2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Mapping-Guided Task Discovery and Allocation for Robotic Inspection of Underwater Structures

Marina Ruediger[1] and Ashis G. Banerjee[2]

***Abstract*— This paper introduces Mapping-based Tasks for Inspection: Discovery and Allocation (Map-TIDAL), a method for generating environmentally informed tasks and distributing them in a heterogeneous multi-robot system for visual inspection of underwater structures. Map-TIDAL leverages the individual robot maps generated during SLAM (without prior knowledge of the environment) and tasks from all the robots through a communication-aware auction process to determine additional inspection locations as the structures are further explored by the robots. This allows the method to adaptively focus on geometrically interesting areas that need detailed inspection while still maintaining good overall coverage with a reasonably small number of inspection tasks. Experiments on both saline and fresh water tanks show that Map-TIDAL yields better coverage while inspecting areas with interesting geometric features more thoroughly, using equal or fewer inspection locations compared to prevalent coverage methods using Voronoi distributions and boustrophedon patterns.**

I. Introduction

Autonomous inspections of hulls and interior spaces are important in assessing the safety and readiness of seagoing vessels and are used to detect corrosion, biofouling, cracks, coating failures, and other damages. A summary of the existing tools and methods for underwater inspections can be found in [1], [2]. However, many challenges persist due to high operational costs (expensive, highly customized platforms); huge scales of operations (detecting small defects in large vessels while operating in the open waters); and lack of system robustness and resilience (robot hardware and autonomous decision-making failures). Consequently, we consider a low-cost cooperative multi-robot system, and present a SLAM-based task discovery and allocation algorithm to generate the inspection tasks from environment meshes without prior geometric knowledge. This approach aims to address key underwater challenges: recovery from the loss or addition of perceptual and/or motor capabilities of robots; extremely slow and unreliable communications; and limited visibility due to turbidity and sparse features that complicate localization and mapping.

Our method is based on a decentralized multi-robot task allocation framework that uses minimal message sizes and intermittent communications [3]. Here, we extend this framework with an adaptive task discovery method that focuses the inspection effort on geometrically interesting areas identified during SLAM mesh generation. Specifically, we contribute a novel algorithm that automatically discovers inspection tasks from SLAM-generated meshes for heterogeneous multi-robot teams performing visual inspections of hulls or fluid-filled interior tanks. The inspection consists of visiting locations defined by position and orientation and capturing color images. Selected locations are referred to as tasks, while potential locations during discovery are candidate inspection points (CIPs).

Section 2 outlines relevant background information on underwater SLAM and task generation algorithms. Section 3 details our task discovery algorithm - Mapping-guided Tasks for Inspection: Discovery and ALLocation (Map-TIDAL), including the task discovery process and integration with task allocation. Section 4 presents the results of our in-water testing and comparisons with other task generation methods. In Section 5, we discuss the advantages and disadvantages, followed by our conclusions in Section 6.

II. Background

A. Underwater SLAM

Many advances have been made in the field of underwater SLAM, as summarized in [4], [5]. Due to the extremity of the underwater environment, many typical sensors used in aerial or ground SLAM solutions are unavailable or have limitations. Vision-based sensors that work well include both monocular [6] and stereo cameras, but care must be taken with shifts in color, water clarity [7], and image distortion [8]. Sonar systems are widely used for range finding-based SLAM [9]–[11], as acoustic waves have significantly lower attenuation rates in water and are not affected by changes in illumination conditions. However, they are often expensive. As an alternative, LiDAR systems have been used, although they work at different frequencies of light than their aerial counterparts [12].

Internal state-based sensors are often fused with external environmental sensors to overcome external sensing challenges. Doppler velocity logs (DVLs) use acoustic pulses to calculate velocity, compasses provide orientation data, inertial measurement units (IMUs) measure both linear and angular acceleration, and depth sensors calculate depth based on water pressure [4]. Often, multiple sensors are fused together to provide a robust solution. For example, [13], [14] combine visual, inertial, and sonar data; [15] uses visual, inertial and depth data; [16] couples DVL, visual, and inertial data; [17] fuses sonar, visual, inertial, and water-pressure data; and [18] presents a new dataset along with

*This work was supported in part by the ONR grant # N00014-21-1-2075.
[1]Marina Ruediger is with Department of Mechanical Engineering, University of Washington, Seattle, WA 98195, USA marina77@uw.edu
[2]Ashis G. Banerjee is with the Department of Industrial & Systems Engineering and Department of Mechanical Engineering, University of Washington, Seattle, WA 98195, USA ashisb@uw.edu

a solution encompassing stereo camera, IMU, DVL, and pressure sensor data.

B. Inspection Coverage

Multi-robot (or more generally multi-agent) coverage problems appear in a wide variety of domains, including inspection and surveillance, search and rescue, agriculture, manufacturing, traffic management, law enforcement, and more. Although many solutions have been proposed, typically, the coverage problem is addressed either through a detailed consideration of the choice of agents' viewpoints over a complex but well-known geometry or through the partitioning of a simple bounded region for a set of agents with known capabilities. For example, measurement uncertainties based on optical parameters are used to generate a set of viewpoints in [19]. An enhanced rapidly-exploring random tree is then used to compute the optimal subset of viewpoints that maintains full coverage and visibility of all the measurement points. A semiautomatic view planner is presented in [20], which uses 3D triangulation between a camera and a laser to actively calculate the viewpoints from an existing mesh model. A multi-robot boustrophedon coverage method for unknown 2D environments using line-of-sight communications among the explorer robots is developed in [21]. This method relies on cell splitting around obstacles to enable coverage using a heterogeneous system of a team of coverer robots and up to two teams of explorer robots. A sensor-based coverage path planning method that uses a generalized Voronoi diagram-based graph to model the energy capacity of multiple robots is presented in [22].

