Received 31 October 2024; revised 20 April, 2025; accepted 2 June, 2025; date of current version 2 June, 2025.

Semantics-aware Predictive Inspection Path Planning

Mihir Dharmadhikari[1] and Kostas Alexis[1]

[1]Norwegian University of Science and Technology (NTNU), O. S. Bragstads Plass 2D, 7034, Trondheim, Norway

Corresponding author: Mihir Dharmadhikari (email: mihir.dharmadhikari@ntnu.no)

This material was supported by the Research Council of Norway under project SENTIENT (NO-321435) and the European Commission through the project AUTOASSESS under the Horizon Europe Grant (101120732).

ABSTRACT This paper presents a novel semantics-aware inspection path planning paradigm called "Semantics-aware Predictive Planning" (SPP). Industrial environments that require the inspection of specific objects or structures (called "semantics"), such as ballast water tanks inside ships, often present structured and repetitive spatial arrangements of the semantics of interest. Motivated by this, we first contribute an algorithm that identifies spatially repeating patterns of semantics - exact or inexact - in a semantic scene graph representation and makes predictions about the evolution of the graph in the unseen parts of the environment using these patterns. Furthermore, two inspection path planning strategies, tailored to ballast water tank inspection, that exploit these predictions are proposed. To assess the performance of the novel predictive planning paradigm, both simulation and experimental evaluations are performed. First, we conduct a simulation study comparing the method against relevant state-of-the-art techniques and further present tests showing its ability to handle imperfect patterns. Second, we deploy our method onboard a collision-tolerant aerial robot operating inside the ballast tanks of two real ships. The results, both in simulation and field experiments, demonstrate significant improvement over the state-of-the-art in terms of inspection time while maintaining equal or better semantic surface coverage. A set of videos describing the different parts of the method and the field deployments are available at https://tinyurl.com/spp-videos. The code for this work is made available at https://github.com/ntnu-arl/predictive_planning_ros.

INDEX TERMS

I. INTRODUCTION

In recent years, a spectrum of contributions has been made in the domain of autonomous robotic exploration, mapping, and inspection path planning [1], [2], [3], [4]. Accelerated by this research, robotic systems have been successfully deployed in a variety of environments including structured industrial settings such as those found in the oil & gas industry [5], [6], [7], [8], [9], [3], or unstructured natural scenes [10] such as subterranean settings [11], [12], [13], [1], [14], [2], [15], [16]. As more robots get deployed in human-made environments, more complex tasks, such as object search and inspection of specific structures, emerge. We refer to the objects and structures relevant to a mission as "semantics". These new tasks require the robot to have an object-level "semantic" understanding of its surroundings and to be able to take action accordingly. As a result, the robotics community has seen an increased interest in metric-semantic representations of the environments [17], [18], [19], [20] and semantics-aware path planning [21], [22], [23], [24], [25].

Despite the increased interest and the progress in semantic scene reasoning [26], [27], [28], the majority of the current research in semantics-aware path planning either treats the semantics individually - without accounting for the relations between them - or only exploits the relations between the semantics seen so far [22], [29], [30]. However, such approaches do not account for the fact that especially in industrial environments, semantics of essential importance for inspection are not distributed arbitrarily but instead present highly structured relationships among them. In particular, facilities like ship ballast water tanks present spatially repeating patterns of the objects of interest (e.g., Figure 1). This opens up the avenue to exploit the repeatable nature of semantics to make predictions about the unseen parts of the environment during an inspection mission and thus greatly improve the efficiency and systematicity of such tasks.

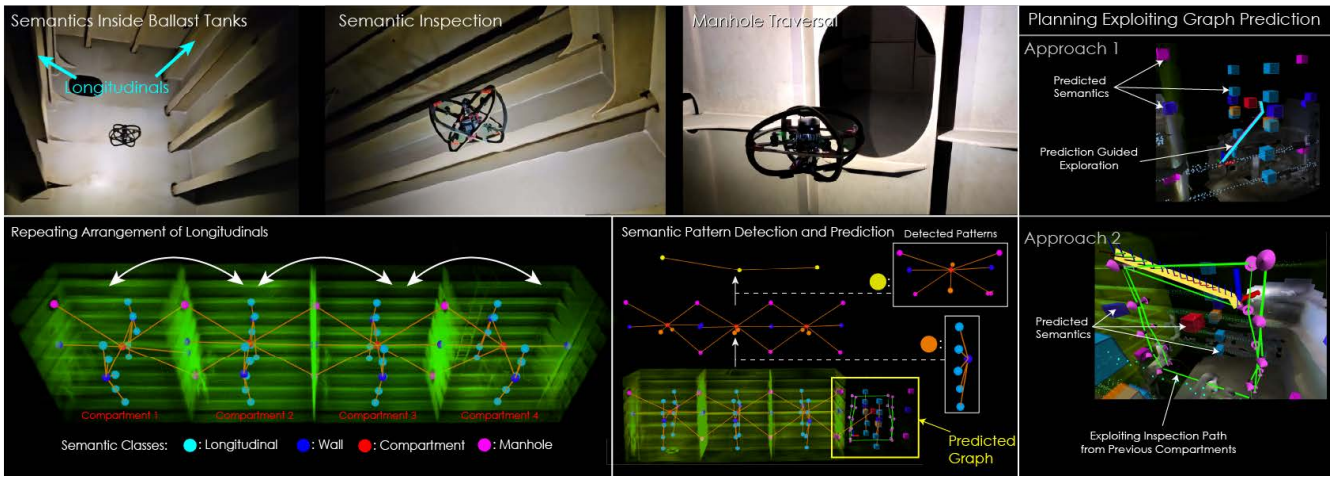


FIGURE 1. Instances of our collision-tolerant aerial robot inspecting and navigating in ballast tank environments using the proposed Semantics-aware Predictive Planner. Additionally, the semantics of interest inside these environments are shown highlighting their spatially repeating arrangement. The bottom center sub-figure shows an instance of pattern detection and graph prediction in the Semantic Scene Graph (SSG). It is on this prediction of semantics that predictive planning, shown in the right two subfigures, takes place.

Motivated by the above, this paper presents a novel semantic inspection paradigm called, "Semantics-aware Predictive Planning" (SPP), for inspection of the semantics of interest in environments where repeatable object patterns exist. The proposed method uses a dual environment representation consisting of a) a volumetric map capturing the geometry [31] and b) a Semantic Scene Graph (SSG) [32], [33], [34]. Using this representation, first, an algorithm to identify repeating patterns of semantics in the SSG is presented. Second, a strategy to predict the evolution of the SSG using the detected patterns is described. Finally, two inspection path planning strategies, tailored to the inspection of semantics inside ballast water tanks, are proposed that exploit these predictions towards superior mission efficiency. The source code of this paper is made available at: https://github.com/ntnu-arl/predictive_planning_ros.

In the simulation studies, the proposed planner is tested and compared against state-of-the-art exploration and inspection path planning methods as benchmarks in a model of a large ballast tank and a factory. The study demonstrates a significant improvement (ranging from 25% to 60% depending on the method compared against) in the inspection time with the use of the proposed strategies. Additionally, tests are conducted in a model of a ballast tank containing imperfect patterns to demonstrate the ability of the proposed method to handle the case of missing semantics successfully.

Finally, the paper presents results from field deployments in the ballast tanks of two oil tanker ships, with the method integrated onboard a collision-tolerant aerial robot [35]. In both ships, the two proposed planning approaches are tested and compared. The robot is successfully able to navigate in the ballast tanks in all the missions, build the SSG, and complete the inspection of the semantics important for inspection. The proposed planning approaches show an improvement of up to 23% in the inspection time compared to a semantics-aware "Baseline" that is not exploiting prediction thus demonstrating the real-world applicability of the method. Significantly improving the efficiency of such inspection missions is essential, especially considering the scale of such environments (often involving multiple ballast compartments across levels) and the limited flight time of the aerial robots that can be used in such deployments given the narrow spaces that they have to navigate through.

The remainder of this paper is organized as follows: Section II outlines related work, followed by the problem formulation in Section III. The proposed approach for SSG pattern prediction and predictive planning is detailed in Sections IV and V respectively. Simulation studies are shown in Section VI, while results from field tests are presented in Section VII. Finally, conclusions are drawn in Section VIII.

II. RELATED WORK

In recent years, there has been a boost in the research on semantics perception. Several works have investigated the problem of semantic segmentation [36], [37], [38], [39] and their representation for planning in the form of an SSG or metric-semantic maps [17], [18], [19], [20]. The authors of [40] propose a framework for fine-grained semantic mapping that supports open-set object queries both within and beyond the depth-sensing range, by storing semantic information within range using a voxel representation and representing information beyond range as rays. [17], [18] present a hierarchical semantic scene graph representation that not only captures the objects and their relations in the scene graph but also creates higher-level abstractions by grouping them into predefined structures such as places, rooms, and buildings. The authors of [19] propose an incremental 3D scene graph generation method using a sequence of RGBD sensor frames. [20] presents a multi-resolution metric-semantic Truncated Signed Distance Field (TSDF)

representation that allocates high-resolution submaps for semantic objects and can achieve object-level consistency over long time horizons. [41] presents an actionable hierarchical scene representation where the levels are guided by the proposed inspection and exploration planner. Despite the progress, these methods do not provide the possibility to detect patterns in the scene graphs or make predictions about the unseen parts of the environment. Furthermore, hierarchical representations like [17] require predefined levels in the hierarchy. Motivated by the above, an increased research output can be seen in the use of semantic understanding in the domain of path planning.

Among the early works in informative path planning, [21] tackles the problem of object search by maintaining object-object and object-scene co-occurrence probabilities in an information map, which is then used for informative path planning. The authors in [23] present an exploration path planning strategy utilizing a semantic topometric map built using structural semantics such as intersections, pathways, etc. As semantics-aware informative path planning requires a combined metric-semantic map. The work in [42] presents a framework to handle multi-layered metric-semantic maps along with its application in semantics-aware exploration path planning.

In recent years, several works have proposed heuristic functions to combine the exploration and semantic mapping objectives. An information gain formulation based on entropy change in the map, where each voxel's contribution to the entropy is weighted by it belonging to an object of interest, is proposed in [43] for Next-Best-View (NBV) generation for the task of 3D reconstruction. The authors of [44] propose an entropy-based function to score the information gain of a pose based on visible semantic features and use it in an NBV exploration path planning formulation. The work in [22] proposes an objective function for viewpoint evaluation combining the objectives of maximizing unknown volume mapping, viewing all surfaces from a given distance for object detection, and viewing all semantics at a different maximum viewing distance. The authors of [45] present an occupancy map encoding the probabilities of the voxels belonging to a semantic class and exploit it in an NBV exploration planning approach to explore and increase the detection confidence of the semantics. Similarly, the effort in [46] tackles the problem of semantic-object search and mapping in unknown environment using the NBV approach with a heuristic balancing exploration and semantic mapping quality. Along these lines, [47] uses a similar heuristic with a frontier-based exploration strategy. The authors in [48] present a multi-layered object-centric volumetric map which is then used to extract semantics-aware frontiers for object centric mapping. Departing from heuristics combining multiple objectives in an additive fashion, our previous work in [29] presents a combined exploration and semantic inspection strategy that utilizes distinct planning behaviors for volumetric exploration, semantic surface reconstruction, and semantic inspection. The contribution in [24] presents a data structure called the semantic belief behavior graph that encodes behavior nodes for various semantics-aware policies. The work in [49] presents a path planning algorithm for Unmanned Aerial Vehicle (UAV) to map large areas (e.g., agricultural fields) that adapts the paths to obtain high-resolution semantic segmentations of areas of interest. The authors of [50] detail a planning framework for active learning in UAV-based semantic mapping. In addition to the above single robot path planning works, the efforts in [51], [52], [53] present collaborative path planning methods for multi-robot semantics mapping.

Semantics-aware planning has also found its utility in navigation tasks. The authors of [54] propose a path planning strategy for a robot using a vision-based localization solution that aims to avoid areas leading to high localization drift using semantically segmented images. The work in [55] presents a RandLA-Net and KNN-based algorithm for metric-semantic mapping and proposes a semantics-aware A* algorithm that accounts for risks associated with the types of obstacles. The local planner presented in [25] uses semantic information to learn traversability for legged robots. Similarly, [56] proposes a local planner for unmanned ground vehicles using a metric-semantic traversability map. Furthermore, several works have focused on object goal navigation. This task involves searching for an object belonging to a given semantic class in an unknown environment. The effort in [30] proposes a probabilistic planning framework that utilizes a Relational Semantic Network trained to estimate the probability of finding the target object within the spatial elements of an SSG. Several learning-based approaches, such as [57], [58], [59], [59], [60], [61], train models to learn relationships between different semantic classes using knowledge graphs or existing SSG datasets, to take actions based on semantics seen in the vicinity of the robot. The work in [62] utilizes SSG along with a Large Language Model to perform online planning for tasks described in natural language.

Beyond the above, this work also relates to the niche community investigating the prediction of the unknown areas of the map. The works in [63] and [64] propose strategies to complete partial occupancy maps using learning-based approaches. On the other hand, the contribution in [65] presents work on map prediction beyond frontiers using previously mapped environments. The above approaches demonstrate the utility of the predicted maps for exploration path planning. However, they do not exploit semantic information. Beyond pure geometric predictions, a plethora of works has happened in semantic scene completion which aims to fill in missing information in a semantic map such as an SSG, occupancy map, or semantic mesh [66]. However, these methods do not make predictions beyond filling in missing information and do not present their use case in path planning tasks.

Compared to the current literature, this paper presents a new paradigm called Semantics-aware Predictive Planning (SPP) and contributes three main methodological contributions:

1) An algorithm is presented to find repeating patterns of semantics in an SSG. Specifically, this work exploits and extends the SUBDUE algorithm [67], to find subgraphs that have multiple instances - exact or inexact - in an SSG indicating a repeating pattern of semantics. The original work tackles the problem of finding structural and relational patterns in data represented as a graph. Key extensions to the SUBDUE algorithm are made to improve the inexact pattern detection by introducing a more accurate search strategy, and to exploit the pose of the vertices in the SSG.
2) Using the detected patterns a strategy to find areas in the SSG that can potentially extend to the detected patterns is proposed.
3) The paper presents two predictive path planning algorithms that exploit the developed SSG prediction strategy for significantly improving the efficiency of robot inspections. Without loss of generality regarding the applicability of the pattern detection and graph prediction process, the predictive inspection planning methods are applied towards the task of visual inspection of ship ballast water tanks. The overall contribution is then extensively verified regarding its efficiency both in simulation and field experiments.

An illustration of the components of this work is shown in Figure 2. A video describing the proposed framework is available here: https://youtu.be/uc1j6VgthM8.

III. PROBLEM FORMULATION

The overall problem considered in this work is that of the inspection of the surfaces of a set Θ of semantics of interest, distributed in an initially unknown volume V, using a camera sensor $\mathcal{Y}_C$ –with Field of View (FoV) $[F_C^H, F_C^V]$– at a maximum viewing distance r_C. The environment is represented as a) a discrete occupancy map $\mathbb{M}$, built using the measurements from a depth sensor $\mathcal{Y}_D$ (FoV $[F_D^H, F_D^V]$, max range r_D), consisting of cubical voxels $m \in \mathbb{M}$ having edge length ρ, and b) a Semantic Scene Graph $\mathbb{G}$, where each vertex ν represents a semantic in the environment and an edge e represents the relationship between the vertices it connects. The part of $\mathbb{M}$ consisting of occupied voxels seen by $\mathcal{Y}_C$ is referred to as the *seen map* and denoted by $\mathbb{M}_C$. Similarly, the part consisting of occupied voxels seen by $\mathcal{Y}_D$ and belonging to the semantics in Θ is referred to as the *semantic map* and denoted by $\mathbb{M}_S$. Given the above, the problem at hand can be cast globally as that of seeing all possible voxels $m \in \mathbb{M}_S$ in V using $\mathcal{Y}_C$ within the required viewing distance r_C.

Definition 1 (Residual Semantic Surface) Let Ξ be the set of all collision-free robot configurations, where a robot configuration $\xi \in \Xi$ consists of the robot's position $[x, y, z]$ in 3D space and its yaw ψ. Let $\Xi_m \subset \Xi$ be the set of configurations from which the semantic voxel $m \in \mathbb{M}_S$ can be seen by $\mathcal{Y}_C$. The residual semantic surface is then defined as $\mathbb{M}_S^{res} = \bigcup_{m \in \mathbb{M}_S} (m |\ \Xi_m = \emptyset)$.

Problem 1 (Semantic Inspection) Given a volume V and an initial configuration $\xi_0 = [x_0, y_0, z_0, \psi_0] \in \Xi$ find a collision-free path σ that when traversed by the robot enables inspection of each voxel $m \in \mathbb{M}_S \setminus \mathbb{M}_S^{res}$ using the camera sensor $\mathcal{Y}_C$.

IV. SCENE GRAPH PATTERN PREDICTION

In the context of this paper, we tackle environments presenting spatially repeating patterns of the semantics of interest. Importantly, the method does not assume perfect pattern repetition and can tackle loosely matching patterns. A prime example of such a setting is the ballast water tanks as shown in Figure 1. This work aims to utilize these patterns to predict the unknown parts of the environment, which can then be used to improve the inspection path planning efficiency.

To enable the above, an efficient pattern detection methodology is needed that is robust to imperfect patterns. To this end, the semantics relevant to the mission are represented as an SSG and a graph pattern detection algorithm, built on top of a graph-based knowledge discovery algorithm called SUBDUE [67], is proposed. A pattern in an SSG is defined as a subgraph that has multiple instances in the given SSG. We present key modifications to SUBDUE for a) improving the inexact pattern detection by introducing a more accurate graph search technique and b) exploiting the pose of the vertices of the SSG.

Utilizing the detected patterns, a strategy to extend the SSG in unknown parts of the environment is presented. We define a set of classes, such as doors, windows, and manholes inside ballast tanks, as **Entry Classes (ECs)**. The vertices in the SSG belonging to ECs, called **Entry Vertices (EVs)**, are used as anchor points for graph extension. The method aims to find EVs that do not belong to a pattern and can extend to one of the detected patterns.

A. Semantic Scene Graphs (SSG)

In this work, an SSG is defined as a directed graph $\mathbb{G}$, with its vertex and edge sets $\mathcal{V}$, $\mathcal{E}$ respectively, whose vertices represent semantics present in the environment and the edges represent relations between them. Each vertex $\nu \in \mathcal{V}$ has a label $\iota(\nu)$ indicating the class of the corresponding semantic. Similarly, each edge $e \in \mathcal{E}$ has a label $\iota(e)$ stating the relation between the two vertices it connects. As a vertex represents an object in 3D space, it has a pose associated with it given by $\mathbf{x}(\nu) = [\mathbf{p}_\nu, \mathbf{R}_\nu]$, where $\mathbf{p}_\nu = [x, y, z]$ is the 3D position of the geometric center of the object and $\mathbf{R}_\nu$ is the rotation matrix representing the orientation of the object in an inertial frame w. Additionally, each vertex has a cuboidal bounding box $\mathbf{b}(\nu)$ encompassing the object represented by ν.

