressed locally with respect to a reference frame placed at the base pose $\mathbf{x}_i$ of the submap. As detailed in Section IV-A, values for base poses $\{\mathbf{x}_i\}_{i=1}^N$ are available to us as solutions from the SLAM pose graph optimization. Let t_i^{global} denote the global reference frame for submap m_i. This t_i^{global} simply equals the 6D base pose $\mathbf{x}_i$ if the SLAM pose graph represents robot trajectories in the global frame.

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

$$\mathcal{M}_{vog} = \left\{ \{m_1, t_1^{global}\}, \{m_2, t_2^{global}\}, \dots \{m_N, t_N^{global}\} \right\} \tag{1}$$

Here we do not explicitly compute $\mathcal{M}_{vog}$ by merging all local occupancy submaps $\{m_i\}_{i=1}^N$ together. Instead we do multiple local lookup queries for any global query made to $\mathcal{M}_{vog}$. More details are in Section III-D.

B. Construction of Local Submaps

A local occupancy submap 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 submap m_i. Here, we assume that the sensor scan belongs to a range finder sensor like a sonar or a lidar.

When determining the time period Δt for accumulating scans within a submap, the trade-off is that submaps should be large enough to have sufficient features for doing loop closures during SLAM but short enough for accumulated odometry error to stay low. For a submap m_i, this time period Δt is computed by keeping the 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 for measurement uncertainties in robot pose values $\{x, y, z, \theta, \phi, \psi\}$ every time iteration.

C. Map Update

All sensor measurement updates to the VOG-map are done locally within the submap in which these readings are observed. Individual sensor scan readings $z^i_{1:t}$ observed in the local occupancy submap m_i are integrated by performing a ray casting operation from sensor scan origins $c^i_{1:t}$ to each measurement endpoint in $z^i_{1:t}$. The occupancy probabilities $P(n|z^i_{1:t})$ for all voxels $n \in m_i$ along each beam are updated according to [7], [13]

$$P(n|z^i_{1:t}) = \left[1 + \frac{1 - P(n|z^i_t)}{P(n|z^i_t)} \frac{1 - P(n|z^i_{1:t-1})}{P(n|z^i_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1} \tag{2}$$

Using log-odds notation and assuming uniform prior probability $P(n) = 0.5$, Eq. 2 simplifies to the familiar log-odds update rule

$$\begin{aligned} L(n|z^i_{1:t}) &= L(n|z^i_{1:t-1}) + L(n|z^i_t) \\ \text{where,} \qquad L(n) &= \log\left[\frac{P(n)}{1 - P(n)}\right] \end{aligned} \tag{3}$$

We too use a clamping policy as in OctoMap [7], that defines an upper and lower bound on the occupancy estimate,

$$L(n|z^i_{1:t}) = \max\left(\min\left(L(n|z^i_{1:t}), l_{max}\right), l_{min}\right) \tag{4}$$

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

The measurement update rules in Eqs. (2)-(4) can be used with any kind of distance sensor, as long as an inverse sensor model is available. For instance, for a beam-based inverse sensor model as used in our robotic application later, the ray casting operations from sensor scan origins $c^i_{1:t}$ to each measurement endpoint in $z^i_{1:t}$ updates the endpoint voxels as occupied and all other voxels along the rays as free.

D. Map Query

All occupancy queries to the VOG-map are done by converting the query (in global coordinates) into multiple queries (in local coordinates) that are passed onto each local submap. Let X^g be the global 3D position of the voxel whose occupancy value needs to be looked up. We use a 4×4 transformation matrix T^i_g to map X^g into a local coordinate X^i computed as $\bar{X}^i = T^i_g \bar{X}^g$ (in homogeneous coordinates) which is then passed as an occupancy query to submap m_i. This is done for all submaps m_i with $i = 1 \ldots N$. The transformation matrix T^i_g is computed by taking the inverse of the global reference frame t^{global}_i for each submap m_i.