III. Method

A. Task Allocation

Map-TIDAL employs a decentralized task allocation framework based on the consensus-based bundle algorithm (CBBA) with domain-specific modifications for underwater multi-robot inspection, as detailed in our previous work [3]. The task allocation system operates through a distributed auction mechanism, where each robot maintains autonomous decision-making capabilities while coordinating with the team through minimal message exchanges — a critical requirement for bandwidth-constrained underwater communication environments based on the general method presented in [23]. We address the convergence behavior of the auctioning process and provide details of the messaging format to minimize the communication bandwidth in [3].

Each robot executes a local task manager responsible for maintaining: (1) known task inventories, (2) bid histories from neighboring robots, (3) communication timing logs, and (4) conflict resolution protocols. The auction process utilizes a reward function based on path optimization, where the reward for incorporating a given task into a robot's bundle is calculated as the difference in total path distance when the task is optimally inserted versus the original path configuration.

The bidding mechanism employs distance-based metrics, where each robot's bid represents the cumulative distance to reach that task location. This approach ensures convergence properties while providing computationally efficient effort estimation for task completion. Upon completion of the local auction phase, bid information is disseminated to neighboring robots for decentralized conflict resolution following the consensus protocols established in [23].

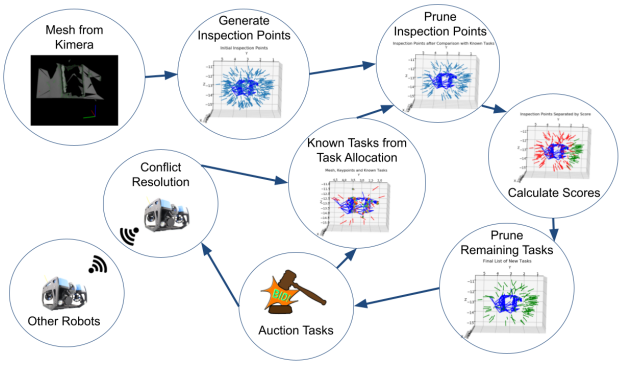

Fig. 1. Flowchart of the task discovery and task allocation processes.

The integration of task allocation with SLAM and task discovery processes in Map-TIDAL enables fully autonomous operation, where newly discovered tasks are seamlessly incorporated into the existing allocation framework and communicated efficiently throughout the robot network. Figure 1 shows the integration between the task allocation and task discovery steps in the pipeline from the SLAM keyframe mesh through the communication and conflict resolution stages. The consensus-based task sharing inherently creates a distributed understanding of spatial coverage across the inspection area. This shared awareness of existing task locations enables intelligent mesh analysis during task discovery, where newly generated candidate inspection points can be evaluated not only for their current viewpoint coverage but also for their contribution to overall coverage gaps identified through the collective task inventory.

B. Task Discovery

In this work, we use Kimera-Multi [24], [25] to fuse IMU, stereo vision, and DVL data. While originally intended for aerial and ground environments, Kimera-Multi is able to handle underwater sensor fusion. With properly calibrated cameras, Kimera-Multi calculates location and generates a lightweight 3D mesh from very sparse keypoint data using structural regularities [26]. This mesh is ideal for underwater inspection since visual features underwater are typically clustered only in areas with strong visual differences, such as patches of barnacles or algae, a scratch or rust damaged area, or a piece of complex geometry like a weld line or sea chest. These visual features are the exact things that need the most attention during inspections. In areas with large, smooth sections with minimal features, the robots will rely on the DVL and IMU for localization and the mesh will be less dense and will generate fewer tasks.

There are four main steps in the task discovery part of the Map-TIDAL process which begins whenever a keyframe

mesh ($\mathcal{M}$) is produced by Kimera.

1) **Candidate inspection point generation:** For each triangle in the mesh $\mathcal{M}$, candidate inspection points (CIPs) are generated by projecting along the surface normal by the ideal length (L_i), as shown in Fig. 2a. L_i is optimized based on camera focal length and water turbidity to maximize image quality while maintaining safe standoff distance from the vessel surface.
2) a) **Pruning based on existing tasks** (when $T_e \neq \emptyset$): Each CIP undergoes spatial filtering against existing tasks T_e using dual geometric constraints: radius of exclusion (r_{ex}) and angle of exclusion (θ_{ex}). A CIP is discarded if any existing task lies within both the spherical radius r_{ex} and the angular cone θ_{ex}, ensuring that retained points provide sufficiently distinct viewpoints for inspection coverage, as shown in Fig. 2b. If T_e contains only points outside of r_{ex} from any CIP, step 2b is performed instead.
   b) **Pruning based on self-comparison** (when $T_e = \emptyset$): When no prior tasks exist within r_{ex} of any CIP, CIPs undergo self-comparison using reduced thresholds ($p_s = 70\%$ of r_{ex} and θ_{ex}) to eliminate redundant inspection points within the candidate set, preventing over-dense task generation in highly featured areas.
3) **Keypoint scoring:** Each remaining CIP is evaluated by projecting the camera field-of-view (FOV) cone and counting intersected mesh keypoints shown in Fig. 2c. CIPs scoring above the threshold (t) percentage of maximum possible keypoints in the local mesh region are promoted to confirmed tasks, prioritizing geometrically complex areas that correlate with potential defects.
4) **Final spacial pruning:** Remaining sub-threshold CIPs undergo iterative comparison against the newly confirmed task set using the spatial exclusion criteria from Step 2a. Tasks are processed sequentially, with each addition expanding the exclusion zones for subsequent candidates, ensuring optimal spatial distribution in the final task set as shown in Fig. 2d.