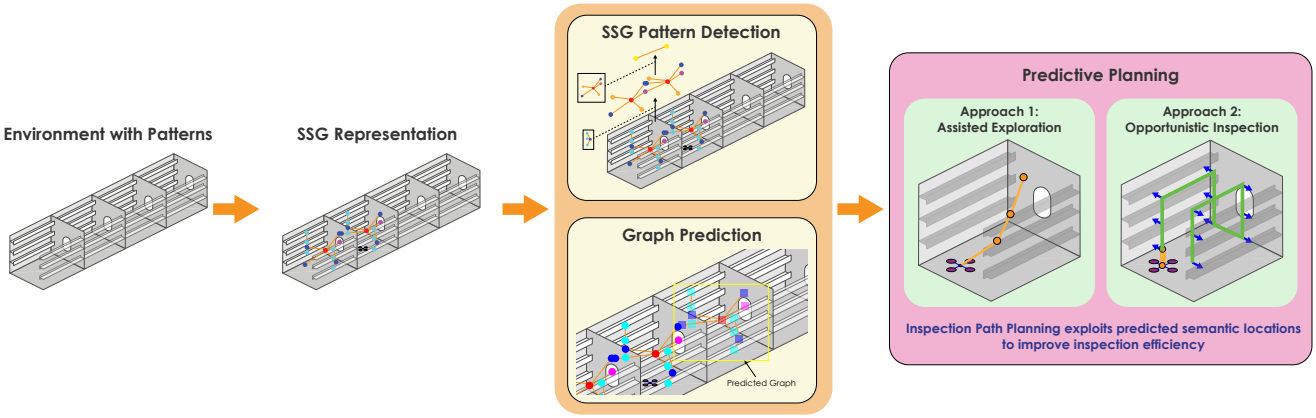


FIGURE 2. **This figure shows the various components of the proposed method and their interaction. The semantics in the environment are represented as a Semantic Scene Graph (SSG). The SSG pattern detection module aims to identify patterns in the SSG, which are then used by the graph prediction module to extend the scene graph into unknown space. The predicted semantic locations are used by the two proposed predictive planning algorithms to improve inspection efficiency. A video describing the proposed framework is available here: https://youtu.be/uc1j6VgthM8.**

B. SSG Pattern Detection - Overview

A pattern in an SSG $\mathbb{G}$ is defined as a subgraph $\mathbb{S}$ (also referred to as a substructure) that has multiple instances in $\mathbb{G}$. To find such substructures, we use and extend the SUBDUE algorithm [67] designed for discovering structural patterns in graph representations of data. SUBDUE uses the Minimum Description Length (MDL) principle to identify a substructure $\mathbb{S}$ that maximizes the compression of the original graph $\mathbb{G}$. The compression is achieved by replacing each instance of the substructure with a single vertex. This process of substructure search and graph compression can be iteratively repeated to form a hierarchical representation $\mathcal{H}$ (as shown in Figure 1) where each level l^i includes the compressed graph $\mathbb{G}^{l^i|\mathbb{S}^i}$ and the identified substructure $\mathbb{S}^i$. Throughout the paper, when we refer to the compressed version of a graph $\mathbb{G}$ created using a substructure $\mathbb{S}$ without referring to its level in the hierarchy, it is denoted as $\mathbb{G}^{1|\mathbb{S}}$. To tackle minor discrepancies between instances of substructures, SUBDUE uses inexact graph-matching techniques to search for substructures whose instances are inexact isomorphs. Contrary to the exact graph-matching algorithms that return a Boolean value stating whether the two graphs under consideration are isomorphs, inexact graph-matching algorithms return a dissimilarity value (hereafter referred to as graph-matching cost) between the two graphs. This work uses the inexact graph-matching algorithm described in [67].

First, we briefly summarize the SUBDUE algorithm from the original paper [67], followed by detailed descriptions of the modifications done in this work. Algorithm 1 details the steps. We provide a detailed video description of the SSG pattern detection algorithm, including our modifications at **https://youtu.be/WrqQu67gz1I**. Let $\mathcal{V}$, $\mathcal{E}$ be the vertex and edge sets of $\mathbb{G}$, respectively. The algorithm begins by creating a priority queue q having the vertices $\nu \in \mathcal{V}$ with unique label as the initial set of substructures. The priority of a substructure $\mathbb{S}$ in q is the compression value $\Gamma(\mathbb{G}, \mathbb{S})$, and the elements in the queue are sorted in ascending order. Therefore the first element $q[1]$ of q has the lowest Γ and represents the substructure that provides the most amount of graph compression. The compression value $\Gamma(\mathbb{G}, \mathbb{S})$ of a substructure $\mathbb{S}$ for compressing the graph $\mathbb{G}$ resulting in the compressed graph $\mathbb{G}^{1|\mathbb{S}}$ is defined as:

$$\Gamma(\mathbb{G}, \mathbb{S}) = \frac{DL(\mathbb{S}) + DL(\mathbb{G}^{1|\mathbb{S}})}{DL(\mathbb{G})}, \tag{1}$$

where $DL(\mathbb{S}), DL(\mathbb{G}^{1|\mathbb{S}})$, and $DL(\mathbb{G})$ are the Description Lengths of the substructure, the compressed graph, and the original graph respectively. In this work, the Description Length of a graph is defined as the sum of the number of vertices and edges in the graph.

In each iteration of the algorithm, every instance $\mathbb{I}^i_{\mathbb{S}} \in \mathcal{I}_{\mathbb{S}}$ ($\mathcal{I}_{\mathbb{S}}$ is the set of instances of $\mathbb{S}$) of each substructure $\mathbb{S} \in q$ is expanded in all possible ways by a single edge and the corresponding vertex, and the extended instances are regrouped into sets of inexact isomorphs to form the new set of substructures $\mathcal{S}_{ext}$ (line 7: **ExtendSubstruct**). The compression value Γ is calculated for each substructure (using the function **Evaluate**) and the substructure is added to another priority queue q_{new}. Only the first $\gamma_b > 0$ number of substructures in q_{new} are retained using the **mod** operator for computational tractability. If $\Gamma(\mathbb{G}, q_{new}[1]) < \Gamma(\mathbb{G}, \mathbb{S}_{best})$, where $\mathbb{S}_{best}$ is the substructure providing the lowest compression value found so far and $q_{new}[1]$ is the first element of q_{new}, then $\mathbb{S}_{best}$ is updated to be $q_{new}[1]$. The old priority queue q is replaced by the truncated q_{new} and the process is repeated until either q is empty or a maximum user-defined number of iterations γ_l are carried out.

To use SUBDUE effectively in the presented work, key feature additions and improvements are necessary. First, despite the inexact graph-matching formulation, the vanilla SUBDUE algorithm had difficulties identifying inexact patterns in sparse graphs presenting a high degree of imperfections (up to 31.25% of instances having imperfection)

Algorithm 1 SUBDUE Substructure Discovery

1: **function** SUBDUE($\mathbb{G}$, γ_b, γ_l)
2: $q \leftarrow \{\nu \in \mathcal{V} | \nu$ has unique label$\}$
3: $\mathbb{S}_{best} \leftarrow q[1]$
4: **while** $q \neq \emptyset$ **and** $\gamma_l > 0$ **do**
5: $q_{new} \leftarrow \{\}$
6: **for all** $\mathbb{S} \in q$ **do**
7: $\mathcal{S}_{ext} \leftarrow$ **ExtendSubstruct**$(\mathbb{S}, \mathbb{G})$
8: **Evaluate**$(\mathcal{S}_{ext})$
9: $q_{new} \leftarrow (q_{new} \bigcup \mathcal{S}_{ext})$ **mod** γ_b
10: $\gamma_l \leftarrow \gamma_l - 1$
11: **if** $\Gamma(\mathbb{G}, q_{new}[1]) < \Gamma(\mathbb{G}, \mathbb{S}_{best})$ **then**
12: $\mathbb{S}_{best} \leftarrow q_{new}[1]$
13: $q \leftarrow q_{new}$
14: return $\mathbb{S}_{best}$

in our study, as highlighted in Section VI.B. This is due to the greedy graph search used to find inexact patterns. Furthermore, while calculating the graph-matching cost, SUBDUE treats all vertices equally. However, a mismatch at a vertex having a higher degree in the graph generally implies a higher dissimilarity between the two graphs. Second, SUBDUE does not provide a way to utilize the vertex pose $\mathbf{x}(\nu)$ in the graph-matching procedure. This is particularly important for SSGs as the vertices represent objects in 3D space.

To tackle these shortcomings, this work presents the following key modifications to the SUBDUE algorithm for substructure discovery:

1) Improvements in the **ExtendSubstruct** method: The extended substructure instance regrouping strategy inside the **ExtendSubstruct** function is improved by using a more accurate search as opposed to the original greedy search (more details in Section IV.C).
2) Modifications to Inexact-Graph Matching:
   a) Since the proposed methods work with SSG representing objects in 3D space, an application not considered in the original work on SUBDUE, the pose $\mathbf{x}(\nu)$ of vertex ν is used in the inexact graph-matching algorithm.
   b) Given two graphs, the inexact graph-matching algorithm returns a cost relating to the transformations required to make them isomorphic. We scale the costs of each transformation by the degree of the vertices involved in the transformation to penalize discrepancies at vertices having a higher degree.

C. Modifications to SUBDUE

1) Improvements in ExtendSubstruct method

The **ExtendSubstruct** is an important function in the SUBDUE algorithm as it performs the key operations of extending and regrouping the substructures to form the new set of substructures. We first describe the modified version of the **ExtendSubstruct** method which is detailed in Algorithm 2. The method extends each instance $\mathbb{I}^i_{\mathbb{S}} \in \mathcal{I}_{\mathbb{S}}$ by a single edge e (and the end vertex if not already in $\mathbb{I}^i_{\mathbb{S}}$) in all possible ways to form the set of extended subgraphs $\mathfrak{I}^i_{\mathbb{S}}$ (lines $4-9$). Let $\mathcal{L}_{\mathbb{S}}$ be the set of all $\mathfrak{I}^i_{\mathbb{S}}$ corresponding to $\mathbb{S}$. The algorithm iteratively searches through the subgraphs in the sets $\mathfrak{I}^i_{\mathbb{S}} \in \mathcal{L}_{\mathbb{S}}$ to find the inexact isomorphs and group them together. In each iteration, the first subgraph, denoted by $\mathbb{I}_+$, in the first non-empty set $\mathfrak{I}^i_{\mathbb{S}} \in \mathcal{L}_{\mathbb{S}}$ is selected and added to the new substructure $\mathcal{S}_+$ (line 14). Next, the graph-matching cost t between $\mathbb{I}_+$ and each element $\mathbb{I}'_+$ of the remaining sets $\mathfrak{I}^j_{\mathbb{S}}, j \neq i$ is calculated. The inexact graph-matching algorithm (with modifications mentioned above) described in [67] is used in this work. The element $\mathbb{I}^j_{best}$ with the lowest cost t^j_{best} from each set $\mathfrak{I}^j_{\mathbb{S}} \in \mathcal{L}_{\mathbb{S}}, j \neq i$ is selected, removed from $\mathfrak{I}^j_{\mathbb{S}}$, and added to $\mathcal{S}_+$ if $t^j_{best} \leq t_{thr}$ (where t_{thr} is a user defined maximum allowed graph-matching cost). This process is repeated until $\mathcal{L}_{\mathbb{S}}$ is empty.

The original implementation of SUBDUE[1] performs the subgraph grouping differently. Figure 3 compares the original and modified approach at one instance of the algorithm. Note that the extended subgraphs of all instances of the substructure under consideration are not shown in the figure for clarity of visualization. We will briefly explain the original implementation with the help of the example in Figure 3. Note that vertices shown in the same color belong to the same semantic class. The original approach does not maintain the subgraph sets $\mathfrak{I}^i_{\mathbb{S}}$ according to the instances of $\mathbb{S}$. Instead, all extended subgraphs are added to the same set $\mathcal{L}^c_{\mathbb{S}}$. The method starts with the first element $\mathbb{I}_+ \in \mathcal{L}^c_{\mathbb{S}}$ and adds it to $\mathcal{S}_+$ (subgraph '(a)' in Figure 3). The graph-matching cost t is calculated between $\mathbb{I}_+$ and the first element in $\mathcal{L}^c_{\mathbb{S}}$ that does not overlap with any element of $\mathcal{S}_+$, which in this example is subgraph '(d)'. If $t \leq t_{thr}$, subgraph '(d)' is added to $\mathcal{S}_+$. The algorithm continues to find the next element not overlapping with any element of $\mathcal{S}_+$. Note that in this example, the subgraph'(e)' is overlapping with '(d)' and hence will not be evaluated even though it is an exact isomorph of '(a)'. This is contrary to the result in the modified approach (Figure 3: Modified Approach) which correctly groups '(a)' and '(e)' together. Once the entire $\mathcal{L}^c_{\mathbb{S}}$ is parsed, $\mathcal{S}_+$ is added to $\mathcal{S}_{ext}$ and the process is repeated with the next element in $\mathcal{L}^c_{\mathbb{S}}$ (in this example '(b)' in Iteration 2).

2) Modifications to Inexact Graph Matching

For two graphs $\mathbb{G}_1$ and $\mathbb{G}_2$, the problem of inexact graph matching is that of finding a mapping $f : \mathcal{V}_1 \mapsto \mathcal{V}_2 \bigcup \{\lambda\}$ –where $\mathcal{V}_1, \mathcal{V}_2$ are the vertex sets of graphs $\mathbb{G}_1, \mathbb{G}_2$ respectively– that minimizes a cost function C_{tot} relating to the transformations required to make $\mathbb{G}_1$ and $\mathbb{G}_2$ isomorphs. If $n(\mathcal{V}_1) \neq n(\mathcal{V}_2)$, where $n(A)$ denotes the number of

[1]https://ailab.wsu.edu/subdue/software/subdue-5.2.2.zip

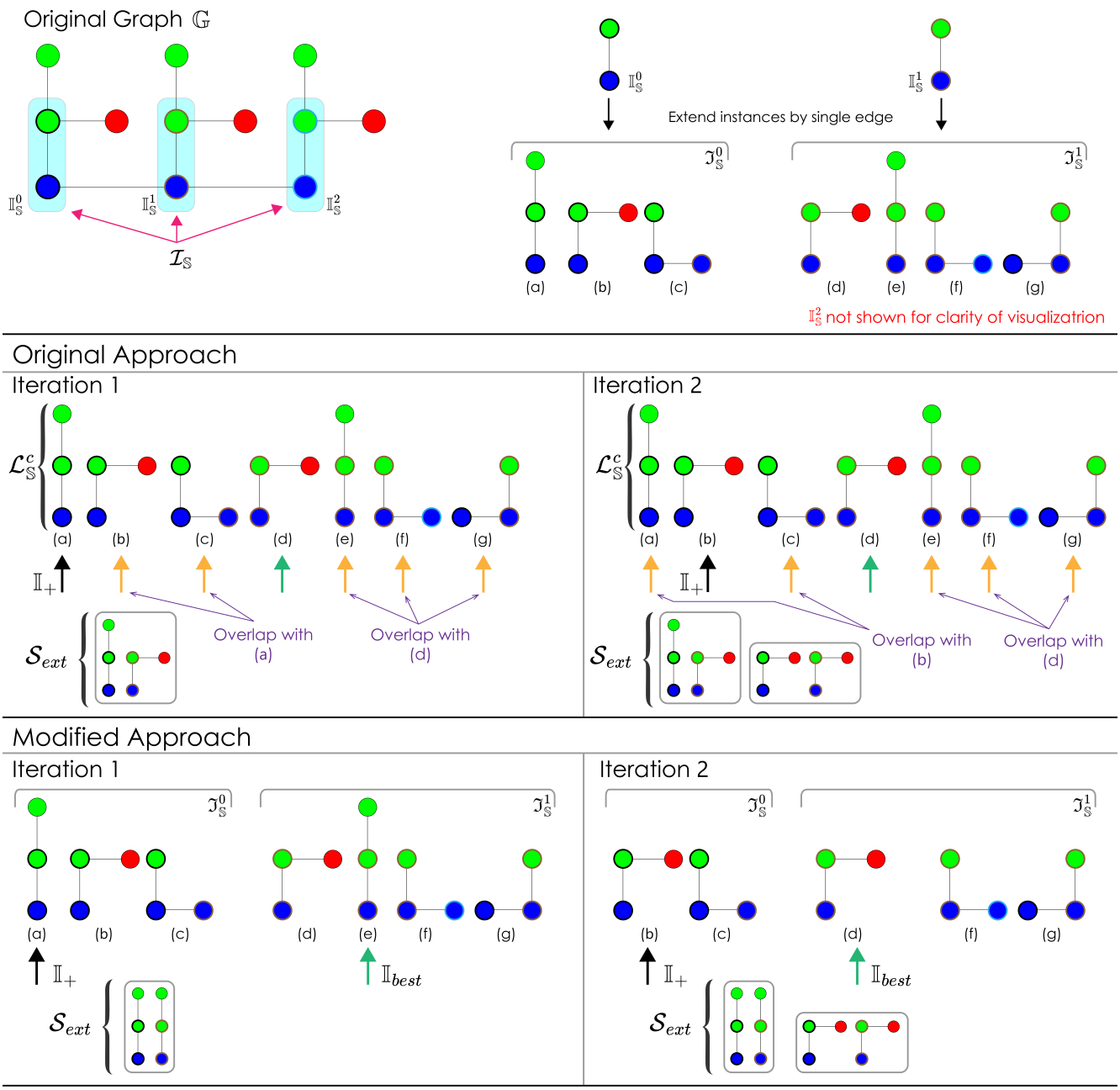


FIGURE 3. **This figure shows the improvement made to the ExtendSubstruct method through an example. Vertices shown in the same color belong to the same semantic class. During the extended instance regrouping step, the modified approach calculates the graph-matching cost between the selected instance and all other instances that can be its isomorphs and selects the best among them. On the other hand, the original algorithm selects the first non-overlapping instance, whose graph-matching cost is within the threshold, and adds it to the substructure group. This can result in an incorrect grouping as shown. However, due to the more accurate grouping approach, the modified algorithm can find the correct groupings.**

elements in a set A, $\mathbb{G}_1$ and $\mathbb{G}_2$ are selected such that $n(\mathcal{V}_1) > n(\mathcal{V}_2)$ and the unmapped vertices from $\mathcal{V}_1$ are mapped to a virtual vertex λ (called the null vertex). A total of 7 transformations are considered in this work, namely, the addition/deletion of a vertex, the addition/deletion of an edge, changing the label of a vertex or an edge, and changing the relative pose of a vertex. It is noted that the original inexact graph-matching algorithm in [67] does not consider the vertex pose $\mathbf{x}(\nu)$. However, since this work operates on SSG with vertices having pose information, this transformation has been added. The mapping f is obtained through a tree search procedure as detailed in [67]. Since our contributions relate to the cost calculation, we refer the readers to the original paper [67] for further details on the algorithm to obtain the mapping f.