The occupancy probability values returned from all local occupancy submaps $\{m_i\}^N_{i=1}$ can now be combined together using the same log-odds update rule as seen in Eq. 3. This is because combining measurements from multiple local submaps is a similar operation as combining multiple measurement updates in a single global map. Also, since every new sensor measurement is incorporated only once (in any one of the local submaps), we do not run the risk of double counting measurements. The log odds occupancy query for a global 3D location X^g is hence expressed as

$$\begin{aligned} L(X^g) &= \sum_{i=1}^{N} L_i(n_i) \\ L(X^g) &= \max\left(\min\left(L(X^g),\ l_{max}\right),\ l_{min}\right) \end{aligned} \tag{5}$$

where, n_i is the voxel in submap m_i that contains local 3D coordinates X^i. $L_i(\cdot)$ implies that the log odds lookup is done in local occupancy submap m_i. l_{min} and l_{max} are the same clamping thresholds as used in Eq. 4.

Occupancy queries to the VOG-map are typically made by the path planner in form of ray casting queries i.e. casting a ray from a robot pose into a given direction and returning occupancy values for voxels along that ray. As we'll see in Section IV-B, the planner requires such ray casting queries for computing collision-free paths and for computing view utility gains of sensor rays casted from next-best-viewpoints.

The process of doing ray casting queries in the VOG-map is illustrated in Algorithm 1. It begins by finding a subset $\mathcal{S}$ of $\{m_i\}^N_{i=1}$ local occupancy submaps that intersect with a ray having origin c (in global coordinates) and direction vector v. It then steps along the ray and computes occupancy

Algorithm 1 Ray casting queries in VOG-map

Initialize: Occupancy values along ray $\mathcal{O} = \emptyset$,
Step size Δs, Number of steps n_{steps}
$\mathcal{S} \leftarrow$ RayIntersectionCheck(c, v, $\mathcal{M}_{vog}$)
for k = 1 to n_{steps} **do**
  $X^g = c + (k\Delta s)v$
  Set $L(X^g) = 0$
  for $i \in \mathcal{S}$ **do**
    Get transform matrix T^i_g for submap m_i
    Compute local coordinates $X^i = T^i_g X^g$
    if $X^i \in m_i$ **then**
      $n_i \leftarrow$ searchVoxel(m_i, X^i)
      $L(X^g) \leftarrow L(X^g) + L_i(n_i)$
  $L(X^g) \leftarrow \max(\min(L(X^g),\ l_{max}),\ l_{min})$
  $P(X^g) \leftarrow$ GetOccupancyProbability$(L(X^g))$
  $\mathcal{O} \leftarrow \mathcal{O} \cup \{P(X^g)\}$
return $\mathcal{O}$

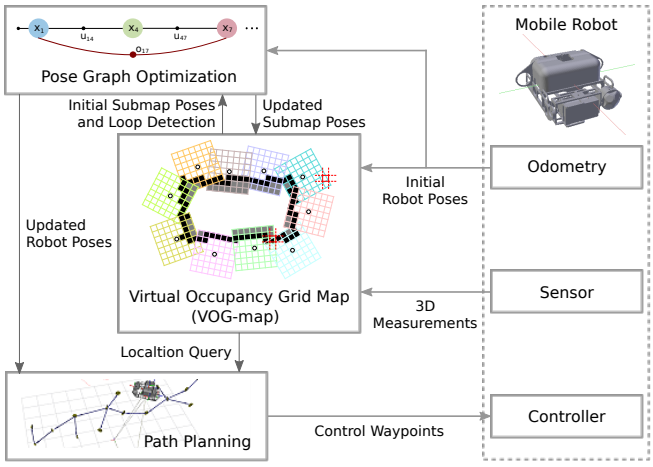


Fig. 3. Overall system employing a virtual occupancy grid map (VOG-map) for real-time planning and SLAM in order to actively map an unknown underwater 3D environment.

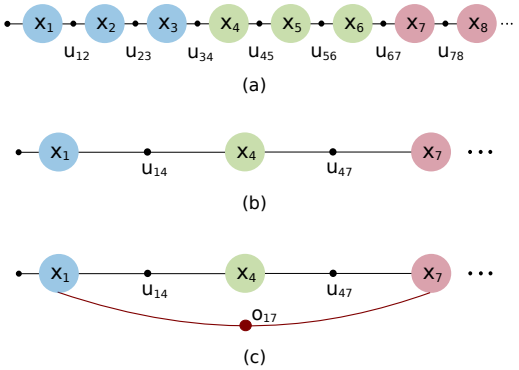


Fig. 4. SLAM pose graph formation. **(a)** Robot poses $\mathbf{x}_j$ as nodes of the graph and odometry constraints $u_{j,j+1}$ as edges. **(b)** Same graph with each node being a base pose $\mathbf{x}_i$ linked to a submap instead of an individual sensor scan. **(c)** The graph with added loop closure constraint o_{17} between non-sequential submap base poses $\mathbf{x}_1$ and $\mathbf{x}_7$.

values for global 3D coordinates X^g at each step using the sum of log odds rule in Eq. 5. As a result, the summation in Eq. 5 is taken only over a subset $\mathcal{S}$ of submaps that intersect with the viewing ray instead of all $\{m_i\}_{i=1}^N$ submaps.