Once the new task list has been finalized, it is sent to the auction phase of Map-TIDAL, which in turn sends the new task information to the other robots in conflict resolution, and updates the master list of tasks. The complete Map-TIDAL process is described in Algorithm 1 and Fig. 1.

IV. Experimental Results

A. Physical Setup and Data Collection

To validate Map-TIDAL, we collected two sets of in-situ data in an OSB test tank with saline water. We used a team of two BlueROV2 [27] robots customized with an NVIDIA Jetson Xavier AGX on-board processor, a WaterLinked A50 DVL (only on the robot of interest), and an OAK-D stereo depth camera. While the tether is used as a safety measure for monitoring and manual control takeover if required, all necessary processing and recording are done on the Jetson.

In the first test, a short video and IMU data with the robot manually held in a stable position were collected. This

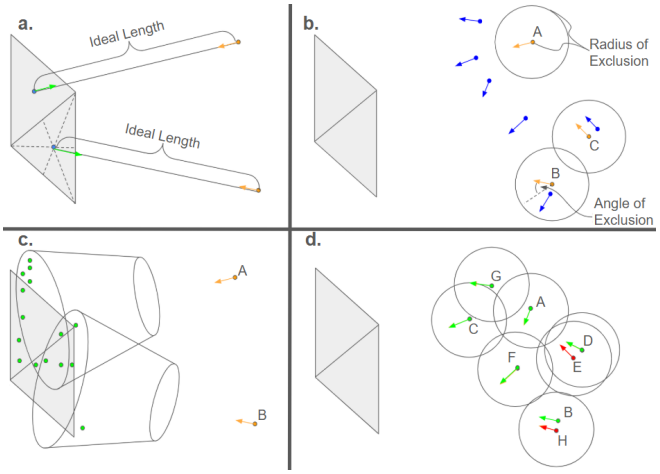

Fig. 2. **a.** Inspection point (orange) generation from mesh (grey) using the centroid (blue) and normal (green). **b.** Comparison of inspection points (orange) to existing tasks (blue), point A has no tasks inside the radius of exclusion, point B has a task in the radius of inclusion, but the angle is outside of the angle of exclusion, point C has a task that is within both the radius and angle of exclusion, and will not be removed. **c.** Calculation of keypoint scores based on the camera field of view. The score is the total of the keypoints (green) that fall in the field of view for the inspection point (orange) normalized by the number of keypoints. **d.** Final pruning on remaining inspection points, done in alphabetical order tasks E and H are removed, the remainder become tasks.

Algorithm 1 Task Discovery Process

Inputs: Kimera mesh ($\mathcal{M}$), existing task list (T_e)
Parameters: L_i, r_{ex}, θ_{ex}, t, FOV
Output: New task list (T_{new})

▷ $\hookleftarrow$ indicates appending to the end of a list
▷ $\emptyset$ indicates an empty list (ordered)

from $\mathcal{M}$ extract keypoints ($\mathcal{K}$) and polygons ($\mathcal{P}$)
$C \leftarrow$ GENERATE POINTS($\mathcal{P}$) ▷ step 1
if T_e is $\emptyset$ **then**
    $C \leftarrow$ PRUNE(C,$\emptyset$,2) ▷ step 2a
else
    $C \leftarrow$ PRUNE(C,T_e,0) ▷ step 2b
$T_{new} \leftarrow \emptyset$
$C_{remains} \leftarrow \emptyset$
$S \leftarrow$ CALCULATE SCORES($C, \mathcal{K}$) ▷ step 3
for i **in** $length(S)$ **do**
    if $S[i] \geq t$ **then**
        $T_{new} \hookleftarrow C[i]$
    else
        $C_{remains} \hookleftarrow C[i]$
$T_{new} \hookleftarrow$ PRUNE($C_{remains}, T_{new}, 1$) ▷ step 4
return T_{new}

Algorithm 2 Pruning

Parameters: L_i, r_{ex}, θ_{ex}, FOV, p_s

function PRUNE(C, C_{comp}, n)
  $precheck \leftarrow 1$ ▷ Boolean that marks if prior tasks exist within r_{ex}
  $C_{pruned} \leftarrow \emptyset$
  $C_{compare} \leftarrow \emptyset$
  if $n = 0$ **then** ▷ C is compared to only C_{comp}
    $C_{compare} \leftarrow C_{comp}$
  else if $n = 1$ **then** ▷ C is compared to C_{comp} w.r.t. own tasks
    $C_{pruned} \leftarrow C_{comp}$
    $C_{compare} \leftarrow C_{comp}$
  else if $n = 2$ **then** ▷ C is compared to itself
    $C_{pruned} \hookleftarrow C[0]$
    $C_{compare} \hookleftarrow C[0]$
    $r_{ex} \leftarrow r_{ex} \times p_s$ ▷ precheck strength
    $\theta_{ex} \leftarrow \theta_{ex} \times p_s$
  for X **in** C **do**
    $d \leftarrow \emptyset$ $\alpha \leftarrow \emptyset$
    $d \hookleftarrow distance(X, Y)$ **for** Y **in** $C_{compare}$
    $\alpha \hookleftarrow angle(X, Y)$ **for** Y **in** $C_{compare}$
    $keep \leftarrow 1$ ▷ Boolean indicating whether to keep a task
    for i **in** 0 to $length(d)$ **do**
      if $d[i] < r_{ex}$ **then**
        $precheck \leftarrow 0$
        if $\alpha[i] < \theta_{ex}$ **then**
          $keep \leftarrow 0$
          break
    if $keep = 1$ **then**
      $C_{pruned} \hookleftarrow X$
      if $n = 1$ **or** $n = 2$ **then**
        $C_{compare} \hookleftarrow X$
  if $n = 0$ **and** $precheck = 1$ **then**
    $C_{pruned} \leftarrow$ PRUNE(C, $\emptyset$, 2)
  return C_{pruned}