The cost calculation is split into two parts. First, the cost related to the change in the relative pose of the vertices, referred to as the pose cost C_P, and second, the cost related

Algorithm 2 Extend Substructures

function EXTENDSUBSTRUCT($\mathbb{G}$, $\mathbb{S}$)
  $\mathcal{S}_{ext} \leftarrow \emptyset$
  $\mathcal{L}_{\mathbb{S}} \leftarrow \emptyset$
  for all $\mathbb{I}^i_{\mathbb{S}} \in \mathcal{I}_{\mathbb{S}}$ **do**
    $\mathfrak{I}^i_{\mathbb{S}} \leftarrow \emptyset$
    for all $e \in \mathcal{E} \mid e$ is neighboring $\mathbb{I}^i_{\mathbb{S}}$ **do**
      $\mathbb{I}_+ \leftarrow$ **ExtendByEdge**$(e, \mathbb{I}^i_{\mathbb{S}}, \mathbb{G})$
      $\mathfrak{I}^i_{\mathbb{S}} \leftarrow \mathfrak{I}^i_{\mathbb{S}} \bigcup \mathbb{I}_+$
    $\mathcal{L}_{\mathbb{S}} \leftarrow \mathcal{L}_{\mathbb{S}} \bigcup \mathfrak{I}^i_{\mathbb{S}}$
  while $\mathcal{L}_{\mathbb{S}} \neq \emptyset$ **do**
    $\mathfrak{I}^i_{\mathbb{S}} \leftarrow$ **PopFront**$(\mathcal{L}_{\mathbb{S}})$
    while $\mathfrak{I}^i_{\mathbb{S}} \neq \emptyset$ **do**
      $\mathbb{I}_+ \leftarrow$ **PopFront**$(\mathfrak{I}^i_{\mathbb{S}})$
      $\mathcal{S}_+ \leftarrow \{\mathbb{I}_+\}$
      for all $\mathfrak{I}^j_{\mathbb{S}} \in \mathcal{L}_{\mathbb{S}}, j \neq i$ **do**
        $t^j_{best} \leftarrow t_{thr}$
        $\mathbb{I}^j_{best} \leftarrow \emptyset$
        for all $\mathbb{I}'_+ \in \mathfrak{I}^j_{\mathbb{S}}$ **do**
          $t \leftarrow$ **GraphMatchingCost**$(\mathbb{I}'_+, \mathbb{I}_+)$
          if $t < t^j_{best}$ **then**
            $t^j_{best} \leftarrow t$
            $\mathbb{I}^j_{best} \leftarrow \mathbb{I}'_+$
        $\mathcal{S}_+ \leftarrow \mathcal{S}_+ \bigcup \mathbb{I}^j_{best}$
        $\mathfrak{I}^j_{\mathbb{S}} \leftarrow \mathfrak{I}^j_{\mathbb{S}} \setminus \mathbb{I}^j_{best}$
      $\mathcal{S}_{ext} \leftarrow \mathcal{S}_{ext} \bigcup \mathcal{S}_+$
  return $\mathcal{S}_{ext}$

to the remaining transformations, denoted by C_R. The total cost C_{tot} is then defined as:

$$C_{tot} = C_P + C_R \tag{2}$$

Modification a) Calculation of Pose Cost C_P: To calculate C_P, two point sets $\mathcal{P}_1$ and $\mathcal{P}_2$ corresponding to the positions of the vertices in $\mathcal{V}_1$ and $\mathcal{V}_2$ respectively, excluding those mapped to λ, are created. A rigid 3D transform $T_{2,1}$ is calculated to align $\mathcal{P}_2$ to $\mathcal{P}_1$, with the mappings in f as the correspondences, by solving the point set registration problem. Next, $\mathcal{P}_2$ is transformed using $T_{2,1}$ to get the point set $\mathcal{P}_{2,1}$ that is aligned with $\mathcal{P}_1$. Then the pose cost C_P of f is calculated as:

$$\begin{aligned} C_P &= \gamma_p \sum_{\nu_i \in \mathcal{V}_1 | f(\nu_i) \neq \lambda} \frac{d(\nu_i, f)}{d_{\max}} \\ d(\nu_i, f) &= \begin{cases} 0, & \text{if } |\mathbf{p}_{\nu_i} - \mathbf{p}_{f(\nu_i)}| \leq d_{\min} \\ |\mathbf{p}_{\nu_i} - \mathbf{p}_{f(\nu_i)}|, & \text{if } d_{\min} \leq |\mathbf{p}_{\nu_i} - \mathbf{p}_{f(\nu_i)}| \leq d_{\max} \\ d_{\max}, & \text{if } d_{\max} \leq |\mathbf{p}_{\nu_i} - \mathbf{p}_{f(\nu_i)}| \end{cases} \end{aligned} \tag{3}$$

where $\mathbf{p}_{\nu_i}$ is the point in $\mathcal{P}_1$ corresponding to vertex $\nu_i \in \mathcal{V}_1$ and $\mathbf{p}_{f(\nu_i)}$ is the point in set $\mathcal{P}_{2,1}$ corresponding to the vertex $f(\nu_i) \in \mathcal{V}_2 \bigcup \{\lambda\}$ which ν_i is mapped to. $\gamma_p > 0$ is a tunable weight for the pose cost. As real SSG may contain noisy data causing the relative positions of the vertices across instances of the substructure to not be identical, a cost is added only if $|\mathbf{p}_{\nu_i} - \mathbf{p}_{f(\nu_i)}| \geq d_{\min}$ where $d_{\min}$ is a user-defined threshold. Similarly, a constant value $d_{\max}$ is assigned to $d(\nu_i, f)$ for all mappings satisfying $|\mathbf{p}_{\nu_i} - \mathbf{p}_{f(\nu_i)}| \geq d_{\max}$.

Modification b) Calculation of C_R ***and Cost Scaling based on Vertex Degree***: Each transformation type $\tau^k, k = 1...6$, except changing the relative pose of a vertex, is assigned a user-defined fixed cost called the transformation cost c_{τ^k}. To make the graphs $\mathbb{G}_1$ and $\mathbb{G}_2$ isomorphs, the vertices, and the edges connecting them, in each pair $\nu_i, f(\nu_i)$ in the mapping f might need to undergo one of the above transformations. This results in a cost $C_T(\nu_i)$ associated with the mapping of each vertex ν_i.

Let $C_{\tau^k}(\nu_i, f(\nu_i))$ be a function that returns a boolean stating if the mapping of vertex $\nu_i \in \mathcal{V}_1$ to the corresponding vertex $f(\nu_i) \in \mathcal{V}_2 \bigcup \{\lambda\}$, requires the transformation τ^k for the $\mathbb{G}_1, \mathbb{G}_2$ to be isomorphs. The cost $C_T(\nu_i)$ of the mapping of vertex ν_i is then defined as:

$$C_T(\nu_i) = \sum_{k=1}^{6} c_{\tau^k} C_{\tau^k}(\nu_i, f(\nu_i)) \tag{4}$$

In the original SUBDUE algorithm, the total cost of a mapping f resulting from the above transformations is the sum of the costs $C_T(\nu_i) \forall \nu_i \in \mathcal{V}_1$. However, an incorrect mapping at a vertex having higher degree in the graph implies a higher discrepancy. Hence, each cost $C_T(\nu_i)$ is scaled with a parameter, $\gamma_{d,i}$, related to the degrees of the vertices $\nu_i, f(\nu_i)$. This parameter is defined as $\gamma_{d,i} = 1 + \frac{\eta}{\eta_{\max}}$, where η is the highest degree among $\nu_i, f(\nu_i)$ and $\eta_{\max}$ is the highest degree of any vertex in $\mathbb{G}_1$ and $\mathbb{G}_2$. Therefore, the cost C_R is calculated as

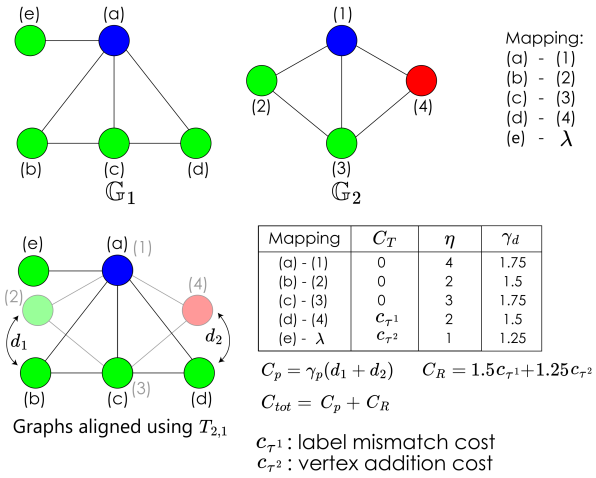
$$C_R = \sum_{\nu_i \in \mathcal{V}_1} \gamma_{d,i} C_T(\nu_i) \tag{5}$$

Figure 4 shows an example of the cost calculation.

D. Scene Graph Prediction using Patterns

The aim of graph prediction is to identify areas in the SSG that can extend to one of the identified patterns. To this end, the proposed approach utilizes vertices whose labels belong to specific classes, hereafter referred to as '**ECs**', as anchor points for graph extension. These classes correspond to objects such as manholes in ballast tanks, doors or windows in a building, etc. The vertices having their label as an Entry Class (EC) are referred to as **Entry Vertices** ϑ^e. The algorithm operates on each level of the hierarchy whose corresponding substructure contains at least one ϑ^e. The algorithm attempts to find ϑ^e where the substructure can fit with no overlap between vertices of different classes and maximum overlap between vertices of the same classes.

The graph prediction takes place in two steps. First, in the graphs in each level of the hierarchical structure, extra



Mapping	C_T	η	γ_d
(a) - (1)	0	4	1.75
(b) - (2)	0	2	1.5
(c) - (3)	0	3	1.75
(d) - (4)	c_{τ^1}	2	1.5
(e) - λ	c_{τ^2}	1	1.25

FIGURE 4. **This figure shows an example of the graph-matching cost calculation between two graphs for a given mapping f. Vertices shown in the same color belong to the same semantic class. The mapping f is shown in the top right corner. The transformations involved in this mapping are: label mismatch ((d) - (4)) and vertex addition ((e) - λ). Each cost $C_T(\nu_i)$ for vertex $\nu_i \in \mathcal{V}_1$, $\mathcal{V}_1$ is the set of vertices of $\mathbb{G}_1$, is scaled by the factor $\gamma_{d,i}$ related to the degree of the relevant vertices.**

vertices called Complementary Entry Vertices $\bar{\vartheta}^e$ are added corresponding to each ϑ^e. $\bar{\vartheta}^e$ have the same label and pose as the corresponding ϑ^e. The $\bar{\vartheta}^e$ addition process is illustrated in Figure 5. Only the levels that have substructures containing ϑ^e are considered for graph prediction. Second, the algorithm attempts to fit the substructure at one of the ϑ^e or $\bar{\vartheta}^e$. This process is illustrated in Figure 6. A detailed video description of the graph prediction is available at **https://youtu.be/b6-3C_rKdiY**.

1) Complementary Entry Vertex Addition

The subsequent section details the process of $\bar{\vartheta}^e$ addition for one level using the example in Figure 5. In 'Step 1', if an instance $\mathbb{I}_\mathbb{S}$ of substructure $\mathbb{S}$ contains a ϑ^e, a $\bar{\vartheta}^e$ is added at the same pose but not included in $\mathbb{I}_\mathbb{S}$ (e.g., '(2)-($\bar{2}$)', '(6)-($\bar{6}$)' in Figure 5). The edges connected to that ϑ^e that are part of $\mathbb{I}_\mathbb{S}$ remain connected to ϑ^e. The remaining edges are connected to $\bar{\vartheta}^e$. An edge is added between ϑ^e and $\bar{\vartheta}^e$. This process is repeated for each ϑ^e in all $\mathbb{I}_\mathbb{S} \in \mathbb{S}$ in that level. It is highlighted that the position of the ϑ^e and $\bar{\vartheta}^e$ are different in Figure 5 only for visualization clarity. In 'Step 2', for any instance $\mathbb{I}_\mathbb{S}$ that connects to a ϑ^e that is not part of it such that the neighbor of that ϑ^e inside $\mathbb{I}_\mathbb{S}$ is not a $\bar{\vartheta}^e$ (e.g., vertex '(10)'), a $\bar{\vartheta}^e$ is added inside $\mathbb{I}_\mathbb{S}$ in the same way as described above. In 'Step 3', for any instance $\mathbb{I}_\mathbb{S}$ that connects to a $\bar{\vartheta}^e$ that is not part of it such that its neighbor inside $\mathbb{I}_\mathbb{S}$ is not a ϑ^e (e.g., vertex '(4)' connecting to '($\bar{6}$)'), this $\bar{\vartheta}^e$ is included into $\mathbb{I}_\mathbb{S}$ (e.g., '($\bar{6}$)' is included in $\mathbb{I}^0_\mathbb{S}$). The graph resulting from the above modification is referred to as the **Modified Graph** $\mathbb{G}'$.

2) Graph Prediction

Once the graph modification is done, ϑ^e and $\bar{\vartheta}^e$ in $\mathbb{G}'$ not part of any instance of a substructure are selected as potential vertices that can extend to one of the substructures. These vertices are referred to as **Loose Entry Vertices** ${}^L\vartheta^e$ or **Loose Complementary Entry Vertices** ${}^L\bar{\vartheta}^e$. We now explain the procedure to select the best ${}^L\vartheta^e$ in the graph for extension. The procedure is described only for ${}^L\vartheta^e$ as the procedure for ${}^L\bar{\vartheta}^e$ is identical. For a level l - with the graph $\mathbb{G}$ and the corresponding substructure $\mathbb{S}$ - of the hierarchical representation $\mathcal{H}$, let ${}^L\mathcal{V}^e$ be the set of all ${}^L\vartheta^e$ in $\mathbb{G}$ and $\mathcal{V}^e_\mathbb{S}$ be the set of all ϑ^e in $\mathbb{S}$. Each vertex ${}^L\vartheta^e_j \in {}^L\mathcal{V}^e$ is checked if it can be extended to $\mathbb{S}$ by calculating an overlap score. For each $\vartheta^e_{i,\mathbb{S}} \in \mathcal{V}^e_\mathbb{S}$, the $\vartheta^e_{i,\mathbb{S}}$ and ${}^L\vartheta^e_j$ are aligned by setting $\mathbf{x}(\vartheta^e_{i,\mathbb{S}}) = \mathbf{x}({}^L\vartheta^e_j)$ and the remaining vertices in $\mathbb{S}$ are transformed accordingly. The overlap between the bounding boxes of the transformed vertices of $\mathbb{S}$ with those of the vertices in $\mathbb{G}$ is computed. For each bounding box $\mathbf{b}(\nu_{k,\mathbb{S}}), \nu_{k,\mathbb{S}} \in \mathcal{V}_\mathbb{S}$, where $\mathcal{V}_\mathbb{S}$ is the set of vertices of $\mathbb{S}$, overlapping with a vertex in $\mathbb{G}$ having the same label as $\nu_{k,\mathbb{S}}$, the overlap score for $\vartheta^e_{i,\mathbb{S}}$ is increased by 1. However, if it overlaps with a vertex that is part of an instance of a substructure or has a different label, this candidate is rejected and the process is continued with the next $\vartheta^e_{i,\mathbb{S}}$. The $\vartheta^e_{i,\mathbb{S}}$ with the highest overlap score is the prediction candidate for ${}^L\vartheta^e_j$. This process is repeated and the best prediction candidate for each ${}^L\vartheta^e$ is calculated. The candidates having the top φ percentage of overlap scores are kept and returned as the graph prediction.

Figure 6 illustrates an example of the process of overlap checking for ${}^L\vartheta^e$. The modified graph in Figure 5 is used in this example. Two loose vertices, '($\bar{2}$)' (a ${}^L\bar{\vartheta}^e$) and '(10)' (a ${}^L\vartheta^e$), exist in the graph. The substructure 'P1' has both an ϑ^e and a $\bar{\vartheta}^e$. Hence both of these locations are possible candidates that can extend to 'P1'. In subfigure 1 (top right), the algorithm tries to extend the graph through '($\bar{2}$)' but the bounding box of one of the vertices of the predicted subgraph graph overlaps with vertex '(1)' which does not have the same class. Hence, this prediction is rejected. The algorithm then tries at vertex '(10)'. Here we see no incorrect overlap and this prediction is selected.

V. PREDICTIVE PLANNING

Without loss of generality regarding the concept of SSG, the detection of patterns and their exploitation in planning, this work focuses on ballast tanks inside ships. Ballast tanks represent industrial environments of great importance and present clear repeating patterns in the semantics of interest such as their longitudinal structures as shown in Figure 1. Functionality-wise, ballast tanks are filled with water to adjust the buoyancy of the ship. Due to constant exposure to salt water, the tanks need to be inspected regularly for corrosion, cracks, and deformation. Specific structures, such as longitudinals, inside the ballast tanks are more important for inspection. The ballast tanks are divided into multiple

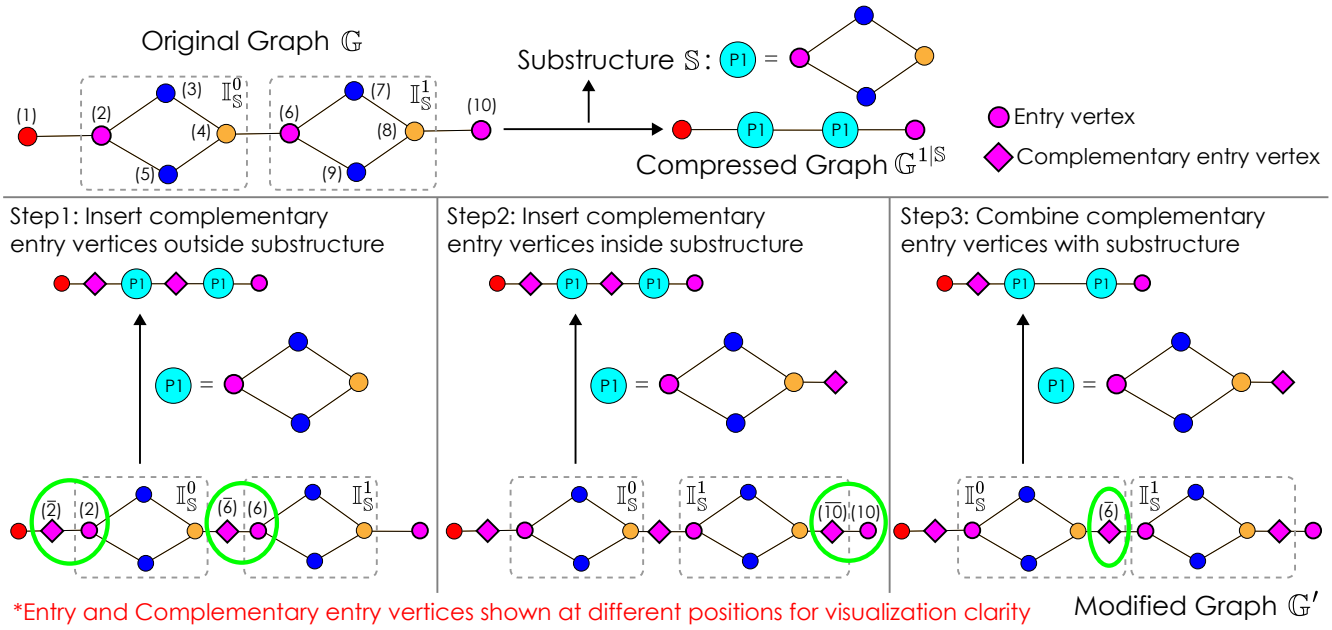


FIGURE 5. **This figure shows the steps involved in the modification of the SSG for graph prediction. Vertices having the same color have the same semantic label. The process inserts additional vertices called Complementary Entry Vertices $\bar{\vartheta}^e$ for each Entry Vertex ϑ^e. The three steps involved in this process are illustrated along with the final modified graph $\mathbb{G}'$.**

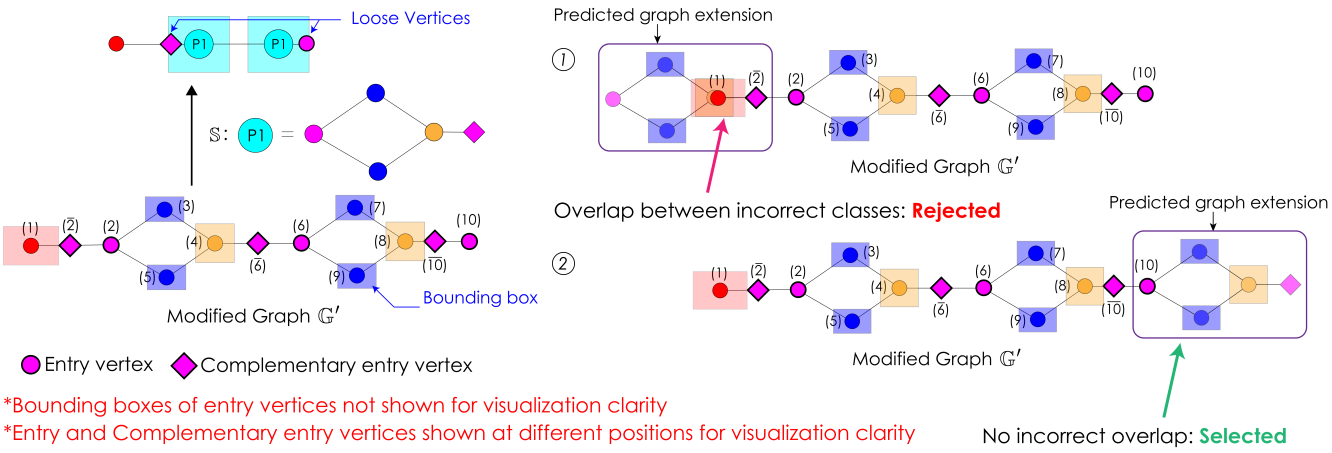


FIGURE 6. **This figure shows an example of the graph prediction process using the detected patterns. The method uses the ${}^{L}\vartheta^e$ and ${}^{L}\bar{\vartheta}^e$ in the modified graph $\mathbb{G}'$ to find the vertex which can extend to the substructure with no overlap between vertices of different classes and maximum overlap between vertices of the same classes. The overlap check at the two candidate vertices is shown in the figure.**

levels with each level consisting of a number of compartments often connected by very narrow openings (as narrow as $0.6\,\mathrm{m} \times 0.4\,\mathrm{m}$) called manholes. In this work, we tackle the problem of autonomous inspection the longitudinals. As Figure 1 shows, the longitudinals form a spatially repeating pattern across the ballast tank compartments.