IV. Robotic Exploration Application

To demonstrate the applicability of our mapping representation, we evaluate the proposed VOG-map structure on an underwater robot actively exploring an unknown 3D environment. For performing SLAM, we use a submap-based pose graph SLAM approach [15] while for path planning, we use a sampling-based next-best-view planning approach [1]. Fig. 3 illustrates our overall system and information flow between individual modules. We elaborate below the SLAM and path planning components of our system.

A. Submap-based Pose Graph SLAM

The submap-based pose graph SLAM approach [15] involves creating submaps locally by accumulating individual sensor scans, followed by using a factor graph representation to express these submaps and their associated poses. The factor graph is then optimized for poses using both odometry links (for sequential poses) and loop closures (for non-sequential poses) using the iSAM optimization library [9].

1) Pose Graph Formation: For every time instant, there is a new pose estimate $\mathbf{x}_j$ and sensor scan available. The relative pose $u_{j,j+1}$ between two subsequent scans can be expressed as $u_{j,j+1} = \mathbf{x}_{j+1} \ominus \mathbf{x}_j$. This is seen in Fig. 4(a), where each pose $\mathbf{x}_j$ is represented as a node of the graph and odometry constraints $u_{j,j+1}$ are represented as edges connecting nodes. As we saw in Section III-B, we are interested in constructing a graph of submaps with nodes representing base poses of these submaps. This is illustrated in Fig. 4(b) which is formed by accumulating a finite set of sensor scans with each scan registered into a coordinate frame placed at the pose of the first scan (referred to as base pose $\mathbf{x}_i$ of submap m_i). Edges connecting base poses of submaps m_i and m_{i+1} are obtained by composition of odometry constraints of poses associated with submap m_i.

To add pairwise loop closure constraints between non-sequential poses $\{m_i, m_k\}$, we can utilize a scan matching technique to get a transformation o_{ik} that best aligns the two submaps. For our application, we convert occupancy grid submaps $\{m_i, m_k\}$ to point clouds and then use iterative closest point (ICP) algorithm to obtain transformation o_{ik} that best aligns the two submaps. For increased robustness to incorrect loop closures, we reject transformations o_{ik} that strongly disagree with expected transformations from the odometry chain. Additionally, we register only partial constraints (in non-degenerate directions) for submap pairs that give degenerate solutions for transformations o_{ik} during ICP registration. This projection of solution to non-degenerate directions is done based on the method described in [18].

2) Pose Graph Optimization: We can now perform *maximum a posteriori* (MAP) inference on resulting factor 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 [3]. 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

$$\mathcal{X}^{MAP} = \underset{\mathcal{X}}{\operatorname{argmin}} \left(\sum_{i=1}^{N} || f(\mathbf{x}_{i-1}, u_{i-1}) - \mathbf{x}_i ||^2_{\Lambda_i} + \sum_{(i,k)\in O} ||h(\mathbf{x}_i, \mathbf{x}_k) - o_{ik}||^2_{\Gamma_{ik}} + \sum_{(i,k)\in O'} ||\mathbf{V}_u h(\mathbf{x}_i, \mathbf{x}_k) - \mathbf{V}_u o_{ik}||^2_{\mathbf{V}_u \Gamma_{ik} \mathbf{V}_u^T} \right), \tag{6}$$

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 set of all tuples (i, k) for which a pairwise registration constraint o_{ik} exists between poses x_i and x_k with associated covariance matrix Γ_{ik}. O' is 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 3D ICP registration factor to only the well-conditioned directions as derived in [18]. We now solve this least squares optimization for sequence of unknown poses $\mathcal{X}$ using the iSAM optimization library [9]. Each time the SLAM optimization computes updated solutions for the sequence of submap base poses $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N\}$, the VOG-map $\mathcal{M}_{vog}$ implicitly gets updated since values for $\{t_i^{global}\}_{i=1}^{N}$ in Eq. 1 will also get updated.