Algorithm 3 Additional Functions

function GENERATE POINTS($\mathcal{P}$)
  for x **in** $\mathcal{P}$ **do**
    $c \leftarrow$ centroid(x)
    $n \leftarrow$ normal(x)
    $p \leftarrow c + L_i \times n$
    $\theta \leftarrow -1 \times n$ ▷ angle expressed as a quaternion
    $C \hookleftarrow (p, \theta)$
  return C

function CALCULATE SCORES($C, \mathcal{K}$)
  $S \leftarrow \emptyset$
  for X **in** C **do**
    $FOV_t \leftarrow transform(FOV, X)$
    $T \leftarrow \sum (Delaunay(FOV_t).find\_simplex(\mathcal{K}) \geq 0)$
    $S \hookleftarrow T/length(\mathcal{K})$
  return S

video data was then used with offline Kimera processing to determine initial feasibility and algorithm development. Figure 3 shows the results of this initial algorithm development on a single mesh frame featuring 326 triangles. The initial parameters were chosen as $L_i = 1.5$ m, $r_{ex} = 0.8$ m, $\theta_{ex} = 0.1$ rad, and $t = 0.4$.The left image with keypoints and the mesh are shown in Fig. 3A and Fig. 3B, showing the concentration of the keypoints on the geometrically and visually interesting places such as the ladder feet and edges of the window and frame, and the two visible edges of the corner. Using these values and an initial list of 21 tasks selected randomly from the CIPs, the number of CIPs is reduced from 326 to 98. This can be further reduced with appropriate parameter tuning.

In the second test, the robot was allowed to slowly rise to a distance of about 0.5 m, and the first 7 keyframes were analyzed to find the ideal parameters for this environment. In pre-test observations, with high water clarity and good indoor lighting in the salt water tank, we found the OAK-D provided the most accurate point clouds when it was positioned approximately 1 meter from the observed surface; accordingly, this was set as the L_i value.

Parameters r_{ex}, θ_{ex}, and t were varied separately with the goals of minimal uninspected area, minimal overcoverage, and between 15-25 tasks. We observed that θ_{ex} produces the number of tasks in that range for values between 0.25 and 0.45 rad, r_{ex} produces tasks in that range between 1.5 and 2.0 m, with the rate of adding additional tasks higher with lower values of r_{ex} and θ_{ex}. The threshold has little effect on the rate of adding tasks, but a large effect on the initial number of tasks, with higher thresholds producing fewer tasks overall, and very little change in the number of tasks produced for any threshold variance over 80%.

B. Analysis

To analyze the effective coverage of the new task locations, we modeled the region of the test tank used during testing. This region includes a flat section with an inset window, two angled sections, and a short part of the longer sides of the tank, as shown in Fig. 4. The bottom of the tank and the surface of the water are not included in the model. The model is voxelized using Open3D libraries. For each point in the discovered task sets, the FOV cone is transformed, and any voxels within the cone are shaded red. If a voxel is seen from multiple positions, the shade of red is made brighter with each point that views it. By analyzing the distribution of the number of times each voxel is seen, we can compare the effectiveness of the variations of each parameter within their effective ranges.

Table I shows the average number of views per voxel, the number of tasks generated, and the percentage of voxels with zero views for various values of r_{ex}, θ_{ex} and t. Ideal values with less than 5% uninspected are shown in green, and for

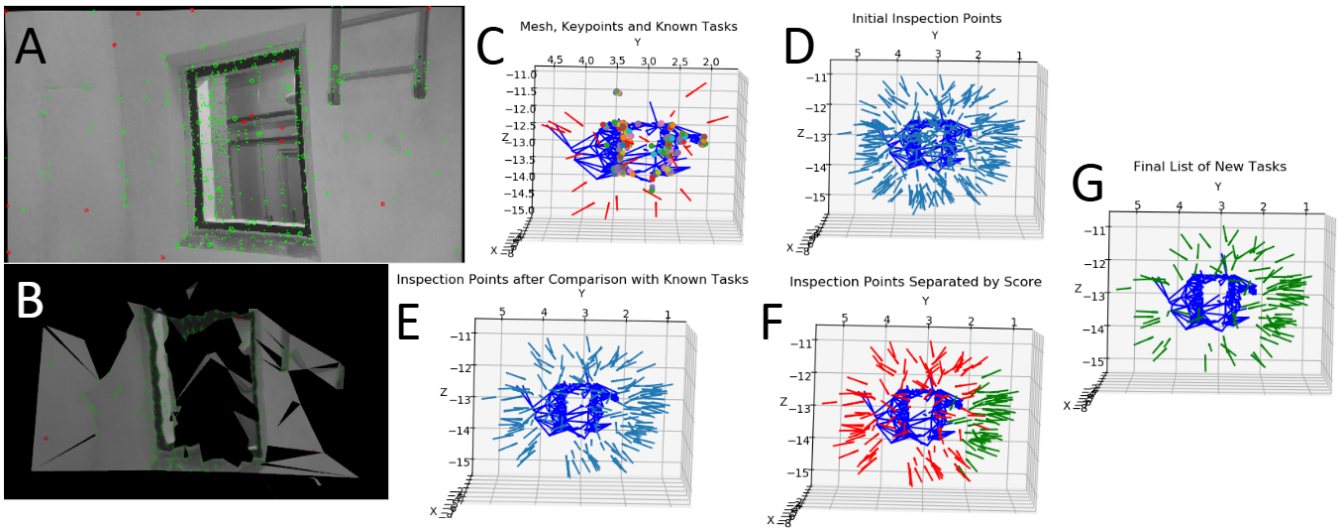

Fig. 3. **A.** Left camera image with tracked keypoints overlaid in green and untracked keypoints in red. **B.** Kimera mesh with A. as texture. **C.** Initial tasks shown as arrows in red, mesh in blue, keypoints as rainbow dots. **D.** The 326 CIPs generated in teal. **E.** The 165 CIPs remaining after the first pruning in teal. **F.** The 49 tasks kept from their keypoint scores in green, with the remaining 116 CIPs in red. **G.** The 98 tasks kept after the final pruning.