Motivated by the above, we first present an SSG generation method to represent the longitudinals and other semantics in the ballast tank as a scene graph. Second, a volumetric exploration and inspection strategy is presented for inspecting the longitudinals in the ballast tank. Finally, we describe two approaches that exploit the predicted extensions of the scene graph to make the inspection planning more efficient.

A. Ballast Tank Scene Graph Generation

In this work, we include the following semantics in the scene graphs: 1) longitudinals, 2) walls, 3) compartments, and 4) manholes. The segmentation happens on the point cloud $\mathcal{P}$ from a 3D LiDAR sensor. In this work, the manhole is considered as the 'Entry Class'. Figure 7 shows the different steps in the SSG generation procedure.

1) Wall Segmentation

For each point cloud $\mathcal{P}$, the four largest (based on the number of points) vertical planes, hereafter referred to as wall planes, are segmented using the RANSAC algorithm. The normal of the plane is set to point towards the current

robot location. Each newly detected wall plane $\mathrm{P}_i, i = 1...4$ is checked against all elements of the set $\mathcal{W}$ of previously detected walls to find a wall $W_i \in \mathcal{W}$ that satisfies the following criteria: a) is coplanar to P_i, b) has its normal pointing in the same direction, and c) has its centroid on the same side of all walls $W_j \in \mathcal{W}, j \neq i$ as the centroid of the point cloud belonging to P_i. If such a wall is found, the point clouds of this wall and P_i are combined, sub-sampled to a desired resolution, the new centroid is calculated, and the detection count for that wall is incremented. Otherwise, a new wall entry is added to $\mathcal{W}$. When the detection count of a wall goes above a threshold n^W_{thr}, a new vertex is added to the SSG at the location of the centroid of the wall with its orientation along the normal of the plane of the wall.

2) Compartment Segmentation

A compartment is defined as the cuboidal volume between the four wall planes detected during the wall segmentation step. The position of the compartment is set to be the centroid of the centroids of these four planes. Similar to the wall segmentation, the detected compartment locations are tracked and stored in a set $\mathcal{Q}$. The algorithm attempts to find a compartment whose position is on the same side of all walls in $\mathcal{W}$ as the newly detected compartment Q. If such a compartment is found, its position is updated to be the mean of its current position and the position of Q, and its detection count is incremented. Else, a new compartment entry is added to $\mathcal{Q}$. When the detection count of a compartment goes above a threshold n^Q_{thr}, a new vertex is added to the SSG. Once a compartment vertex is added, edges are added between it and the wall vertices whose centroids and the compartment's position lie on the same side of all other walls in $\mathcal{W}$.

3) Longitudinal Segmentation

It can be seen from Figure 1 that the longitudinals are always attached to walls and are parallel to each other. Exploiting these facts, longitudinal segmentation takes place only in the part of $\mathcal{P}$ close to the detected walls. First, the points $\mathcal{P}^L_i$ from the input $\mathcal{P}$ that lie within a distance d_w from the planes $\mathrm{P}_i, i = 1...4$ are extracted. A set $\mathbb{L}_i$ of parallel lines is segmented out from each $\mathcal{P}^L_i$ as candidate longitudinals. Each longitudinal is characterized by the equation of the line corresponding to it and the centroid of the points belonging to the line. The equations of the newly detected lines $\mathbb{L}_i$ are compared against that of the longitudinals L_j in the set $\mathfrak{L}$ of previously detected longitudinals. For a line $l^k_i \in \mathbb{L}_i$, the algorithm attempts to find a longitudinal with a matching line equation and its centroid lying on the same side of all walls in $\mathcal{W}$ as the centroid of l^k_i. If such a longitudinal is found, its centroid is updated to be the mean of its current centroid and the centroid of l^k_i, and its detection count is incremented. Else, a new longitudinal entry is created. When the detection count of a longitudinal goes above a threshold n^L_{thr} (a tunable parameter), a new vertex is added to the SSG, and an edge is added between it and the corresponding wall.

4) Manhole Segmentation

We utilize the work from [68] for manhole segmentation. Once a manhole is detected, a new vertex is added to the graph and an edge is added between it and the closest two compartment vertices.

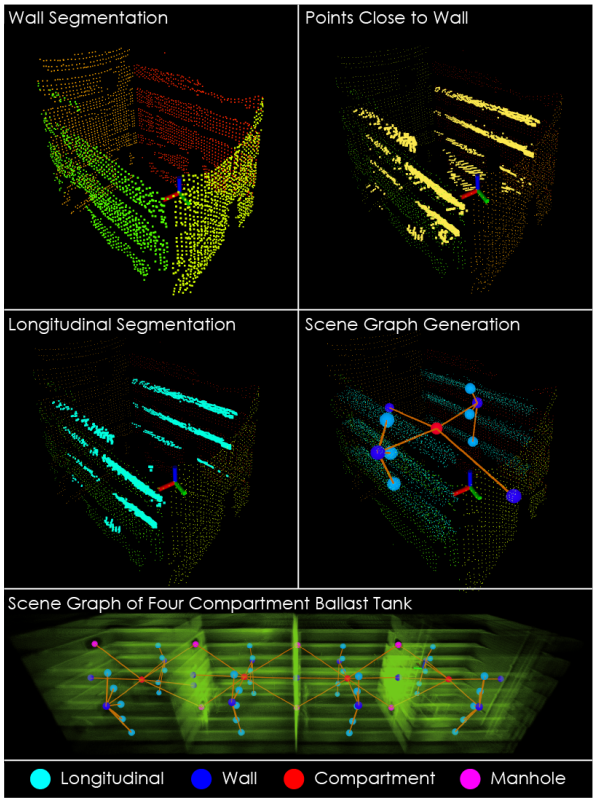


FIGURE 7. Figure showing different steps in the SSG generation in real data. The top four subfigures show the data from one LiDAR scan showing the extracted wall planes, part of the point cloud close to the wall planes, lines extracted within this point cloud, and the current state of the SSG. The subfigure on the bottom shows the final SSG and map at the end of the mission.

B. Predictive Planner Overview

As the ballast tank is divided into multiple compartments, the planner tackles the problem of exploration and longitudinal inspection one compartment at a time. The planner starts with no information about the ballast tank except the number of compartments to inspect. It operates in three planning modes namely Volumetric Exploration (VE), Semantic Inspection (SI), and Predictive Planning (PP). At the beginning of the mission, the planner starts in the VE mode, which aims to map the entire compartment the robot is currently in with $\mathcal{Y}_D$. Upon completion, the SI mode calculates viewpoints and paths to view the part of $\mathbb{M}_S$ within that compartment with $\mathcal{Y}_C$ at a distance r_C. Note that the semantic segmenta-

tion and scene graph generation is taking place in real-time during the entire mission.

After all detected longitudinals in the current compartment are inspected, the pattern detection and graph prediction steps (described in Section IV) are triggered. If a successful prediction is made and a feasible path - calculated using the method described in [68] - for traversing the manhole (the entry vertex) used for prediction exists, the robot is commanded to go through this manhole. Otherwise, the robot is guided to traverse through the closest untraversed manhole. In the first case, the PP mode is triggered which utilizes the predicted longitudinal locations to improve the exploration and inspection efficiency. We present two approaches (hereafter referred to as submodes) for this mode. The first approach called Assisted Exploration (PP-AE), guides the robot towards predicted longitudinal locations as the robot goes through the manhole used for prediction. The second approach called Opportunistic Inspection (PP-OI), utilizes the viewpoint sequence (hereafter referred to as predicted path) from the SI mode in previous compartments and follows the same sequence (with necessary collision-free path planning) as long as the longitudinal corresponding to the viewpoints in the sequence are detected. The flow diagram of the planner is presented in Figure 8 and a detailed video description is available at **https://youtu.be/XDTPodpD4SE**. The subsequent subsections detail each planning mode.

C. Volumetric Exploration (VE)

In this paper, we use our previous open-sourced work on Graph-based Exploration Path Planning [1], [13], called GBPlanner, for the VE mode. GBPlanner operates in a bifurcated local-global planning architecture, where the local planner is responsible for providing efficient, collision-free paths for mapping unknown space within a local volume around the robot. On the other had, the global planner is used to reposition the robot to frontiers of unexplored space when the local planner is unable to provide informative paths, as well as provide return-to-home functionality when the robot's battery reaches its endurance limit.

The VE mode utilizes the local planner of GBPlanner. In each exploration iteration, the planner builds a 3D collision-free graph $\mathbb{G}_{VE}$ within a local volume around the robot. Next, an information gain $\Upsilon_{VE}(\nu_j)$ called Volume Gain is calculated for each vertex $\nu_j, j = 1...\beta$ (β is the number of vertices in $\mathbb{G}_{VE}$) in $\mathbb{G}_{VE}$ that relates to the amount of unknown volume the robot would see if it was at ν_j. The algorithm then calculates the shortest paths σ_j from the vertex ν_1 corresponding to the current robot location to each vertex ν_j in $\mathbb{G}_{VE}$ using Dijkstra's algorithm and computes the accumulated Exploration Gain $\Lambda_{VE}(\sigma_j)$ for each path σ_j as:

$$\Lambda_{VE}(\sigma_j) = e^{-\zeta \mathcal{Z}(\sigma_j, \sigma_e)} \sum_{i=1}^{\beta_j} \Upsilon_{VE}(\nu_i) e^{-\mu \mathcal{D}(\nu_1, \nu_i)}. \quad (6)$$

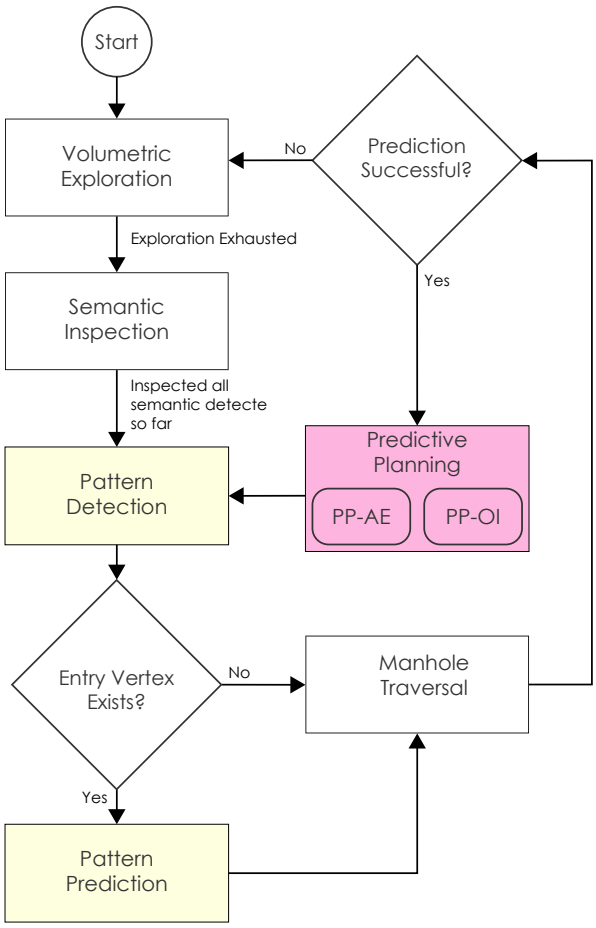


FIGURE 8. Figure showing the flow diagram of the predictive planner.

Here, $\mathcal{D}(\nu_1, \nu_i)$ is length of the shortest path from ν_1 to ν_i along $\mathbb{G}_{VE}$. The function $\mathcal{Z}(\sigma_j, \sigma_e)$ calculates a similarity metric between the path σ_j and a straight line path σ_e along the current exploration direction having the same length penalizing a change in the exploration direction. The constants $\zeta > 0$ and $\mu > 0$ are tunable parameters, and β_j is the number of vertices in σ_j. The path σ_{best} with the highest Exploration Gain is selected and executed by the robot. Upon completion, the process is repeated until Υ_{VE} of all the vertices in $\mathbb{G}_{VE}$ is below a user defined threshold $\Upsilon_{VE,\min}$ at which point the volumetric exploration is complete. Instead of switching to the global planning mode, the planner then switches to the SI mode for inspection of the semantics.

D. Semantic Inspection (SI)

The SI mode aims to find viewpoints and paths to view the voxels in $\mathbb{M}_S$ with the camera sensor $\mathcal{Y}_C$. Given the maximum viewing distance r_C and the FoV of $\mathcal{Y}_C$ as $[F_C^H, F_C^V]$ radians the viewpoint calculation procedure is as follows. For a longitudinal L, let $\mathbf{a} = [d_l^x, d_l^y, d_l^z, p_c^x, p_c^y, p_c^z]$ be the coefficients representing the equation of the line l_L corresponding L. The unit vector $\mathbf{d}_l = [d_l^x, d_l^y, d_l^z]$ represents the direction of the line and the point $\mathbf{p}_c = [p_c^x, p_c^y, p_c^z]$ is

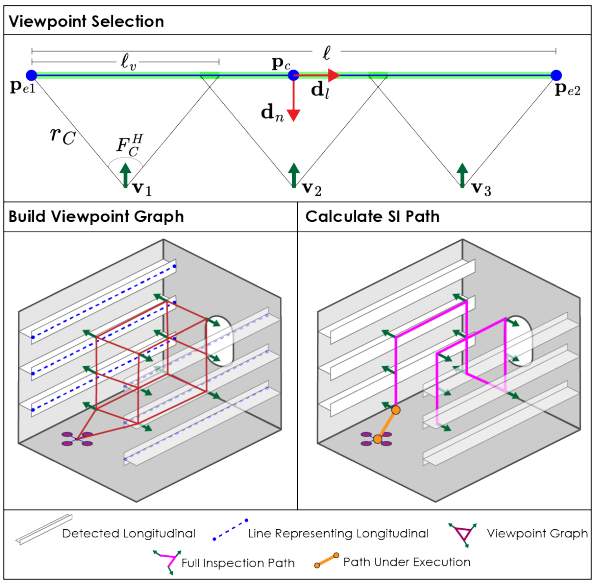


FIGURE 9. **This figure illustrates the steps involved in the Semantic Inspection mode. The planner calculates uniformly distributed viewpoints along each longitudinal respecting the FoV and max range constraints of $\mathcal{Y}_C$. A collision-free graph is built connecting the viewpoints and a tour passing through all viewpoints is calculated. Both of these steps are shown in the bottom two subfigures.**

a point on the line in the inertial frame of reference. In this work, $\mathbf{p}_c$ is selected to be the centroid of the point cloud corresponding to the longitudinal. Let $\mathbf{d}_n$ be the unit vector orthogonal to $\mathbf{d}_l$ pointing along the normal of the wall to which L is attached to. Let $\mathbf{p}_{e1}$ and $\mathbf{p}_{e2}$ be the two endpoints of the longitudinal - both lying on l_L - such that $\mathbf{d}_l$ points from $\mathbf{p}_{e1}$ to $\mathbf{p}_{e2}$. The planner selects equally spaced viewpoints along l_L at a distance $r_C \cos(\frac{F_C^H}{2})$ from it as shown in Figure 9. The number of viewpoints n is given by $n = \lceil \frac{\ell}{\ell_v} \rceil$, where $\ell = |\mathbf{p}_{e2} - \mathbf{p}_{e1}|$ is the length of the longitudinal and $\ell_v = 2r_C \sin(\frac{F_C^H}{2})$ is the length of the section of the longitudinal that a viewpoint will see such that the farthest point viewed is at a distance r_C. Then the position of the viewpoint $\mathbf{v}_i, i = 1...n$ is calculated as

$$\mathbf{v}_i = \mathbf{p}_{e1} + \frac{(2i-1)\ell_v}{2}\mathbf{d}_l + r_C \cos(\frac{F_C^H}{2})\mathbf{d}_n \tag{7}$$

The heading of the viewpoints is set to be along $-\mathbf{d}_n$. The viewpoints that don't view any unseen parts of $\mathbb{M}_S$ or have overlap with other viewpoints are removed. Once the viewpoints for all longitudinals are calculated, the planner attempts to connect them and the current robot location using collision-free straight line edges to form a graph $\mathbb{G}_{SI}$. If any viewpoints are not connected to the graph, extra points are uniformly sampled in the free space to connect the remaining viewpoints to $\mathbb{G}_{SI}$. Next, the tour σ_{tsp} visiting all viewpoints is calculated by solving the Travelling Salesman Problem (TSP) using the Lin-Kernighan-Helsgaun (LKH) [69] heuristic. The travel cost between two viewpoints is given by the length of the shortest path along $\mathbb{G}_{SI}$ between them. The path, along $\mathbb{G}_{SI}$ up to the first viewpoint in the tour is executed by the robot and the viewpoints are recalculated. If any viewpoints are added/deleted or their location has changed more than a threshold ϱ_{thr}, a new graph $\mathbb{G}_{SI}$ is built and a new tour is calculated. Otherwise, the path up to the next viewpoint in the tour is commanded to the robot. Figure 9 shows an illustration of the steps involved in the SI mode.