B. Path Planning

Since the 3D environment that we are exploring is unknown beforehand, the planner must *actively map* the scene instead of relying on predefined waypoints. For actively mapping scenes with no prior information on the scene structure, a commonly employed approach is next-best-view (NBV) based path planning [1, 14, 16].

Next-best-view based planning approaches operate by greedily selecting the next-best robot pose (based on some information gain criteria) followed by executing a collision-free path to this pose. The VOG-map structure is compatible with any path planner that operates off an occupancy grid map. For purposes of our application, we utilize a sampling-based next-best-view planning approach briefly described below. More details can be found in [1].

1) View Sampling and Utility Computation: Starting at any current pose, the robot incrementally builds a geometric tree in configuration space using the RRT algorithm [11]. The RRT tree has a branching structure containing nodes and edges that terminate after a finite number of branches. Nodes represent robot configurations that are potential next-best future viewpoints, while edges connecting two nodes represent collision-free paths between the two configurations.

For each node n_k in the sampled RRT tree, a view utility value $\mathbf{Gain}(n_k)$ is computed as a summation of the unmapped volume that can be explored by traversing all nodes along that branch, that is,

$$\mathbf{Gain}(n_k) = \mathbf{Gain}(n_{k-1}) + \mathbf{Visible}(\mathcal{M}_{vog}, \xi_k) e^{-\lambda c(\sigma_{k-1}^k)}, \tag{7}$$

where n_{k-1} is the node previous to n_k along the same branch, $\mathcal{M}_{vog}$ is the VOG-map representation of the world, ξ_k is the robot configuration at node n_k, σ_{k-1}^k is the collision-free path between n_{k-1} and n_k and $c(\sigma_{k-1}^k)$ is the path traversal cost. Finally, $\mathbf{Visible}(\mathcal{M}_{vog}, \xi_k)$ is the total number of visible and unmapped voxels seen along all sensor rays cast out from robot configuration ξ_k at node n_k.

For computing unmapped voxel count $\mathbf{Visible}(\mathcal{M}_{vog}, \xi_k)$ as well for computing the collision-free path σ_{k-1}^k, the path planner needs to perform ray casting queries on the VOG-map using the method illustrated in Algorithm 1.

2) View Planning: Having computed a view utility value $\mathbf{Gain}(n_k)$ for every node n_k in the RRT tree, the best node n_{best} is selected as the node with highest gain, and the branch connecting current robot configuration to n_{best} is considered as the best branch. The robot now executes a path corresponding to the first segment of this best branch. Only the first segment of the best branch is executed since as the robot reaches and observes the world from a new node, its representation of the world map $\mathcal{M}_{vog}$ gets updated, which may lead to choosing of a different branch as the best branch. For computational efficiency, however, remainder of the best branch in the previous iteration is used to initialize the RRT tree in the next planning iteration.

As part of the receding horizon strategy, tree creation only happens until number of nodes reach a maximum value N_{max}. However, if $\mathbf{Gain}(n_{best})$ remains zero, tree construction is continued until number of nodes reach N_{tol}, where $N_{tol} >> N_{max}$. If the gain $\mathbf{Gain}(n_{best})$ in executing the best branch continues to remain zero, the active mapping task is considered solved and the algorithm terminates.

V. Simulated Experiments

We first evaluate our VOG-map based active mapping approach in a 3D simulated underwater environment being explored by a hovering autonomous underwater vehicle (HAUV). We compare the results of our approach against the next-best-view based active mapping algorithm in [1] that uses a single global occupancy map for planning.

A. Simulation Setup

For performing simulation-based evaluation, we require a closed-loop setup interfacing robot state estimation and low-level control with the path planning and SLAM algorithm. We hence adopt the *UUV Simulator* [12], 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. 8. For sensing, the simulated vehicle is equipped with a profiling sonar sensor that produces real-time 1D scan of its environment. A schematic of the geometry of a profiling sonar sensor can be seen in Fig. 6. 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 $\theta = 1^o$.