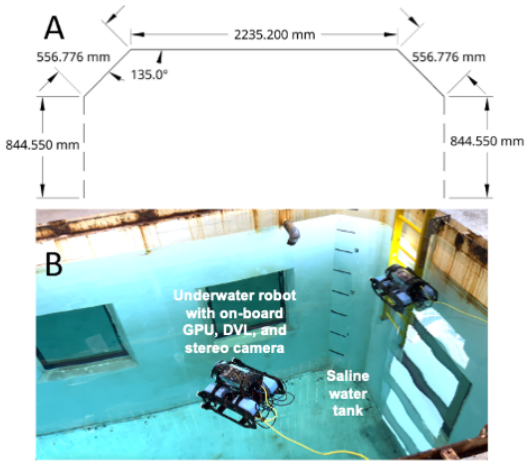

Fig. 4. **A.** Top view of the test tank with dimensions. Solid lines show geometry used to generate inspection points for Voronoi and sweeping pattern methods, dashed lines show the edges of the voxelized region used to evaluate method effectiveness. **B.** Test tank with two modified BlueROV2s with the main robot at the center.

TABLE I
AVERAGE VIEWS PER VOXEL, NUMBER OF TASKS, AND PERCENT UNINSPECTED

t	r_{ex} (m)	θ_{ex} (rad)					
		0.25			0.3		
		avg	num	%	avg	num	%
70%	1.5	4.26	27	1.49	3.97	25	3.71
	1.6	3.63	24	7.60	3.19	21	8.06
	1.7	2.85	22	4.28	2.30	18	21.16
	1.8	2.71	19	6.18	2.79	19	5.77
75%	1.5	4.05	26	3.05	3.50	23	5.69
	1.6	3.12	22	11.73	2.83	20	9.53
	1.7	2.56	20	3.81	2.38	18	6.79
	1.8	2.48	18	6.84	2.55	18	6.42
80%	1.5	3.26	23	4.06	3.06	21	4.40
	1.6	2.62	22	11.90	2.31	19	8.07
	1.7	4.06	25	3.71	3.15	21	5.13
	1.8	3.78	23	6.35	3.09	20	6.45

Avg indicates the average views per voxel, num the number of tasks generated with that set of parameters, and % the percentage of voxels that receive no views. Green values indicate optimal performance, yellow values acceptable performance, and red values indicate significant overcoverage.

these values, between 15 and 25 tasks, and an average of less than 2.5 views per voxel are also shown in green. Yellow values indicate values outside of that range, but under 30 tasks or with an average lower than 3.5, while red values indicate averages over 3.5. The best coverage is achieved at $r_{ex} = 1.7\ m$, $\theta_{ex} = 0.25\ rad$, and $t = 75\%$ but other values also show acceptable results. In general, lower r_{ex} and θ_{ex} values lead to higher numbers of tasks with better coverage, but a higher average coverage.

C. Comparison Metrics

Metrics on inspection quality such as the coverage percent and averages can be easily compared, but finding a metric that also shows the distribution of overcoverage can be a challenge. Some overcoverage can be desired, especially around complex geometry to ensure that all angles are viewed to reveal all defects, but too much overcoverage is a sign that there are more inspection points than needed-leading to wasted time and energy. But what is the ideal balance that ensures coverage without too much overcoverage? While many options are present, we have chosen to examine the distribution of number of views per voxel compared to the gamma distribution. While this function is typically used for wait time variables, the inherent 0 minimum, left skew, and exponentially decreasing tail make the gamma distribution good for our purposes as well. The gamma distribution has the probability density function in (1).

$$f(x;\alpha,\beta) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} e^{-x/\beta} \tag{1}$$

By choosing the shape parameter α and the scale parameter β, we can modify the shape of the distribution to fit our desired inspection requirements of having a minimal number of voxels inspected 0 times, a peak inspection value between 1 and 2, and a low number of voxels with a large degree of overcoverage. The peak value of the gamma distribution is at $\beta(\alpha - 1)$. Using 2 as the peak value and setting $\alpha = 4$, gives $\beta = \frac{2}{3}$. This choice of parameters gives less than 1% of uninspected voxels fewer than 0.5 times. Since the gamma distribution is continuous while the number of views is discrete, this cut-off of 1% is used while discretizing the gamma distribution data using integer centered bins.

D. Comparisons

We compare Map-TIDAl with two prevalent coverage methods, namely Voronoi diagrams and boustrophedon patterns. Voronoi diagrams partition the covered space into regions that contain all the points closest to a point within a neighborhood. A survey of Voronoi methods for robotic coverage is provided in [28]. We specifically employ the WebSVG's Voronoi editor [29], which uses a seed generation method that controls the spread and regularity of the partitions, and allows weighted distributions and wall avoidance. Boustrophedon patterns are based on sweeping linearly back and forth over a space. Our implementation is based on [30], where optimized paths are calculated using a generalized traveling salesman problem formulation, which can be modified during runtime with additional no-fly zones.