E. Predictive Planning - Assisted Exploration (PP-AE)

The PP-AE, the first submode of PP, builds upon the VE mode and modifies the information gain to guide the planner to focus on the predicted semantic locations. Using the transform used to align the ϑ^e or $\bar{\vartheta}^e$ of the substructure with the selected ${}^L\vartheta^e$ or ${}^L\bar{\vartheta}^e$, the bounding boxes of the longitudinals in the predicted graph are transformed to be at the predicted longitudinal locations. The part of the map $\mathbb{M}$ lying in these bounding boxes is referred to as the predicted semantic map and is denoted by $\mathbb{M}_S^P$. The PP-AE submode operates in an iterative fashion similar to VE and, in each iteration, builds a collision-free graph $\mathbb{G}_{AE}$ around the current robot location. It then calculates a new information gain $\Upsilon_{AE}(\nu_j)$ for each vertex ν_j in $\mathbb{G}_{AE}$. The remaining procedure to select the best path remains the same as VE with Υ_{VE} being replaced by Υ_{AE} in Equation 6. The new information gain $\Upsilon_{AE}(\nu_j)$, for a vertex ν_j, is formulated as:

$$\Upsilon_{AE}(\nu_j) = (\alpha + \delta)\Upsilon_S(\nu_j) + (1-\alpha)\Upsilon_{VE}(\nu_j). \tag{8}$$

Here, $\Upsilon_{VE}(\nu_j)$ is the Volume Gain as explained in Section V.C. $\Upsilon_S(\nu_j)$, called the Semantic Gain, is defined as the number of unknown voxels in $\mathbb{M}_S^P$ seen by the depth sensor $\mathcal{Y}_D$ if the robot was at vertex ν_j. α, called the overlap ratio, is the ratio of the number of detected longitudinals overlapping with the predicted longitudinals to the number of predicted longitudinals. To calculate the overlap, each predicted longitudinal is checked to find if its bounding box overlaps with any of the detected longitudinals. Two longitudinals are considered to be overlapping if their bounding boxes have more than a user-defined κ percent overlap. The larger the overlap ratio, the higher the contribution of Semantic Gain, leading to the planner focusing more on exploring areas with predicted semantics. The tunable parameter δ $(0 \leq \delta \leq 1)$ enables the planner to have a small bias for exploring predicted semantic areas in the initial few PP-AE planning iterations in each compartment when the overlap ratio is very small or zero.

When the overlap ratio α goes above a user-defined threshold α_{thr}, the PP-AE submode exits and the planner switches to the SI mode to inspect the detected longitudinals. This allows the robot to stop the exploration early on without mapping the entire compartment but after finding the predicted semantics of interest.

F. Predictive Planning - Opportunistic Inspection (PP-OI)

The PP-OI is the second submode for the PP mode. This approach assumes more exact repeatability in the patterns

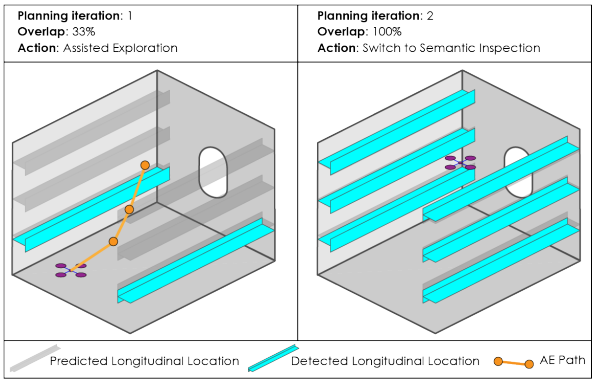


FIGURE 10. **This figure shows illustrations of two key planning steps in the Assisted Exploration (PP-AE) submode. In the planning iteration on the left, the overlap between the detected and predicted semantics is 33% and the planner takes a PP-AE step. In the iteration on the right (not necessarily the immediate next iteration) a 100% overlap between the predicted and detected semantics is seen. Hence the planner switches to the SI mode.**

in the SSG than PP-AE. If the robot traverses through a manhole used for prediction, the PP-OI is triggered. First, the viewpoints $\mathbf{v}_i, i = 1...n$ (n as defined in Section V.D) in the TSP tour σ_{tsp} from the previous compartment are transformed to match the predicted longitudinals. The transform used to align the ϑ^e or $\bar{\vartheta}^e$ of the substructure with the selected ${}^L\vartheta^e$ or ${}^L\bar{\vartheta}^e$ is applied to the viewpoints in σ_{tsp}. Next, for the first viewpoint $\mathbf{v}_i$ in σ_{tsp}, if a longitudinal overlapping with the predicted longitudinal L that $\mathbf{v}_i$ will be inspecting is detected, then a collision-free path from the current robot location to $\mathbf{v}_i$ is calculated using using the RRT* [70] algorithm. The robot traverses this path and the procedure is repeated for the next viewpoint in σ_{tsp}. If such a longitudinal is not detected, an exploration planning step guiding the robot towards $\mathbf{v}_i$ is taken using a modification of the algorithm used for VE mode. The core algorithm remains identical with the only modification being in the function $\mathcal{Z}$ in Equation 6, with each path compared against the straight line path from the current robot location to $\mathbf{v}_i$ instead of σ_e. These exploration steps are repeated until a longitudinal overlapping with L is detected, at which point a collision-free path to $\mathbf{v}_i$ is calculated and the procedure is repeated for the next viewpoint in σ_{tsp}. If the longitudinal is not found within a predefined number of exploration steps, the planner skips $\mathbf{v}_i$ and the procedure is continued for the next viewpoint in σ_{tsp}. Figure 11 illustrates indicative planning steps of the algorithm.

VI. SIMULATION STUDIES

This paper presents two large-scale simulation studies. In the first study, both predictive planning approaches are tested and compared against several exploration and inspection planning methods. We compare the methods in two distinct environments. The first is a large-scale ballast tank consisting of 8 compartments. Each compartment has dimensions length $\times$ width $\times$ height $= 10\,\mathrm{m} \times 10\,\mathrm{m} \times 10\,\mathrm{m}$

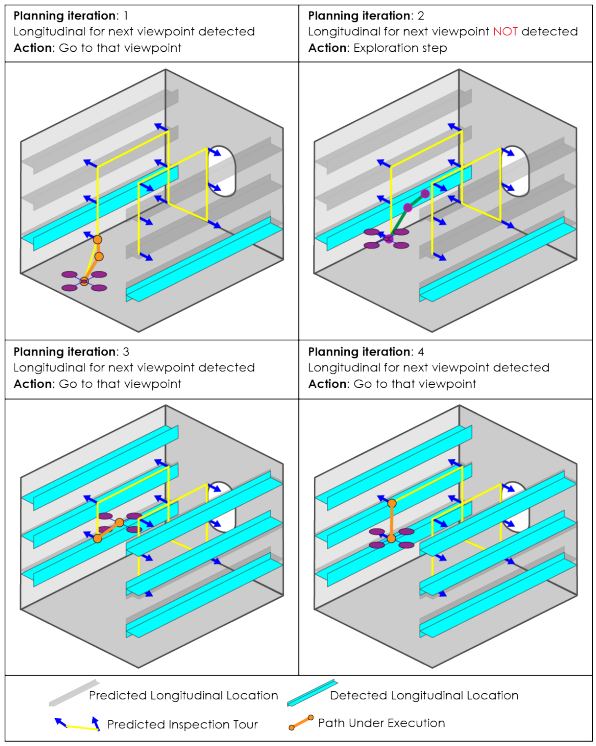


FIGURE 11. **This figure shows four consecutive planning iterations describing the Opportunistic Inspection (PP-OI) submode. As the robot enters through a manhole used for graph prediction, the PP-OI submode is triggered. The robot attempts to follow the predicted path as long as the longitudinals viewed by the viewpoints in the path are detected. If not, then the planner takes exploration steps until that longitudinal is detected.**

with eight longitudinals to inspect. Longitudinals are metal beams mounted on the walls of ballast tanks to enhance structural integrity. An illustration of these components is provided in Figure 12. The compartments are interconnected by large manholes (one between each pair of compartments) of dimensions height $\times$ width $= 2.0\,\mathrm{m} \times 1.5\,\mathrm{m}$ which serve as the EVs. The second environment is an industrial, factory-like setting consisting of several process pipes, which serve as the semantics of interest for inspection. The environment is split into 8 rooms connected by doors of dimensions height $\times$ width $= 2.0\,\mathrm{m} \times 1.5\,\mathrm{m}$ that act as the EVs. Each room contains additional structures, making the environment more cluttered than the ballast tank. The environments and the semantics inside them are shown in Figure 12.

The second study aims to evaluate the performance of the method in the presence of missing semantics and, therefore, inexact graph patterns. This study is conducted in the ballast tank environment. We procedurally remove one longitudinal from a few compartments in the ballast tank, thus ablating over the number of compartments missing a semantic.

The simulations are conducted in the Gazebo simulator with a model of the Resilient Micro Flyer (RMF-Owl) [35] collision-tolerant aerial robot. The method described in Section V.A is used to build the SSG for the simulation in the ballast tank, whereas the segmentation camera provided by Gazebo is used for the factory simulation.

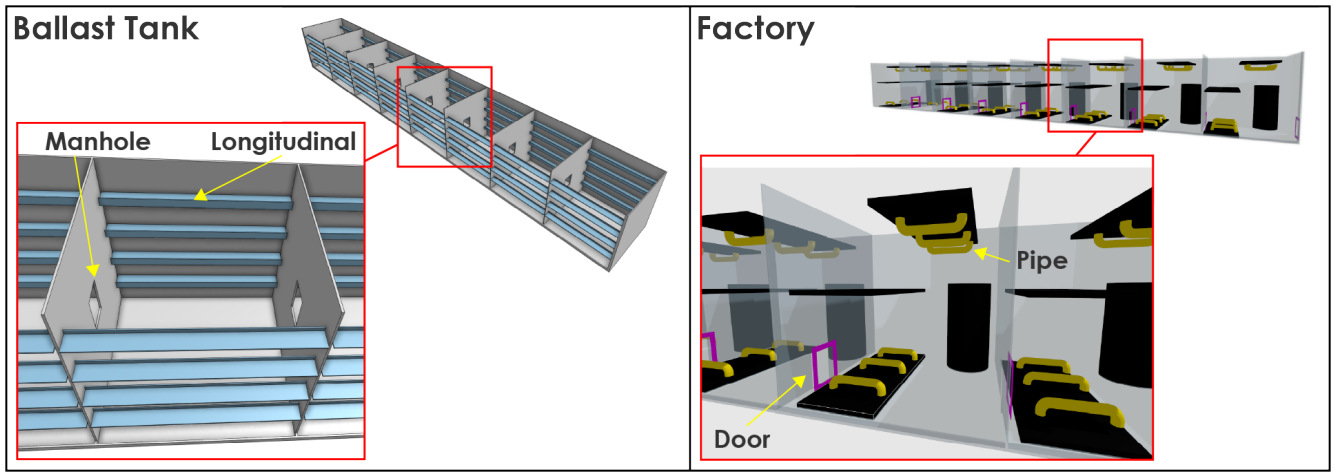


FIGURE 12. **This figure shows both simulation environments and the semantics within them. The first environment is a ballast tank divided into 8 compartments connected by manholes of dimensions height $\times$ width $= 2.0\,\mathrm{m} \times 1.5\,\mathrm{m}$ as the EVs. Each compartment contains 8 longitudinals which are the semantics to inspect. The second environment, shown on the right, is a factory containing several pipes that serve as semantics for inspection. The factory consists of 8 rooms connected by doors of dimensions height $\times$ width $= 2.0\,\mathrm{m} \times 1.5\,\mathrm{m}$ as the EVs.**

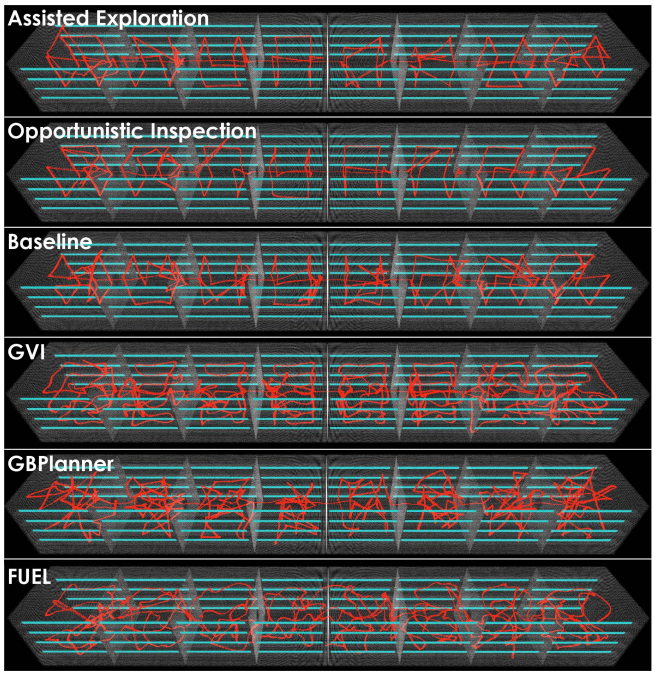


FIGURE 13. **Figure showing the ballast tank simulation environment including the longitudinals along with the robot path from one mission of each planner. The proposed predictive planner significantly outperforms the Baseline, GVI, and pure exploration planners.**

A. Comparative Study

This study compares the performance of our method against the following exploration and inspection path planning strategies.

- **Baseline:** This approach uses the planning strategy detailed in Section V.B without triggering the PP mode and only uses the VE and SI modes. Comparison against this allows us to study the benefit of the predictive planning paradigm against a pure semantics-aware inspection planning approach.
- **GVI:** In this approach, our previously published work on autonomous exploration and General Visual Inspection (GVI) of ballast tanks [71] is utilized. The method performs volumetric exploration along with visual inspection of all mapped surfaces from a given desirable viewing distance. Comparison against this method allows us to study the benefits of using a semantics-aware inspection strategy as opposed to the general inspection of all surfaces.
- **GBPlanner:** We compare against our previous open-sourced graph-based exploration path planner, GBPlanner [1], [13]. To ensure that the planner sees all surfaces, the FoV and the maximum range of the sensor used for exploration are set to be the same as $\mathcal{Y}_C$. This is to compare against a pure exploration, and thus naturally less efficient, method.
- **FUEL:** Finally, another exploration path planner, FUEL [72], that employs a fast frontier-based exploration strategy is tested. The sensor parameters are constrained in the same way as GBPlanner for complete coverage.

As the last two methods do not have an explicit way to navigate through narrow manholes, a large manhole was selected in the ballast tank for the two planners to plan through without the need for explicit detection. Such a size of manhole is not representative of real-life ship ballast tanks but without this environment modification, our evaluation runs indicated that neither GBPlanner nor FUEL would be able to complete exploration if a typical manhole size (e.g., $0.8\,\mathrm{m} \times 0.6\,\mathrm{m}$) was opted for. However, the proposed predictive planner, Baseline, and GVI use the explicit manhole detection and navigation strategy described in [68].

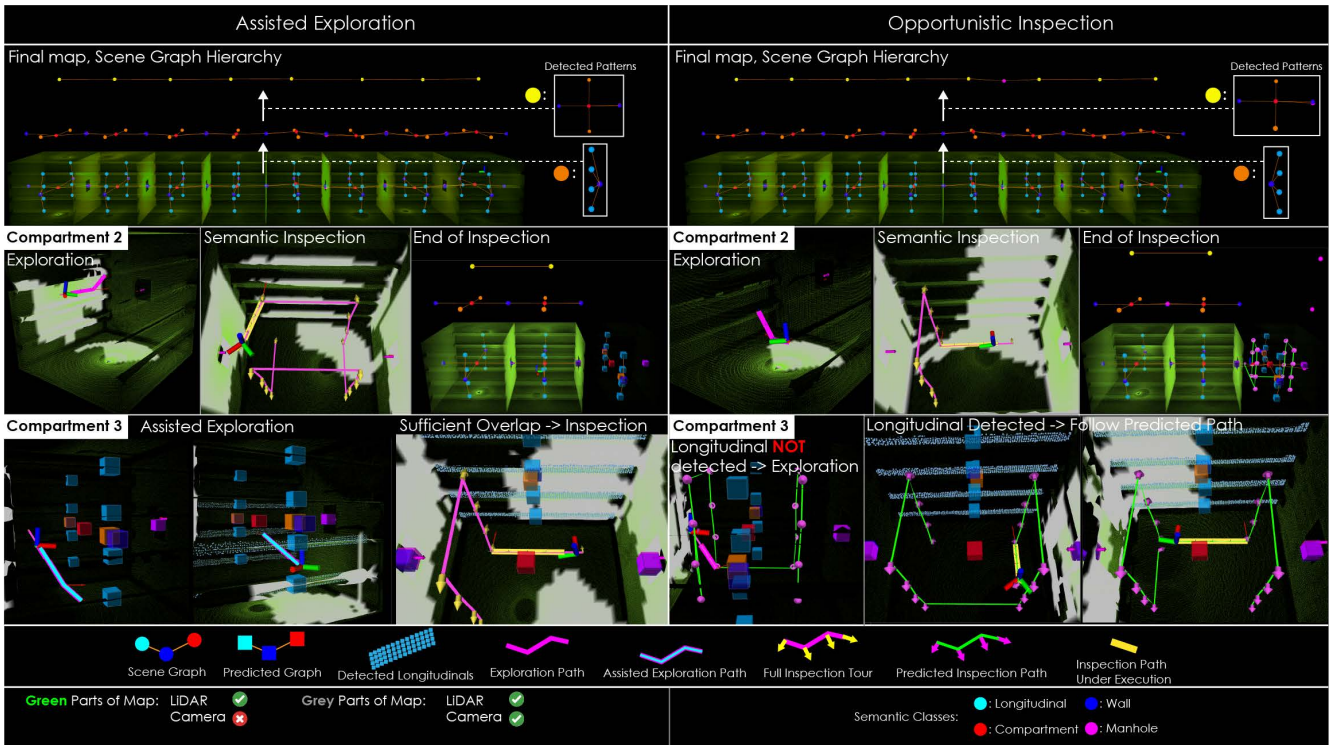


FIGURE 14. **Figure showing indicative steps and final map for the two predictive planning approaches in the ballast tank simulation study. The left half of the figure illustrates the PP-AE submode showing indicative planning paths, scene graph, detected patterns, and predicted graph. The right half subfigure shows the results for the PP-OI submode. The planned paths, detected patterns, predicted graph and paths are presented.The cyan colored point cloud and SSG vertices correspond to the semantics to be inspected.**

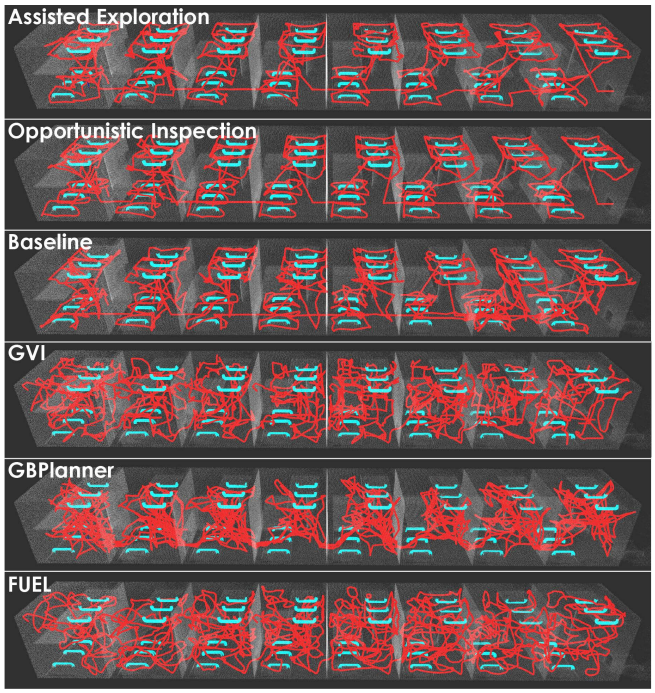


FIGURE 15. **Figure showing the factory simulation environment including the pipes along with the robot path from one mission of each planner. The proposed predictive planner significantly outperforms the Baseline, GVI, and pure exploration planners.**