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 on the Bluefin HAUV. We hence simply simulate added gaussian noise for state estimates obtained in these three directions. In the x, y and yaw directions, the Bluefin HAUV uses IMU/DVL sensors that suffer from drift that grows unbounded with 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

$$\widehat{P}_{t+1} = \widehat{P}_t \oplus \left(P_{t+1} \ominus P_t \oplus \Delta t \mathcal{N}(0,\, \Sigma)\right), \tag{8}$$

where $\widehat{P}_1 = P_1$. In Eq. 8, 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

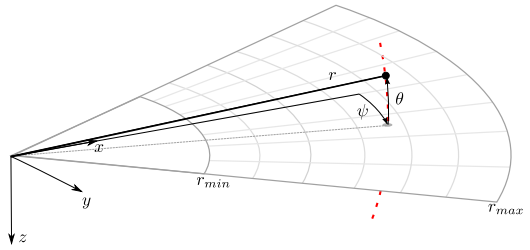


Fig. 6. Geometry of a single sonar scan.

TABLE I

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

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 $\sigma_\psi^2 = 10^{-7}\ rad^2/s$.

B. Dataset: Ship Propeller of SS Curtiss

The simulated environment consists of the propeller model of *SS Curtiss* as shown in Fig. 7. The size of the propeller is approximately 7m × 4m × 6m. The vehicle starts from the starboard side and navigates around the propeller for exploration and mapping within a bounding box of size 12m × 7m × 3.5m. Table I lists the dataset parameters that have been used. When evaluating our approach against the next-best-view planner in [1], 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.

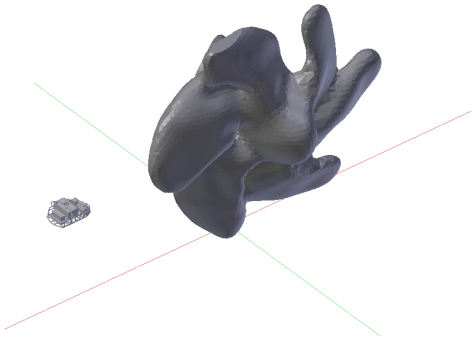


Fig. 7. Propeller of SS Curtiss and simulated HAUV.

TABLE II

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**

Fig. 5 shows the obtained 3D reconstructions registered using ICP against the ground truth 3D reconstruction for three different scenarios. Fig. 5(a) shows the 3D reconstructions obtained from the baseline next-best-view algorithm [1] operating using a standard single global occupancy map, Fig. 5(b) shows reconstructions using the VOG-map without loop closures and finally Fig. 5(c) shows reconstructions using the VOG-map with loop closures. It should be noted that the results in Fig. 5(b), Fig. 5(c) are from the same experimental run with the loop closure information not being used for mapping for Fig. 5(b). Fig. 5(a) is from a different experimental run. Table II 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. 5(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.

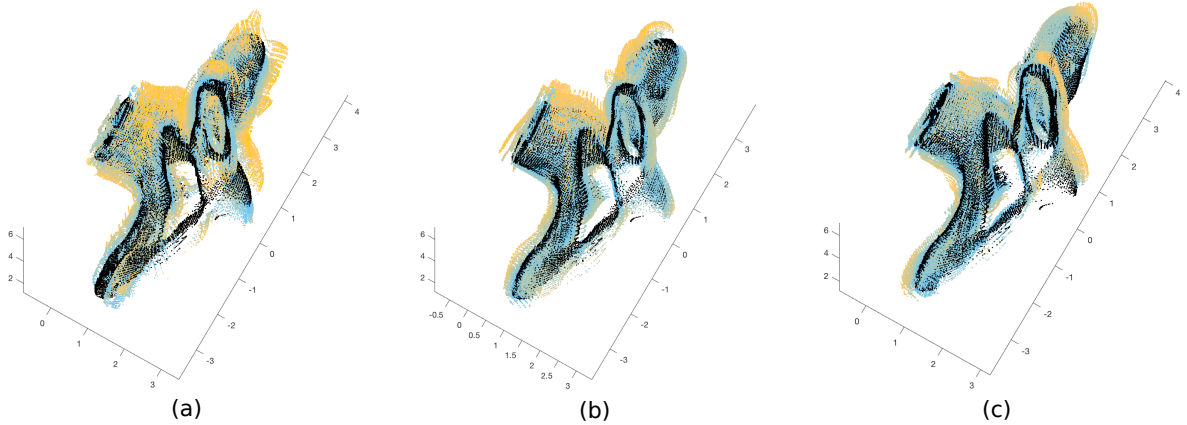


Fig. 5. 3D Reconstruction results registered using ICP against 3D reconstruction from ground truth odometry for three different scenarios. (a) shows reconstruction from baseline next-best-view algorithm [1] operating using a single global occupancy map, (b) shows reconstruction using VOG-map without 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.