For both the Voronoi and sweeping patterns, we use a projection of the rectangular surface onto the wall of the tank with a distance offset of 1 m and an angle perpendicular to the surface of the tank. A top-down section of the tank portion considered is shown in Fig. 4. Fig. 5 shows the best version of each generation method and Fig. 6 shows a histogram chart with the views per voxel overlaid with the ideal gamma based distribution generated in Section IV.C. The parameters of the Boustrophedon and Voronoi methods are chosen to be comparable in the number of tasks to those generated through Map-TIDAL, and as close to the ideal distribution as possible. The boustrophedon pattern is generated using two interpolation points between sections longer than 1 m, with a lateral overlap of 0.1 m. The Voronoi partition is generated with 20 points including wall avoidance. Map-TIDAL uses $r_{ex} = 1.7$ m, $\theta_{ex} = 0.25$ rad, and $t = 75\%$.

From the histogram in Fig. 6, it can be seen that Map-TIDAL achieves 3.81% uninspected voxels with an average of 2.56 views per voxel in 20 tasks as compared to the Voronoi (17.32% uninspected, 1.39 average views, 20 tasks) and boustrophedon (8.94% uninspected, 1.80 average views, 30 tasks). The average of absolute differences between Map-TIDAL and the ideal distribution is 0.029, while the Voronoi (0.100) and boustrophedon (0.081) display larger differences.

E. Additional Experimental Validation

The third test was conducted in a fresh water pool with no additional parameter tuning, as the pool visibility was similar to that in the OSB test tank. This test involved more movement of the robot, with a surface not directly perpendicular to the robot, similar to what it would see if moving along an underwater vessel hull at an angle. Figure 7 shows the results of the first and second mesh frames in one testing instance. Based on observations made during testing, The set of inspection tasks appears well distributed over the environment, even though the Kimera mesh visualization is not complete. The later mesh frames (not shown) indicate that wall reflection on the surface of the water causes additional mesh irregularities, implying additional visual processing should be developed to handle regions near the surface. Overall, this test demonstrates the usefulness of the Map-TIDAL method in different environments with varying geometric complexities.

Quantitatively, using a similar voxelization method to that in the saline water tank, we note that Map-TIDAL yields the most efficient and effective coverage with only 1.9% uninspected voxels and 2.04 average views per voxel based on just 12 task locations. On the other hand, the boustrophedon method generates 28 task locations for 0.0% uninspected voxels and 1.66 average views, and the Voronoi method produces 24 task locations for 0.9% uninspected voxels and 1.39 average views. These results reinforce the correlation between regions with interesting geometry and keypoint scores to prune tasks in the Map-TIDAL method. Accordingly, it focuses on inspecting the region near the intersection of the two walls, bulkhead, and slanted pool floor, as shown in Fig. 8.

V. Discussion

Qualitative analysis of the brightest red areas in Fig. 5B shows a clear concentration of inspection views on the center and upper right side of the tank using the Map-TIDAL method. This corresponds to the position of the safety ladder during testing. Both the sweeping and Voronoi patterns lack this ability to concentrate the tasks around the most interesting sections (geometry) of the inspected structures. However, they can potentially alleviate this problem by incorporating weighted distributions during task generation. Their main limitation is that they require a model in order to generate the inspection points. The sweeping pattern tool is able to recalculate quickly on the fly when a user adjusts the model of the space, but still requires user input, preventing the system from being fully autonomous. The Voronoi patterns also calculate quickly, but require the geometric limits of the space before starting. Changing the environment for either case would overwrite the existing tasks with a new set, which would be challenging, since underwater communication is slow and unreliable. This limits the effectiveness of both methods in dynamic environments with minimal a priori knowledge of the geometry.

The proposed Map-TIDAL method is not without limitations. Since task discovery is based on the meshes generated during SLAM, it does not require prior knowledge of the geometry to be inspected. However, the quality of the generated tasks will depend heavily on the quality of the

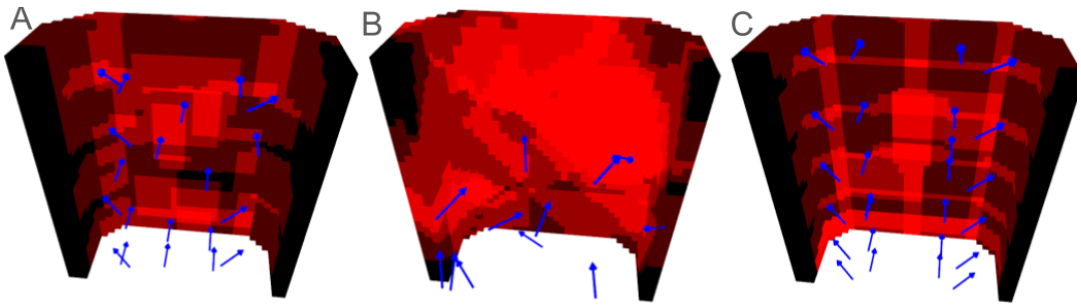

Fig. 5. Simulated coverage analysis for **A** Voronoi, **B** Map-TIDAL, and **C** sweeping patterns with task locations shown in blue and voxels colored more red with each time they are viewed.

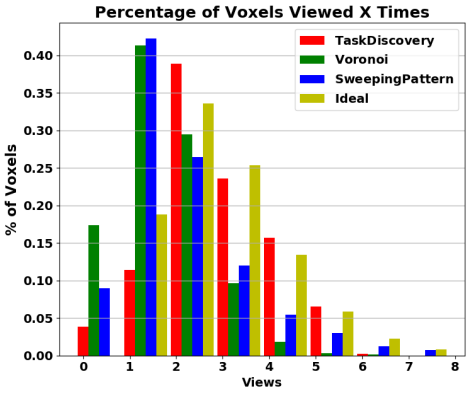

Fig. 6. Histogram showing normalized views per voxel for different task generation methods compared with the discretized ideal gamma distribution.