A total of 5 missions are conducted for each of the planners in both simulation environments. The environment and the robot's path from one mission of each planner are shown in Figures 13, and 15 for the ballast tank and the factory respectively. Figures 14 and 16 show the final map, the SSG at the end of the mission along with the patterns detected and hierarchy built, indicative planning steps, and an indicative graph prediction made during the missions of the predictive planner using the PP-AE and PP-OI submodes for the respective simulations. The parameters used in these missions are listed in Table 1. The quantitative results comparing the performance of each method are shown in Table 2. In the factory environment, the viewpoint sampling strategy of the SI mode is modified to accommodate generic semantics. Viewpoints are sampled within a cuboidal bounding box with dimensions $b_1+r_C\times b_2+r_C\times b_3+r_C$, where $\mathbf{b}=[b_1,b_2,b_3]$ is the bounding box of the semantic under inspection. The remaining procedure to build the graph $\mathbb{G}_{SI}$ and calculate the inspection path remains the same. The metrics used for comparison are the following:

- **Inspection time per compartment / room (s):** This is the average time spent by the robot in each compartment / room. For FUEL and GBPlanner, we calculate this time as the total mission time divided by the number of compartments, since these methods do not explicitly split their planning steps per compartment / room. Note that for PP-AE and PP-OI, we only account

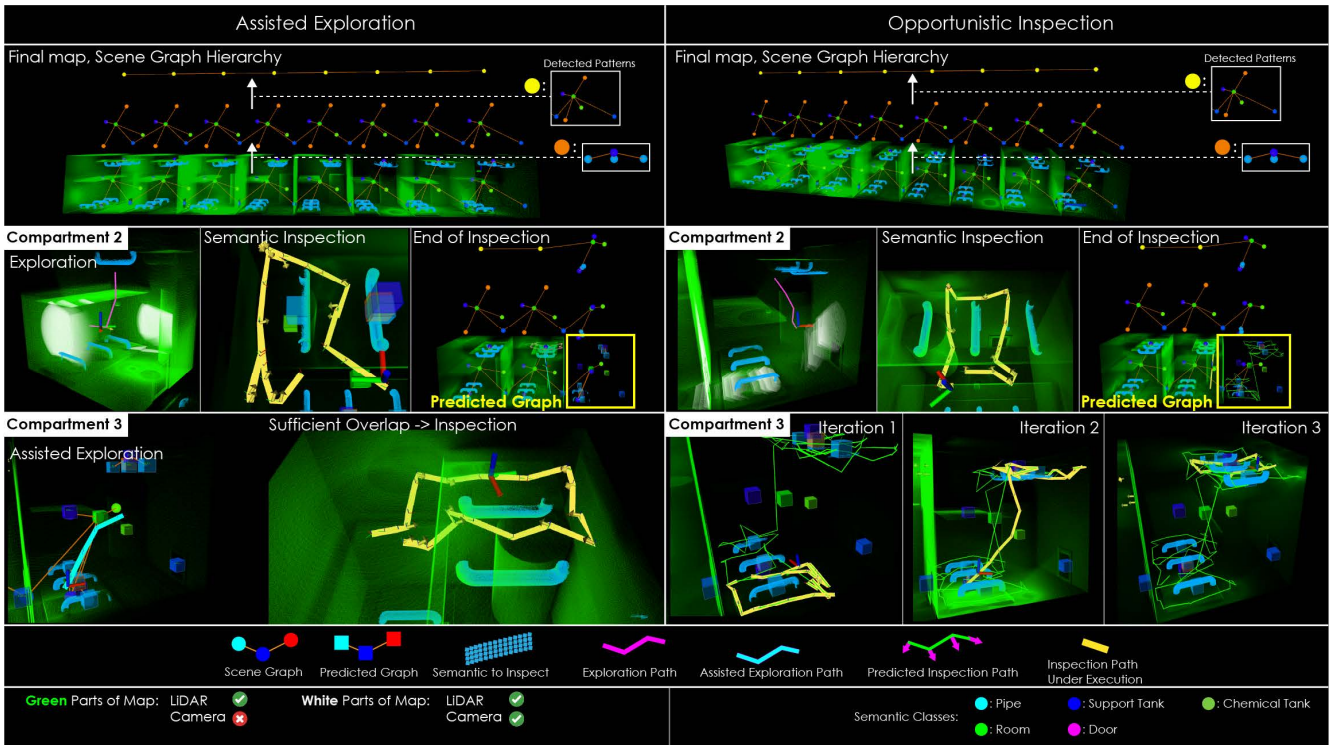


FIGURE 16. **Figure showing indicative steps and final map for the two predictive planning approaches in the factory simulation study. The left half of the figure illustrates the PP-AE submode showing indicative planning paths, scene graph, detected patterns, and predicted graph. The right half subfigure shows the results for the PP-OI submode. The planned paths, detected patterns, predicted graph and paths are presented. The cyan colored point cloud and SSG vertices correspond to the semantics to be inspected.**

for the compartments / rooms in which the respective submode was used, meaning we indicate the effect of exploiting environment prediction. Naturally, the more the compartments / rooms within which prediction can be exploited, the higher the importance of predictive planning.

- **Semantic Surface Coverage (%):** This relates to the percentage of semantic surface seen by the camera sensor $\mathcal{Y}_C$ during the mission at the desired viewing distance r_C. In simulations, we have access to the ground truth mesh of the semantics. For each robot pose in the mission, rays are cast using the camera model $\mathcal{Y}_C$ from that pose. The mesh faces intersected by the rays are accumulated over the entire mission. The semantic surface coverage is then defined as the percentage of the number of cumulative seen mesh faces to the total number of faces in the mesh.
- **Path length (m):** The total distance traveled by the robot in a mission.

As shown in Table 2, the proposed predictive planning submodes outperform the rest in terms of inspection time per compartment/room while achieving comparable (often higher) semantic surface coverage. Specifically, the PP-AE and PP-OI submodes outperform the Baseline by 25% and 38%, GVI by 45% and 54%, and the exploration planners by 52% and 60% in the ballast tank environment. In the factory environment, they outperform the Baseline by 27% and

Parameter	Value
SUBDUE Related Parameters	
γ_b	3
γ_l	30
t_{thr}	0.2
Vertex/Edge Addition Cost	1.0
Vertex/Edge Deletion Cost	1.0
Vertex Label Substitution Cost	4.0
Edge Label Substitution Cost	1.0
γ_p	1.0
$d_{\min}$	0.5 m
$d_{\max}$	4.0 m
Planning Parameters	
$[F_D^H, F_D^V]$	$[360, 90]^\circ$
$[F_C^H, F_C^V]$	$[90, 60]^\circ$
r_C	3.0 m
μ	0.05
ζ	0.01
δ	0.2
α_{thr}	0.9
Max speed $v_{\max}$	2.0 m/s

TABLE 1. **Parameters used in the presented Simulation Study**

Method	Inspection Time per Compartment / room (s)	Semantic Surface Coverage(%)	Path Length (m)	Computation Time (ms)
Comparative Study 1 : Ballast Tank				
PP-AE	48.97	98.96	528.47	148.81
PP-OI	40.33	99.93	522.12	31.58
Baseline	65.58	99.87	645.69	277.51
GVI	88.98	99.46	976.59	2046.46
GBPlanner	103.06	96.83	1068.35	102.61
FUEL	101.79	98.44	867.99	42.23
Comparative Study 2: Factory				
PP-AE	255.48	98.26	1376.3288	125.43
PP-OI	214.40	98.72	1356.0789	46.93
Baseline	349.35	98.65	1438.0326	165.90
GVI	565.89	91.25	2539.0344	2666.89
GBPlanner	622.46	73.06	3187.9088	130.29
FUEL	629.98	79.70	2618.4168	38.33

TABLE 2. Quantitative Results from Comparative Simulation Study. It can be clearly observed that the two predictive planning submodes PP-AE and PP-OI outperform the other methods due to their ability to exploit pattern prediction. Even if the Baseline approach uses the semantic information for inspection, it significantly underperforms compared to PP-AE and PP-OI. The average computation times for each method are shown in the last column. For the PP-AE, PP-OI the computation time is reported only for the planning steps when the planner was operating in the respective submodes.

38%, GVI by 55% and 62%, and the exploration planners by 59% and 66%. This highlights the benefit of using the predictive planning approach. It is noted that the semantics in the factory environment require the robot to access narrow parts between the walls and the pipes. Since the exploration planners only aim to map the given volume, these parts of the semantics are not seen properly. Hence, the two exploration planners have lower semantic surface coverage in the factory environment as compared to the ballast tank.

B. Ablation Study for Missing Semantics

This study aims to evaluate the performance of the proposed method under the circumstances of imperfect patterns. We create 5 versions of the ballast tank shown in Figure 13 where in the k^{th} version, one longitudinal is removed from k compartments. The purpose of this study is twofold. First, to evaluate the ability of the modified SUBDUE algorithm in the presence of imperfect patterns and compare its performance against the original SUBDUE algorithm. Second, to demonstrate the ability of both Predictive Planning submodes, PP-AE and PP-OI, to handle imperfect patterns. To this end, the PP-AE and PP-OI submodes, as well as the Baseline are tested in all versions of the ballast tank. The original SUBDUE algorithm is tested on the SSG of the full tank of each version, and the results are compared with the modified version. Note that the exploration planners and the GVI are not tested as they do not explicitly account for semantics, therefor their performance will not be affected in any meaningful way by the missing semantics.

Figure 17 presents the patterns detected and the hierarchy built for each version of the ballast tank and the quantitative results of this study are shown in Table 3. The left side fig-

Method	Inspection Time / Compartment(s)	Semantic Surface Coverage(%)	Path Length (m)
Number of compartments missing a longitudinal: 1			
PP-AE	47.5097	99.8947	514.2059
PP-OI	39.4998	99.9659	531.9714
Baseline	64.7500	99.576	618.2521
Number of compartments missing a longitudinal: 2			
PP-AE	45.5292	98.5946	515.5791
PP-OI	42.5706	98.8924	540.3622
Baseline	62.3510	99.5951	613.6175
Number of compartments missing a longitudinal: 3			
PP-AE	47.1471	98.6304	518.9256
PP-OI	43.6990	99.0683	538.8774
Baseline	62.4917	98.6113	635.8960
Number of compartments missing a longitudinal: 4			
PP-AE	46.0173	99.7938	514.2457
PP-OI	45.5803	98.9333	552.1726
Baseline	63.9893	98.9602	610.8366
Number of compartments missing a longitudinal: 5			
PP-AE	44.6311	99.8967	510.0772
PP-OI	46.515	99.3527	571.7858
Baseline	62.5788	98.59	627.1848

TABLE 3. Ablation Study for Missing Semantics. We evaluate the two predictive planning submodes and the Baseline in ballast tanks with varying number of compartments missing one longitudinal each. The results show that the proposed planner is able to handle the imperfect patterns successfully showing significant improvement over the Baseline.

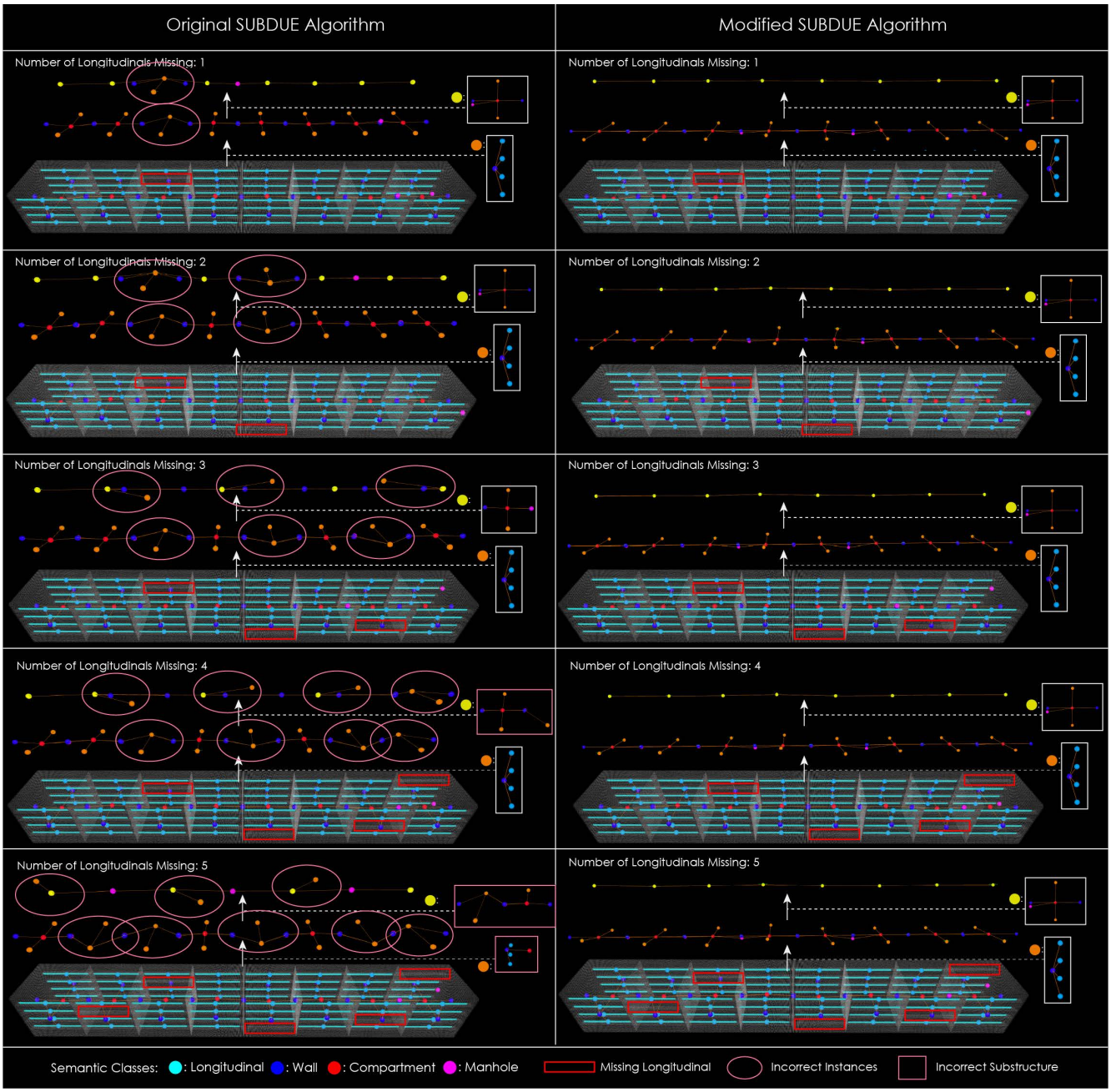


FIGURE 17. **Figure showing the 5 versions of the simulation environment with inexact patterns along with the detected patterns and hierarchies of compressed graphs. The left side figures show the original SUBDUE algorithm tested in each version of the ballast tank. It can be clearly seen that the modified algorithm was able to identify the correct patterns but the original algorithm did not.**

ures of Figure 17 show the results of the original SUBDUE algorithm and those on the right side show the modified one. It is evident that the modified SUBDUE algorithm was able to identify the correct patterns despite the missing semantics in all tests, while the original algorithm failed to do so. The original algorithm, at instances, incorrectly grouped the 'Compartment' vertex in the instances of the pattern with missing longitudinals. As the modified algorithm accounts for the pose of the vertices, the incorrect grouping of the 'Compartment' vertex is penalized. Furthermore, the penalty for incorrectly mapped high-degree vertices (such as the 'Compartment') leads to a higher graph-matching cost for this incorrect mapping.

Despite the missing semantics, both Predictive Planning submodes were able to provide comparable semantic coverage as the missions with perfect patterns. This shows the ability of the planner to handle imperfect patterns. At the same time, they outperformed the Baseline by more than 25% in terms of inspection time, a result that is comparable to the case of perfect patterns. For the PP-OI submode, the

average inspection time per compartment increases slightly as the number of compartments with missing longitudinals increases. This can be attributed to the fact that the robot needs to perform additional exploration steps in the compartments with missing longitudinal to search for longitudinals at the predicted locations. A slight decrease can be observed in the inspection time per compartment for the PP-AE and the Baseline solutions as the robot needs to inspect fewer semantics. However, in all cases, the changes are small and the Predictive Planning submodes show a significant advantage over the Baseline.

VII. FIELD EXPERIMENTS

To demonstrate the field readiness of the proposed SPP paradigm, we present extensive field deployments in real ballast tanks inside two oil tanker ships. The deployments were conducted in the side section of the ballast tanks (the point cloud map is shown in Figure 1) using a variant of the Resilient Micro Flyer (RMF)-Owl [35] collision-tolerant aerial robot. In both deployments, three missions were conducted testing the PP-AE and PP-OI submodes along with the Baseline approach described in Section VI. The performance of the methods is compared using the metrics detailed in Section VI. Since we do not have access to ground truth meshes for the semantic surface coverage calculation, we create the meshes using the segmented point cloud. As the longitudinals can be approximated by rectangles, simplified rectangle-shaped meshes were created using the dimensions given by the segmented point cloud. The parameters used by the predictive planner in both field deployments are listed in Table 4. The dataset from both deployments will be open-sourced soon extending our previously released ballast tank dataset available at https://github.com/ntnu-arl/ballast_water_tank_dataset.

Parameter	Value
SUBDUE Related Parameters	
γ_b	3
γ_l	30
t_{thr}	0.2
Vertex/Edge Addition Cost	1.0
Vertex/Edge Deletion Cost	1.0
Vertex Label Substitution Cost	4.0
Edge Label Substitution Cost	1.0
γ_p	1.0
$d_{\min}$	$0.5\,\mathrm{m}$
$d_{\max}$	$4.0\,\mathrm{m}$
Planning Parameters	
$[F_D^H, F_D^V]$	$[360, 90]^\circ$
$[F_C^H, F_C^V]$	$[85, 64]^\circ$
r_C	$1.0\,\mathrm{m}$
μ	0.05
ζ	0.01
δ	0.2
α_{thr}	0.9
Max speed $v_{\max}$	$1.5\,\mathrm{m/s}$

TABLE 4. Parameters used in Field Deployments

A. Collision-tolerant Aerial Robot System Description

All experiments are conducted using an upgraded version of the collision-tolerant aerial robot RMF-Owl presented in [35]. With its collision-tolerant frame, the robot is designed for autonomous navigation in confined environments. The robot presents a small form factor of length $\times$ width $\times$ height $= 0.38\,\mathrm{m} \times 0.38\,\mathrm{m} \times 0.24\,\mathrm{m}$ weighing $1.45\,\mathrm{kg}$. It utilizes a 4s 5000mAh battery providing an endurance of 10 min. The robot carries a Khadas VIM4 Single Board Computer (SBC) having $\times 4$ 2.2Ghz Cortex-A73 cores, along with $\times 4$ 2.0Ghz Cortex-A53 cores implementing an Amlogic A311D2 big-little architecture. The entire autonomy software stack including SLAM, position control and predictive planning runs onboard this computer. The SBC interfaces with a Pixracer autopilot for low-level control. The sensing suite of the robot contains an Ouster OS0-64 LiDAR used as $\mathcal{Y}_D$ (FoV: $[360, 90]^\circ$, max range: $100\,\mathrm{m}$), a Blackfly S color camera used as $\mathcal{Y}_C$ (Resolution: 720×540, FoV: $[85, 64]^\circ$), and a VectorNav VN100 IMU. This computing and sensing suite allows the robot to maintain a lightweight and small form factor while providing autonomous exploration and inspection capabilities in confined environments. The robot uses CompSLAM [73], a multi-modal SLAM framework, for accurate odometry and mapping. Additionally, a Model Predictive Controller [74] is used for tracking trajectories given by the proposed Predictive Planner.