VI. Real-world Experiments

We also conduct real-world experiments with the HAUV shown in Fig. 8 and compare the results of our VOG-map based active mapping approach against active mapping using a single global occupancy map as used in [1].

A. 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 Section V-A. We design a scenario as shown in Fig. 9 in which the HAUV starts in a position in an underwater tank where it can only see a limited portion of its environment. It has to now actively map the tank using the VOG-map to create a reconstruction of the scene that includes cylindrical tank walls, rectangular aluminum box, and the ladder. Table III lists the experiment parameters. The size of the tank is $7\text{m} \times 7\text{m} \times 3\text{m}$ 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.

B. Real-world Experiment Results

The navigation payload of the HAUV is quite accurate in this rather small contained environment. 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. 10 shows standard global occupancy map and VOG-map reconstructed from local submaps. We reconstruct the VOG-map from local submaps only for a visual comparison. Fig. 11 shows 3D scene reconstructions generated without and with loop closures. It can be seen that, due to loop closures, the reconstruction in Fig. 11(b) has corrected drift. Since the tank environment is small, the drift accumulation is not as significant as for the simulated propeller dataset.

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. 10, 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.

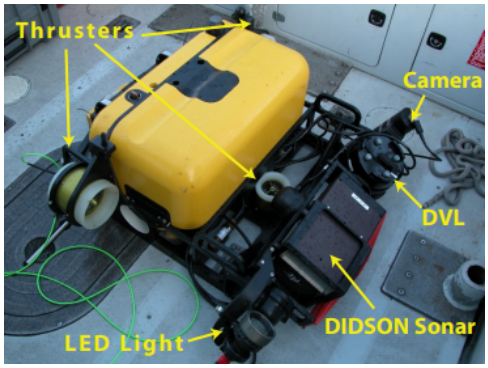


Fig. 8. Bluefin hovering autonomous underwater vehicle (HAUV).

C. Runtime Performance

The computation time for each location query in VOG-map by the planner is of the order $O(N) + N_s O(\log v_L)$ where N is total number of submaps up till current iteration, N_s is a subset of submaps returned from the ray intersection check as seen in Algorithm 1, and v_L is number of voxels in each local submap (assumed to be same for all submaps for simplicity). $O(N)$ represents time taken to do ray intersection check for N submaps which is typically a fast check as it computes intersection of ray with an axis aligned bounding box around each submap. $O(\log v_L)$ represents query time in each local octree having v_L number of voxels.

In contrast, a merging approach that generates a merged global map by transforming each local submap voxel into a common global coordinate frame and uses that merged map for planning will have a location query complexity of order $N_m O(v_L) + O(\log(N{*}v_L))$, where N_m is number of local submaps to be merged in current iteration. $O(v_L)$ is time taken for converting local submap voxel coordinates into global coordinates incorporated in the merged global map. $O(\log(N{*}v_L))$ is the query time in a merged global octree with $N{*}v_L$ voxels. An incremental merge strategy that merges only the most recent local submap into a previously merged global map will have $N_m = 1$ for all iterations except when there is a loop closure. For iterations with loop closures, $N_m = N$ since all submaps now need to be re-merged and that results in a very expensive operation.

When operating in the same environment for a long time, number of overlapping submaps would increase causing N_s in the $N_s O(\log v_L)$ term in the VOG-map query time to