SLAM meshes. In murky water, additional sensors, such as the DVL, are necessary to maintain localization when no objects are visible, since any errors in localization result in corresponding errors in task positions. In addition, it is possible that the generated tasks may be placed in areas the robots cannot access due to the fact that only the map viewed by the individual robot is available during task discovery. It would be useful to address this accessibility problem in the future through motion planning.

VI. Conclusions

Here, we present a new algorithm, Map-TIDAL, for online task discovery from a SLAM generated environment mesh through candidate inspection points generation and pruning based on their keypoint scores and proximity to other tasks. These tasks are efficiently allocated to a multi-robot system through a decentralized auctioning and conflict resolution process. As new geometry is discovered (explored), the corresponding tasks are distributed, avoiding the need to maintain models of the inspected structures and focusing the inspection on areas with a large number of visually distinct features. With suitable parameter adjustment and accurate SLAM, Map-TIDAL produces task locations that provide excellent coverage without causing overinspection.

Future works will include investigating collision avoidance during SLAM-based task discovery and allocation with a larger system of robots. Another potential direction is multi-modal sensor fusion, where fusion of camera images with side scan sonar data could augment localization and map generation abilities. Side scan sonar data would also provide more accurate meshes for Map-TIDAL's task discovery process. Although Map-TIDAL currently focuses on visual inspection data, the algorithm could be extended for other inspection sensors, such as ultrasonic thickness testers, or for the use of additional tools, such as brushes, scrapers, and probes. Map-TIDAL is designed for underwater use, but has potential for use in challenging aerial environments with limited communications and visual range if adapted for a system of flying drones.

Acknowledgment

We would like to thank Wade Kempf, Tyler Paine, Matthew Craw, and Martin Renken for their mentorship and continued support of our research. We acknowledge the use of NIPRGPT in text editing.

References

[1] B. Lin and X. Dong, "Ship hull inspection: A survey," *Ocean Eng.*, vol. 289, p. 116281, 2023.

[2] R. Eckholdt Andersen, R. Yding Brogaard, and E. Boukas, "Remote inspection techniques: A review of autonomous robotic inspection for marine vessels," *IEEE Trans. Field Robot.*, vol. 2, pp. 1–20, 2025.

[3] M. Ruediger, T. M. Paine, and A. G. Banerjee, "Simulation of underwater environments to investigate multi-robot systems for marine hull inspection," in *IEEE OCEANS, Hampton Roads*, pp. 1–7, 2022.

[4] X. Wang, X. Fan, P. Shi, J. Ni, and Z. Zhou, "An overview of key SLAM technologies for underwater scenes," *Remote Sensing*, vol. 15, no. 10, p. 2496, 2023.

[5] M. Heshmat, L. Saad Saoud, M. Abujabal, A. Sultan, M. Elmezain, L. Seneviratne, and I. Hussain, "Underwater SLAM meets deep learning: Challenges, multi-sensor integration, and future directions," *Sensors*, vol. 25, no. 11, p. 3258, 2025.

[6] F. Hidalgo, C. Kahlefendt, and T. Bräunl, "Monocular ORB-SLAM application in underwater scenarios," in *OCEANS - MTS/IEEE*, pp. 1–4, 2018.

[7] L. M. Hodne, E. Leikvoll, M. Yip, A. L. Teigen, A. Stahl, and R. Mester, "Detecting and suppressing marine snow for underwater visual SLAM," in *IEEE/CVF Conf. Comp. Vis. Pattern Recognit. Workshops*, pp. 5097–5105, 2022.

[8] T. Łuczyński, M. Pfingsthorn, and A. Birk, "The pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings," *Ocean Eng.*, vol. 133, pp. 9–22, 2017.

[9] E. Westman, A. Hinduja, and M. Kaess, "Feature-based SLAM for imaging sonar with under-constrained landmarks," in *IEEE Int. Conf. Robot. Autom.*, pp. 3629–3636, 2018.

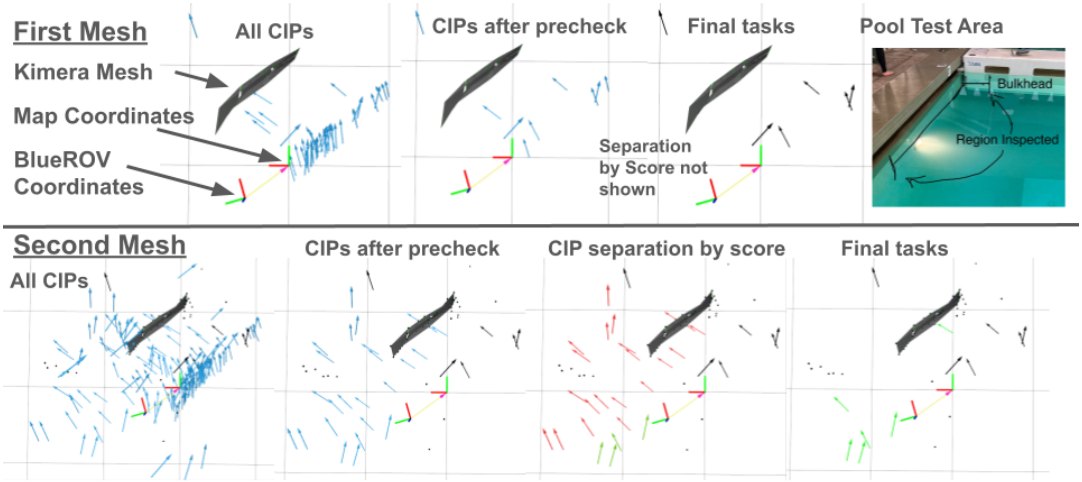

Fig. 7. Task discovery with steps shown over the first two mesh frames in pool testing. CIPs are colored blue initially. In the case of the first mesh, the final tasks are shown in black, which remain when they are used in the second mesh as the existing tasks. For the second mesh, during the score separation step, ones that are selected are green, while the ones subject to final pruning are red, and the final selection is shown in light green. These light green tasks are then combined with the existing list of tasks for the next mesh frame.