B. Field Deployment 1

The first evaluation was conducted in an oil tanker, and the robot was deployed inside the side section of one of its ballast tanks. A point cloud map of the tank can be seen in Figure 18. Each compartment of the selected section had the dimensions length $\times$ width $\times$ height $= 4.8\,\mathrm{m} \times 2.9\,\mathrm{m} \times 5.5\,\mathrm{m}$. The compartments were each connected by two manholes of dimensions height $\times$ width $= 0.8\,\mathrm{m} \times 0.6\,\mathrm{m}$ and $0.4\,\mathrm{m} \times 0.6\,\mathrm{m}$ located at a height of $0.8\,\mathrm{m}$ and $4.3\,\mathrm{m}$ respectively from the ground. Each compartment had a total of 12 longitudinals, however, for safety reasons the maximum operating height of the robot was limited to $4.5\,\mathrm{m}$ reducing the scope of inspection to 10 longitudinals per compartment. Three missions were conducted testing the PP-OI and PP-AE submodes, as well as the Baseline. In each mission, the robot was tasked to explore and inspect the longitudinals in four compartments. It is highlighted that the Predictive Planner did not have access to any information about the ballast tank –including the compartment dimensions, number of longitudinals, etc– other than the number of compartments to inspect. Subsequently, we describe each mission in detail. A video detailing Field Deployment 1 is available at **https://youtu.be/XhT3nVC9d-I**.

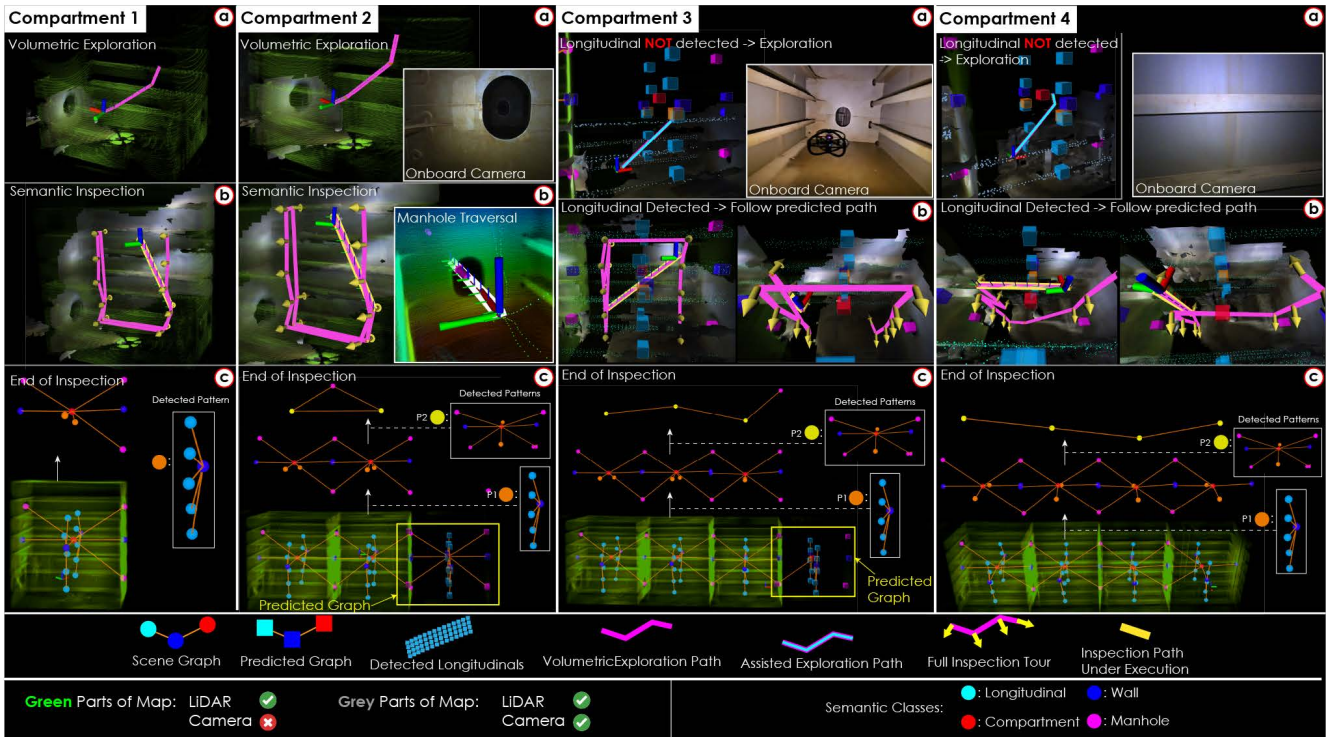


FIGURE 18. **Field Deployment 1: Mission 1. The figure shows the scene graph built, paths planned, and patterns found in each compartment. The robot performed VE and SI steps in the first two compartments after which a successful graph prediction was made. The robot operated in PP-AE submode in compartments 3 and 4. The planner was able to find correct patterns and utilize the prediction for efficient exploration and inspection.**

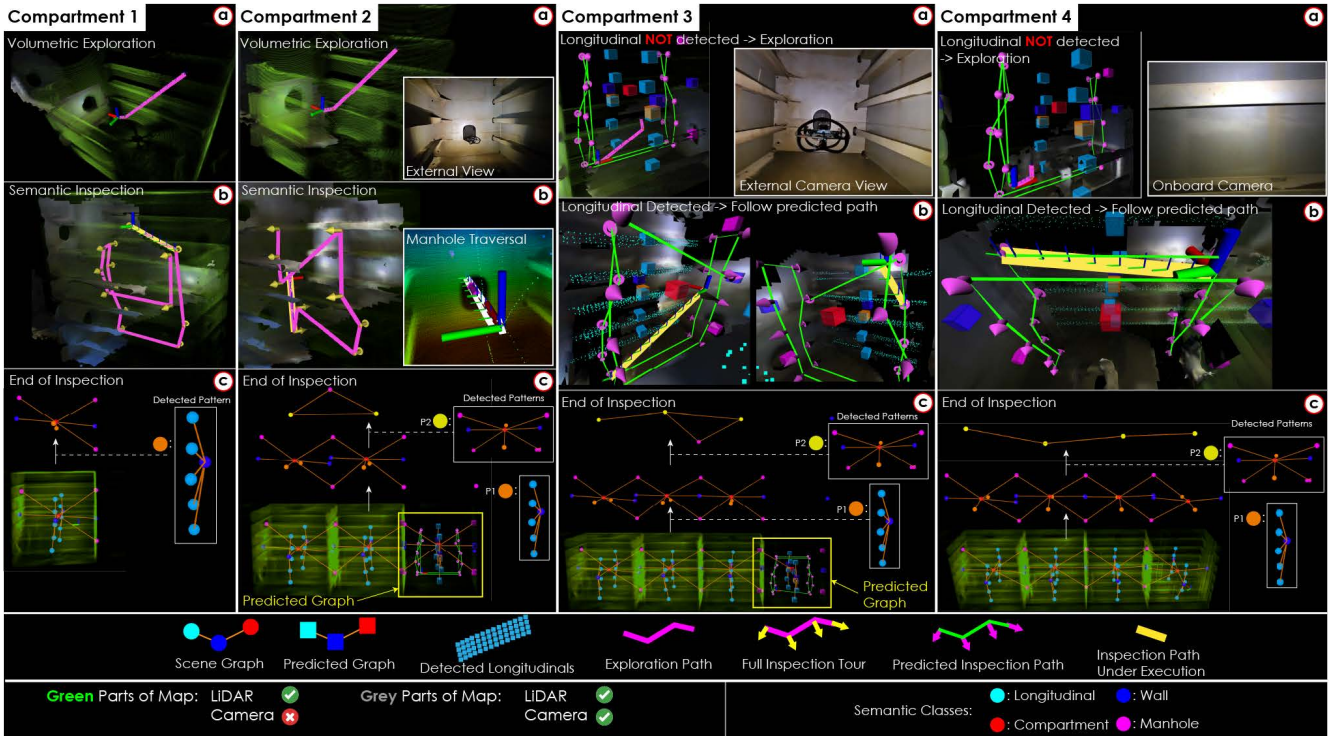


FIGURE 19. **Field Deployment 1: Mission 2. After performing exploration and inspection in the first two compartments without any graph prediction, a successful prediction was made at the end of the inspection of the second compartment. The planner then operated in the PP-OI submode in the 3^{rd} and 4^{th} compartments. The predicted graph and paths, along with indicative planning steps, are shown in the figure.**

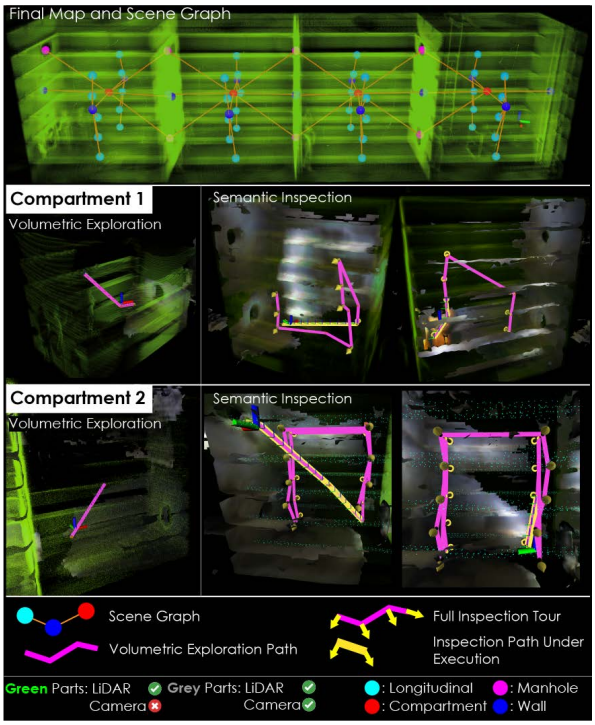


FIGURE 20. Field Deployment 1: Mission 3. The figure shows indicative planning steps, final map, and scene graph built during this mission. The Baseline approach was used in this mission hence no pattern detection or graph prediction was made.

1) Mission 1: Assisted Exploration

In this mission, the robot started in the first compartment in the VE mode and then switched to the SI mode upon exploring the entire compartment (the definition of compartment here is as per Section V.A). Once the compartment was inspected, the pattern detection and prediction step was triggered. The detected pattern consisted of five longitudinal vertices connected to a wall vertex as shown in Figure 18 (Compartment 1.c). As the pattern had no ϑ^e, the robot navigated through the nearest untraversed manhole and switched to the VE mode for exploration followed by SI for longitudinal inspection. Upon completion, the pattern detection step was triggered and a two-level hierarchical structure was built as shown in Figure 18 (Compartment 2.c). The pattern found in the first level (referred to as 'p1' and denoted by the orange circle) consists of five longitudinal vertices connected to a wall vertex. The pattern in the second level (referred to as 'p2' and denoted by the yellow circle) consists of a compartment vertex connected to two wall vertices, two vertices representing the substructure 'p1', and four manholes. As the substructure 'p2' contained ϑ^e (the manholes), the graph prediction step was triggered and the prediction made by the planner is shown in Figure 18 (Compartment 2.c).

Given this prediction, the robot traversed through the manhole used for the prediction, entered the next compartment, and switched to the PP-AE submode. The planner performed PP-AE steps until the overlap ratio $\alpha \geq \alpha_{thr}$. At which point the planner switched to the SI mode to inspect the longitudinals. In this compartment, only 1 PP-AE step was conducted. After finishing the inspection, the pattern detection and prediction step was triggered. A hierarchy of two levels was built with similar substructures found at the end of the inspection of Compartment 2. The robot traversed through the manhole used for graph prediction, switched to the PP-AE submode, and continued the mission. Indicative planning steps in compartments 3 and 4 are shown in Figure 18 (Compartment 3) and (Compartment 4) respectively. The total mission time in this case was 279 s.

2) Mission 2: Opportunistic Inspection

Similar to Mission 1, the robot started in the first compartment in the VE mode. The behavior of the planner in the first two compartments was similar to that in Mission 1. At the end of the inspection of the second compartment, the pattern detection step was triggered and a two-level hierarchy shown in Figure 19 (Compartment 2.c) was built. As the pattern 'p2' contains ϑ^e, a graph prediction was made. The predicted graph extension and the predicted path can be seen in Figure 19 (Compartment 2.c).

The robot navigated through the manhole used for prediction, entered the next compartment, and switched to the PP-OI submode. In the first planning iteration, no longitudinal was detected, so the planner performed an exploration step as shown in Figure 19 (Compartment 3.a). At this point, the longitudinal corresponding to the first viewpoint in the predicted path was detected and the robot planed a path towards that viewpoint. The robot continued the mission visiting each viewpoint in the predicted path. Two indicative steps are shown in Figure 19 (Compartment 3.b). Upon inspecting Compartment 3, the pattern detection and prediction step was triggered and patterns similar to those at the end of Compartment 2 were detected. The robot traversed through the relevant manhole and continued the mission. Indicative planning steps in Compartment 4 are shown in Figure 19 (Compartment 4). The total mission time in this mission was 258 s.

3) Mission 3: Baseline

The Baseline approach, as described in Section VI, was used for comparison. The planner does not use the pattern detection and graph prediction steps in this mission. In each compartment, the robot started in the VE mode, switched to SI mode when fully explored, and traversed through the closest manhole upon completing the inspection. The final map and the scene graph built during the mission are shown in Figure 20 along with indicative VE and SI paths. The total mission time was 318 s.

The performance of all methods is evaluated using the metrics presented in Section VI. The quantitative results are

shown in Table 5 (Field Deployment 1). It is clear from the table that the proposed predictive planning submodes outperform the Baseline in terms of inspection time per compartment and path length while maintaining comparable semantic surface coverage. Note that the inspection time for the PP-AE and PP-OI submodes considers only the last two compartments in which the two submodes were used. This demonstrates that the predictive planning approach is more effective even in real-world scenarios.

C. Field Deployment 2

The second field deployment was conducted in a ballast tank of another oil tanker. The robot was deployed in the side section of the tank. This tank presents a layout similar to that in the first deployment but with a different internal structure. This deployment further shows the repeatability of the method in real-world scenarios. Each level of the ballast tank had compartments of dimensions length $\times$ width $\times$ height $= 4.8\,\mathrm{m} \times 2.9\,\mathrm{m} \times 5.5\,\mathrm{m}$. The compartments were connected by manholes of dimensions height $\times$ width $= 0.8\,\mathrm{m} \times 0.6\,\mathrm{m}$ and $0.4\,\mathrm{m} \times 0.6\,\mathrm{m}$. Similar to Field Deployment 1, we tested the PP-AE, PP-OI, and the Baseline solutions. The robot was tasked to inspect 4 compartments in each mission. The maximum flying height was limited to $4.5\,\mathrm{m}$ allowing the inspection of 10 longitudinals in each compartment. A video detailing Field Deployment 2 is available at **https://youtu.be/7Efu8N33XqU**.

1) Mission 1: Assisted Exploration

The planner started in the first compartment in the VE mode, switched to SI mode when fully explored and upon completion, triggered the pattern detection. The detected pattern consisted of one wall vertex connected to five longitudinal vertices as shown in Figure 21 (Compartment 1.c). Since the substructure did not contain ϑ^e, the robot traversed to the next compartment through the closest manhole and switched to the VE mode. At the end of the inspection, the pattern detection step was triggered and a two-level hierarchical graph structure was built as shown in Figure 21 (Compartment 2.c). The pattern 'p1' (orange circle) in the first level consisted of one wall vertex connected to five longitudinal vertices, whereas 'p2' in level 2 (yellow circle) consisted of one compartment vertex connected to two wall vertices, two 'p1' vertices, and four manholes. As 'p2' contained ϑ^e, the graph prediction step was triggered and the predicted graph is shown in Figure 21 (Compartment 2.c).

The robot navigated to the next compartment through the manhole used for graph prediction and switched to the PP-AE submode. Two PP-AE steps were conducted (shown in Figure 21 (Compartment 3.a)) until sufficient overlap between the detected and predicted longitudinals was achieved, at which point the planner switched to the SI mode. Upon completion, the pattern detection step was triggered. Similar substructures as at the end of Compartment 2 were detected and the graph prediction was carried out(Figure 21 (Compartment 3.c)). The robot navigated to the next compartment, passing through the relevant manhole, and the process was continued. Figure 21 shows indicative planning steps in each of the compartments. The total mission time was $276\,\mathrm{s}$.

2) Mission 2: Opportunistic Inspection

During Mission 2, the behavior of the predictive planner in the first two compartments was similar to that in Mission 1. Upon completing the inspection in the second compartment, the pattern detection step was triggered. The hierarchy built was similar to that in Mission 1 as shown in Figure 22 (Compartment 2.c). The graph prediction step was carried out and the predicted graph and predicted inspection path are shown in Figure 22 (Compartment 2.c). The robot navigated through the relevant manhole to enter the next compartment and the PP-OI submode was activated. As no longitudinals were detected at the beginning of the first iteration, the planner took an exploration step (Figure 22 (Compartment 3.a). At this point, the longitudinal corresponding to the next viewpoint in the predicted path was detected and the planner planed a path to that viewpoint. This process was repeated until all viewpoints in the predicted path were visited. Next, the pattern detection and graph prediction steps were carried out. After traversing through the relevant manhole, the PP-OI submode was triggered as successful graph prediction was made. Figure 22 shows indicative planning steps for each compartment. The total mission time in this case was $267\,\mathrm{s}$.

3) Mission 3: Baseline

The Baseline approach showed similar behavior as that in Field Deployment 1. In each compartment, the planner performed the VE and SI steps without triggering the pattern detection and graph prediction steps. The final map, scene graph, and indicative planning steps in two compartments are shown in Figure 23. The total mission time was $314\,\mathrm{s}$.

The quantitative results comparing the performance of the three missions are shown in Table 5 (Field Deployment 2). Similar to Field Deployment 1, the predictive planning submodes outperformed the Baseline, with the PP-OI having the best performance in terms of inspection time per compartment.