TABLE III
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

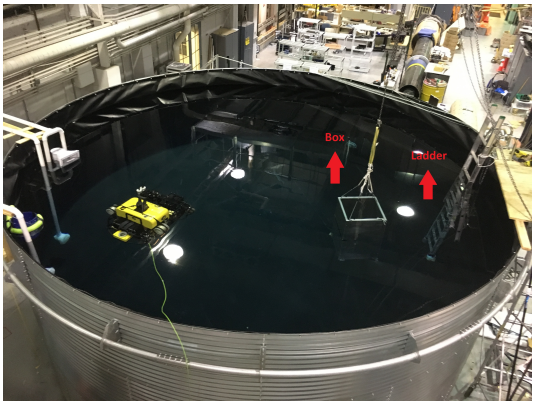
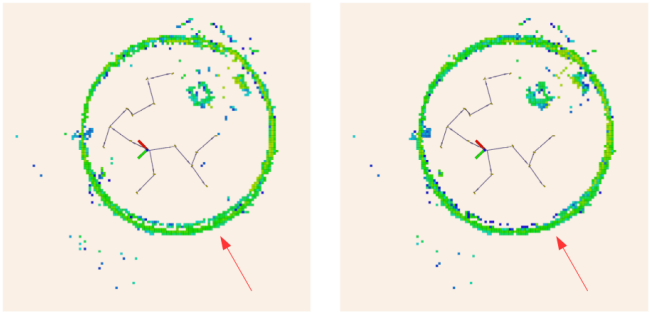
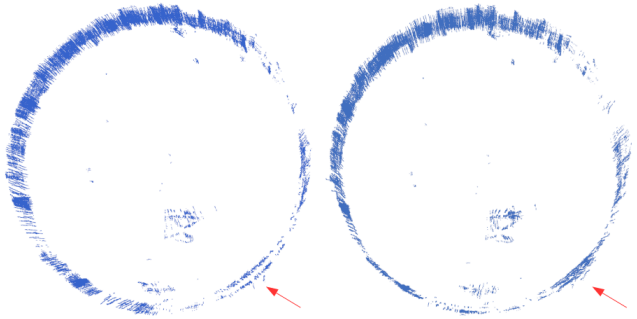


Fig. 9. Bluefin hovering autonomous underwater vehicle (HAUV) placed in an underwater tank with an aluminum box and a ladder.



(a) (b)

Fig. 10. **(a)** Top view of standard global occupancy grid. **(b)** Top view of VOG-map, reconstructed from local submaps for comparison. Yellow background shows extent of exploration bounding box. Regions indicated with red arrows show areas where loop closures using the VOG-map are able to correct accumulated drift.



(a) (b)

Fig. 11. 3D scene reconstructions: **(a)** Without loop closures. **(b)** With loop closures. Regions indicated with red arrows show areas where loop closures using VOG-map correct accumulated drift. Since tank environment is small, drift accumulation is not as significant as for the simulated propeller dataset.

dominate and rise linearly with number of submaps. This causes the planner runtime using VOG-map to increase with time. Currently our system planning using VOG-map in the underwater tank exhibits real-time behaviour until $\sim$75 submaps. This increasing runtime issue can be alleviated to some degree with the incremental merging strategy since rise of runtime with number of submaps there is logarithmic. However, it comes at the cost of having very expensive loop closure iterations where the term $N_m O(v_L)$ would have a large value. In that case, it is still possible to have the planner run in real-time by having it on a separate thread, but the planner would then plan based on an outdated map.

VII. Conclusions and Future Work

This paper presented the virtual occupancy grid map (VOG-map), a submap-based mapping approach, that can correct global map drift using loop closure constraints from the SLAM system while maintaining free space information for path planning. We demonstrated use of VOG-map based representation for an underwater SLAM system that actively plans paths in both simulated and real-world environments.

As future work, on the experimental side, we intend to test our approach for actively mapping unstructured underwater 3D environments out in the field. On the algorithmic side, we would firstly like to make our degeneracy-based loop closure checks more robust. The degeneracy check currently relies on a manually tuned threshold for determination of degenerate directions. Secondly, our current planning time increases with number of submaps if operating in the same environment for long times. One way to address this issue is to merge and restrict number of submaps. It is also possible to group overlapping local occupancy maps together for more efficient query. Thirdly, we would like to extend our active mapping approach to an active SLAM approach by taking into account minimization of localization uncertainties in computing information gain for future viewpoints.

References

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

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

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

[4] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision*. Springer, 2014, pp. 834–849.

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

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

[7] 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.

[8] M. Hsiao, E. Westman, G. Zhang, and M. Kaess, "Keyframe-based dense planar SLAM," in *Proc. International Conference on Robotics and Automation (ICRA), IEEE*, 2017.

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

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

[11] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[12] 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 *OCEANS 2016 MTS/IEEE Monterey*. IEEE, 2016, pp. 1–8.

[13] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, vol. 2. IEEE, 1985, pp. 116–121.

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

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

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

[17] R. Wagner, U. Frese, and B. 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*. IEEE, 2014, pp. 2691–2698.

[18] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 809–816.