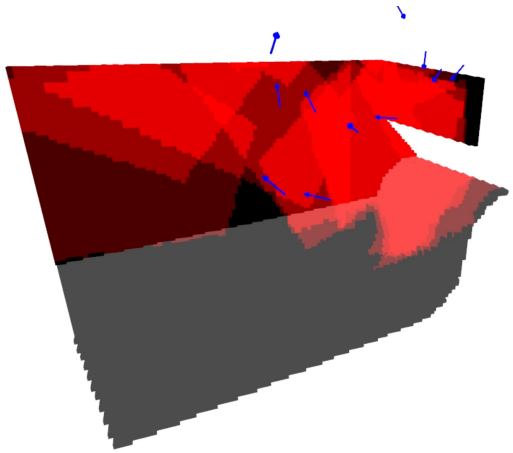

Fig. 8. Voxelized model of fresh water pool with the tasks generated by Map-TIDAL. Red areas show the projected field of view for each task location; gray surfaces are only presented for visualization purposes.

[10] J. Wang, F. Chen, Y. Huang, J. McConnell, T. Shan, and B. Englot, "Virtual maps for autonomous exploration of cluttered underwater environments," *IEEE J. Oceanic Eng.*, vol. 47, no. 4, pp. 916–935, 2022.

[11] N. Loi, Y. Z. Tan, E. W. Goh, and M. H. Ang, "Sonar SLAM in structured underwater environments," in *OCEANS - Singapore*, pp. 1–10, 2024.

[12] D. C. Estrada, F. R. Dalgleish, C. J. D. Ouden, B. Ramos, Y. Li, and B. Ouyang, "Underwater LiDAR image enhancement using a GAN based machine learning technique," *IEEE Sensors J.*, vol. 22, no. 5, pp. 4438–4451, 2022.

[13] J. Zhang, F. Han, D. Han, J. Yang, W. Zhao, and H. Li, "Integration of sonar and visual–inertial systems for SLAM in underwater environments," *IEEE Sensors J.*, vol. 24, no. 10, pp. 16792–16804, 2024.

[14] S. Pan, Z. Hong, Z. Hu, X. Xu, W. Lu, and L. Hu, "RUSSO: Robust underwater SLAM with sonar optimization against visual degradation," *IEEE/ASME Trans. Mechatron.*, pp. 1–12, 2025.

[15] S. Ding, T. Zhang, Y. Li, S. Xu, and M. Lei, "Underwater multi-sensor fusion localization with visual-inertial-depth using hybrid residuals and efficient loop closing," *Measurement*, vol. 238, p. 115245, 2024.

[16] S. Xu, K. Zhang, and S. Wang, "AQUA-SLAM: Tightly coupled underwater acoustic-visual-inertial SLAM with sensor calibration," *IEEE Trans. Robot.*, vol. 41, pp. 2785–2803, 2025.

[17] S. Rahman, A. Quattrini Li, and I. Rekleitis, "SVIn2: A multi-sensor fusion-based underwater SLAM system," *Int. J. Robot. Res.*, vol. 41, no. 11-12, pp. 1022–1042, 2022.

[18] S. Xu, J. Scharff Willners, J. Roe, S. Katagiri, T. Luczynski, Y. Petillot, and S. Wang, "Tank dataset: An underwater multi-sensor dataset for SLAM evaluation," *Int. J. Robot. Res.*, p. 02783649251364904, 2025.

[19] Y. Liu, W. Zhao, H. Liu, Y. Wang, and X. Yue, "Coverage path planning for robotic quality inspection with control on measurement uncertainty," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 5, pp. 3482–3493, 2022.

[20] A. Mavrinac, X. Chen, and J. L. Alarcon-Herrera, "Semiautomatic model-based view planning for active triangulation 3-D inspection systems," *IEEE/ASME Trans. Mechatron.*, vol. 20, no. 2, pp. 799–811, 2015.

[21] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient boustrophedon multi-robot coverage: An algorithmic approach," *Ann. Math. Artif. Intell.*, vol. 52, pp. 109–142, 2008.

[22] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu, "A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints," *IEEE Trans. Cybernet.*, vol. 44, no. 3, pp. 305–314, 2014.

[23] L. Brunet, H.-L. Choi, and J. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, 2009.

[24] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping," in *IEEE Intl. Conf. Robot. Autom.*, pp. 1689–1696, 2020.

[25] Y. Tian, Y. Chang, F. H. Arias, C. Nieto-Granda, J. P. How, and L. Carlone, "Kimera-Multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2022–2038, 2022.

[26] A. Rosinol, T. Sattler, M. Pollefeys, and L. Carlone, "Incremental Visual-Inertial 3D Mesh Generation with Structural Regularities," in *IEEE Int. Conf. Robot. Autom.*, pp. 8220–8226, 2019.

[27] "BlueROV2." `https://bluerobotics.com/store/rov/bluerov2/`, Feb 2022.

[28] M. Zhou, J. Li, C. Wang, J. Wang, and L. Wang, "Applications of Voronoi diagrams in multi-robot coverage: A review," *J. Marine Sci. Eng.*, vol. 12, no. 6: 1022, 2024.

[29] WebSVG, "Voronoi editor." https://github.com/WebSVG/voronoi.

[30] R. Bähnemann, N. Lawrance, J. J. Chung, M. Pantic, R. Siegwart, and J. Nieto, "Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem," in *Field Service Robot.: Results 12th Int. Conf.*, pp. 277–290, Springer, 2021.