D. Summary

We presented field deployments in the ballast tanks of two ships demonstrating the field readiness of our method. The planner operating in both predictive planning submodes was tested along with the Baseline solution that does not exploit pattern detection and prediction. The PP-AE and PP-OI submodes show an improvement in the inspection time per compartment of 17.1%, 22.6% respectively in

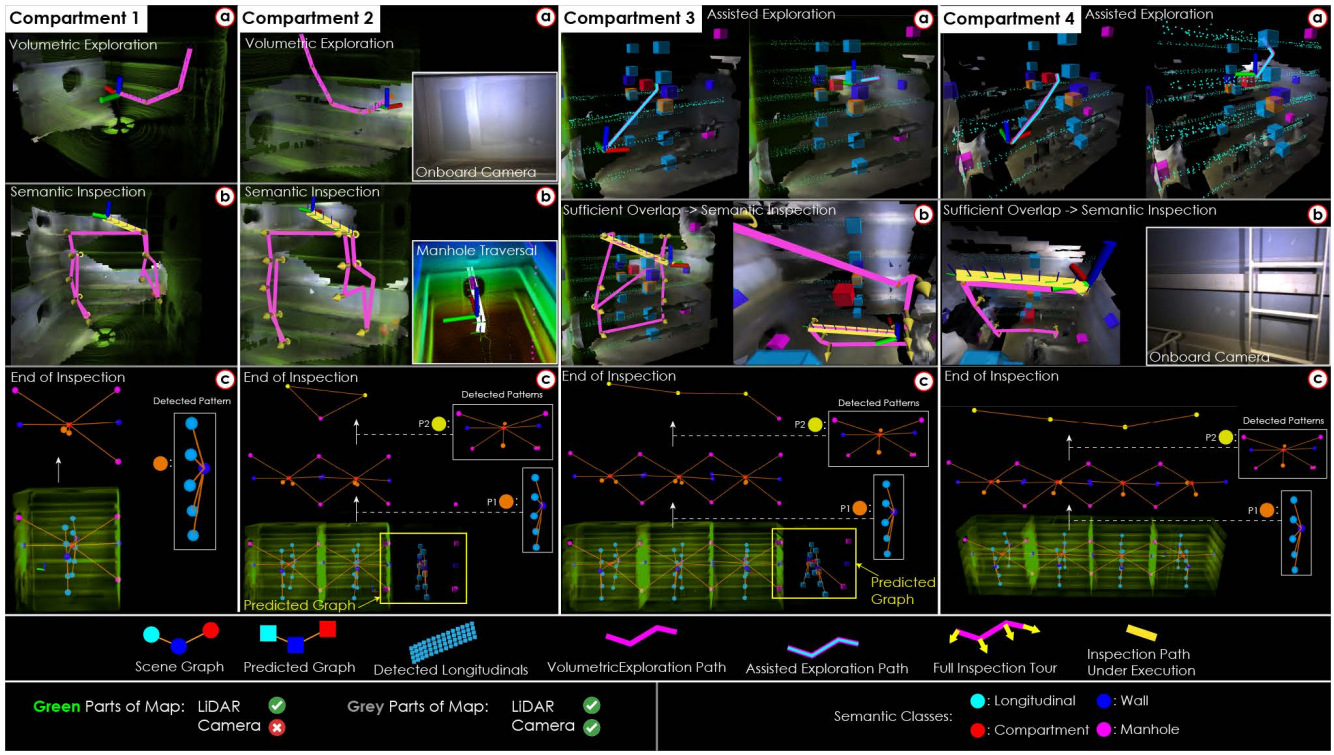


FIGURE 21. **Field Deployment 2: Mission 1. The robot started in the first compartment in the VE mode. As no graph prediction was made in the first two compartments, the PP mode was not used. After successful prediction at the end of the second compartment, the PP-AE submode was used in Compartment 3. The same behavior was observed in Compartment 4. The PP-AE paths, scene graphs, and maps can be seen in the figure.**

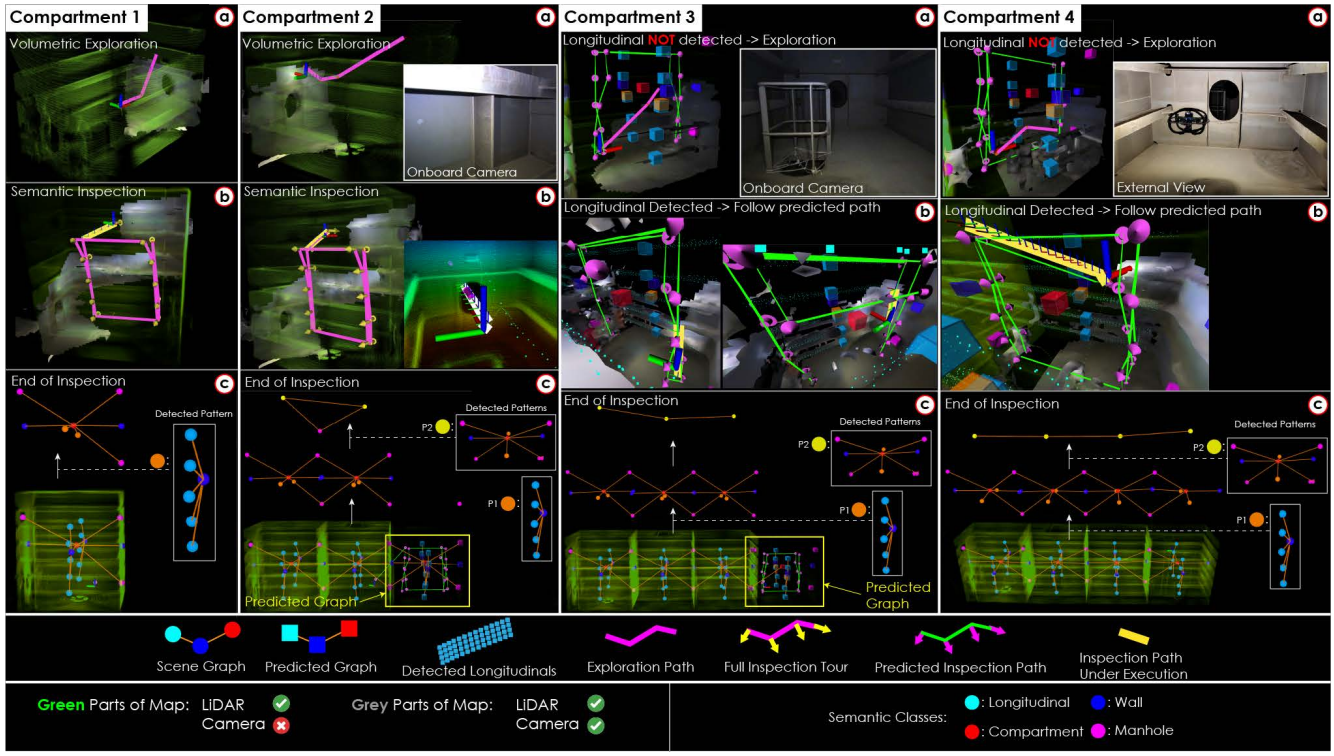


FIGURE 22. **Field Deployment 2: Mission 2. The figure shows indicative planning steps from each compartment. During the four-compartment mission, no graph prediction was made in the first two compartments and only VE and SI modes were used. At the end of the inspection of the second compartment, a successful graph prediction was made and the planner switched to PP-OI submode upon entering the third compartment. Similarly the planner operated in the PP-OI submode in the fourth compartment as well.**

Method	Inspection Time / Compartment (s)	Semantic Surface Coverage (%)	Path Length (m)	Computation Time (ms)
Field Deployment 1				
PP-AE	61.0	97.4603	119.78	270.54
PP-OI	57.0	97.3727	115.77	63.91
Baseline	73.5	96.3218	129.17	426.62
Field Deployment 2				
PP-AE	58.5	95.9184	128.55	245.02
PP-OI	55.5	95.3719	125.96	71.29
Baseline	72.0	93.9935	131.55	414.75

TABLE 5. Quantitative results from both Field Deployments. We deploy the two predictive planning submodes and the Baseline in two ships. The results show that the proposed submodes outperform the Baseline in terms of inspection time per compartment while achieving equal or better semantic coverage in both deployments. The average computation times for each method are shown in the last column. For the PP-AE, PP-OI the computation time is reported only for the planning steps when the planner was operating in the respective submodes.

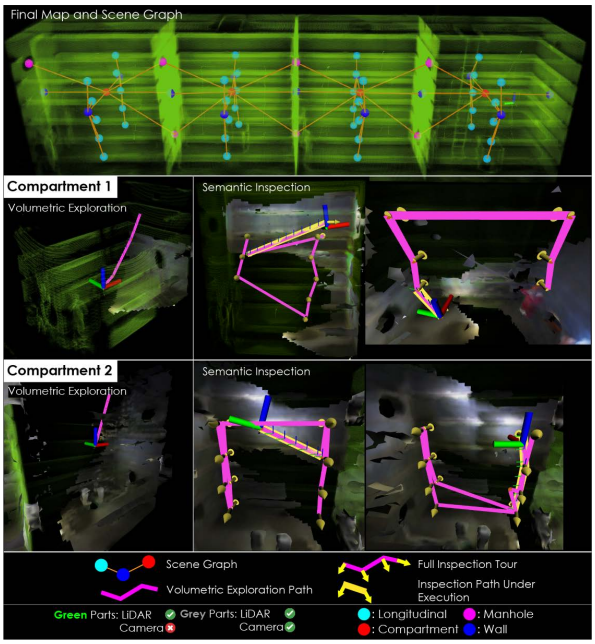


FIGURE 23. Field Deployment 2: Mission3. The figure shows indicative planning steps for the Baseline solution. The planned paths, scene graph, and final map are shown in the figure. It is highlighted that the Baseline does not do pattern detection and graph prediction thus using only the VE and PP-OI modes.

Field Deployment 1 and 19.4%, 23.6% respectively over the Baseline. The deployments show that the semantics-aware predictive planning paradigm proposed in this paper can make the inspection planning more efficient while obtaining the same semantic coverage. Among the two predictive planning approaches presented in the paper, the PP-OI submode provides marginally better results than PP-AE as shown by the quantitative results aligning with the trend seen in the simulation study.

VIII. CONCLUSIONS

This paper presents a new paradigm called "Semantics-aware Predictive Planning" that exploits structured spatial semantics patterns in an environment. Specifically, we represent the semantics in the environment as a Semantic Scene Graph. The method exploits and extends a graph structure discovery algorithm called SUBDUE. Key modifications are made to the original algorithm for handling the case of inexact patterns more accurately and for utilization of the pose information of the semantics in the SSG. Using these patterns, the work proposes a strategy to identify areas in the SSG that can extend to the detected patterns and predicts the evolution of the SSG. Finally, we present two inspection path planning strategies that exploit these graph predictions, and tailor them to the application of ship ballast tank inspection. Through simulation studies in a model of a ballast tank and and industrial factory-like environment, the paper demonstrates that the proposed predictive planning paradigm can perform semantic inspection faster (ranging from 25% to 60% less inspection time) than state-of-the-art exploration and inspection planners as well as a semantics-aware "Baseline" solution that used the same volumetric exploration and semantic inspection planners without exploiting the graph predictions, while maintaining equal or higher semantic coverage. Furthermore, we show the benefit of the proposed modifications to SUBDUE through a simulation study in a ballast tank having imperfect patterns. The method is deployed in the ballast tanks of two oil tanker ships onboard a collision-tolerant aerial robot. Both the proposed predictive planning strategies along with the Baseline are tested resulting in a total of 6 experiments. The proposed approaches are able to achieve up to 23% improvement over the Baseline showing the real-world applicability of the method.

Regarding future work, we have identified four possible avenues. The first relates to a probabilistic formulation of the pattern detection strategy. Currently, the output of the pattern detection algorithm is independent of the previous times it was triggered. Furthermore, it does not account for the detection probabilities of the semantics. Future work can focus on an algorithm that can account for the above and

provide a confidence metric over the detected patterns and track them over time. The second future research direction can be to include prior knowledge about patterns in both the pattern detection as well as graph prediction steps. Third, investigation can happen on graph prediction strategies that do not restrict the prediction to the use of specific entry classes. This will enable this paradigm to be used in a larger variety of environments. Finally, research can focus on path planning for actively searching for patterns based on prior knowledge or graph predictions.

REFERENCES

[1] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, vol. 37, no. 8, pp. 1363–1388, 2020.

[2] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund *et al.*, "Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge," *arXiv preprint arXiv:2103.11470*, 2021.

[3] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel and R. Siegwart, "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6423–6430. [Online]. Available: https://github.com/ethz-asl/StructuralInspectionPlanner

[4] C. Cao, H. Zhu, H. Choset, and J. Zhang, "Tare: A hierarchical framework for efficiently exploring complex 3d environments." in *Robotics: Science and Systems*, 2021.

[5] A. Shukla and H. Karki, "Application of robotics in onshore oil and gas industry—a review part i," *Robotics and Autonomous Systems*, vol. 75, pp. 490–507, 2016.

[6] I. Sa and P. Corke, "Vertical infrastructure inspection using a quadcopter and shared autonomy control," in *Field and service robotics*. Springer, 2014, pp. 219–232.

[7] C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter, "Anymal in the field: Solving industrial inspection of an offshore hvdc platform with a quadrupedal robot," in *12th Conference on Field and Service Robotics (FSR 2019)*, 2019.

[8] G. Caprari, A. Breitenmoser, W. Fischer, C. Hürzeler, F. Tâche, R. Siegwart, O. Nguyen, R. Moser, P. Schoeneich, and F. Mondada, "Highly compact robots for inspection of power plants," *Journal of Field Robotics*, vol. 29, no. 1, pp. 47–68, 2012.

[9] B. Chan, H. Guan, J. Jo, and M. Blumenstein, "Towards uav-based bridge inspection systems: A review and an application perspective," *Structural Monitoring and Maintenance*, vol. 2, no. 3, pp. 283–300, 2015.

[10] L. Bartolomei, L. Teixeira, and M. Chli, "Fast multi-uav decentralized exploration of forests," *IEEE Robotics and Automation Letters*, 2023.

[11] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, and K. Alexis, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[12] M. Tranzatto, M. Dharmadhikari, L. Bernreiter, M. Camurri, S. Khattak, F. Mascarich, P. Pfreundschuh, D. Wisth, S. Zimmermann, M. Kulkarni *et al.*, "Team cerberus wins the darpa subterranean challenge: Technical overview and lessons learned," *arXiv preprint arXiv:2207.04914*, 2022.

[13] M. Kulkarni, M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott, R. Siegwart, M. Hutter, and K. Alexis, "Autonomous teamed exploration of subterranean environments using legged and aerial robots," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3306–3313.

[14] N. Hudson, F. Talbot, M. Cox, J. Williams, T. Hines, A. Pitt, B. Wood, D. Frousheger, K. L. Surdo, T. Molnar *et al.*, "Heterogeneous ground and air platforms, homogeneous sensing: Team csiro data61's approach to the darpa subterranean challenge," *arXiv preprint arXiv:2104.09053*, 2021.

[15] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanskỳ, D. Heřt, M. Petrlík, T. Báča, V. Spurnỳ *et al.*, "Darpa subterranean challenge: Multi-robotic exploration of underground environments," in *International Conference on Modelling and Simulation for Autonomous Systems*. Springer, Cham, 2019, pp. 274–290.

[16] S. Scherer, V. Agrawal, G. Best, C. Cao, K. Cujic, R. Darnley, R. DeBortoli, E. Dexheimer, B. Drozd, R. Garg, I. Higgins, J. Keller, D. Kohanbash, L. Nogueira, R. Pradhan, M. Tatum, V. K. Viswanathan, S. Willits, S. Zhao, H. Zhu, D. Abad, T. Angert, G. Armstrong, R. Boirum, A. Dongare, M. Dworman, S. Hu, J. Jaekel, R. Ji, A. Lai, Y. Hsuan Lee, A. Luong, J. Mangelson, J. Maier, J. Picard, K. Pluckter, A. Saba, M. Saroya, E. Scheide, N. Shoemaker-Trejo, J. Spisak, J. Teza, F. Yang, A. Wilson, H. Zhang, H. Choset, M. Kaess, A. Rowe, S. Singh, J. Zhang, G. A. Hollinger, and M. Travers, "Resilient and modular subterranean exploration with a team of roving and flying robots," *Field Robotics*, 2021.

[17] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," 2022.

[18] N. Hughes, Y. Chang, S. Hu, R. Talak, R. Abdulhai, J. Strader, and L. Carlone, "Foundations of spatial perception for robotics: Hierarchical representations and real-time systems," *The International Journal of Robotics Research*, 2024. [Online]. Available: https://doi.org/10.1177/02783649241229725

[19] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, "Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 7515–7525.

[20] L. Schmid, J. Delmerico, J. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena, "Panoptic multi-tsdfs: a flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8018–8024.

[21] C. Wang, J. Cheng, W. Chi, T. Yan, and M. Q.-H. Meng, "Semantic-aware informative path planning for efficient object search using mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 8, pp. 5230–5243, 2021.

[22] S. Papatheodorou, N. Funk, D. Tzoumanikas, C. Choi, B. Xu, and S. Leutenegger, "Finding things in the unknown: Semantic object-centric exploration with an MAV," in *IEEE International Conference on Robotics and Automation*, London, United Kingdom, May 2023.

[23] S. Fredriksson, A. Saradagi, and G. Nikolakopoulos, "Robotic exploration through semantic topometric mapping," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 9404–9410.

[24] M. F. Ginting, D. D. Fan, S.-K. Kim, M. J. Kochenderfer, and A. akbar Agha-mohammadi, "Semantic belief behavior graph: Enabling autonomous robot inspection in unknown environments," 2024. [Online]. Available: https://arxiv.org/abs/2401.17191

[25] P. Roth, J. Nubert, F. Yang, M. Mittal, and M. Hutter, "Viplanner: Visual semantic imperative learning for local navigation," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023.

[26] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1689–1696.

[27] D. Maggio, Y. Chang, N. Hughes, M. Trang, D. Griffith, C. Dougherty, E. Cristofalo, L. Schmid, and L. Carlone, "Clio: Real-time task-driven open-set 3d scene graphs," *arXiv preprint arXiv:2404.13696*, 2024.

[28] H. Bavle, J. L. Sanchez-Lopez, M. Shaheer, J. Civera, and H. Voos, "Situational graphs for robot navigation in structured indoor environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9107–9114, 2022.

[29] M. Dharmadhikari and K. Alexis, "Semantics-aware exploration and inspection path planning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3360–3367.

[30] M. F. Ginting, S.-K. Kim, D. D. Fan, M. Palieri, M. J. Kochenderfer, and A. akbar Agha-mohammadi, "Seek: Semantic reasoning for object goal navigation in real world inspection tasks," in *Robotics: Science and Systems*, 2024.

[31] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[32] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[33] U.-H. Kim, J.-M. Park, T.-j. Song, and J.-H. Kim, "3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents," *IEEE Transactions on Cybernetics*, vol. 50, no. 12, pp. 4921–4933, 2020.

[34] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *Robotics: Science and Systems*, 2020.

[35] P. D. Petris, H. Nguyen, M. Dharmadhikari, M. Kulkarni, N. Khedekar, F. Mascarich, and K. Alexis, "Rmf-owl: A collision-tolerant flying robot for autonomous subterranean exploration," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022, pp. 536–543.

[36] R. Zurbrügg, H. Blum, C. Cadena, R. Siegwart, and L. Schmid, "Embodied active domain adaptation for semantic segmentation via informative path planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8691–8698, 2022.

[37] B. Yin, X. Zhang, Z. Li, L. Liu, M.-M. Cheng, and Q. Hou, "Dformer: Rethinking rgbd representation learning for semantic segmentation," *arXiv preprint arXiv:2309.09668*, 2023.

[38] D. Jia, J. Guo, K. Han, H. Wu, C. Zhang, C. Xu, and X. Chen, "Geminifusion: Efficient pixel-wise multimodal fusion for vision transformer," 2024.

[39] J. Zhang, H. Liu, K. Yang, X. Hu, R. Liu, and R. Stiefelhagen, "Cmx: Cross-modal fusion for rgb-x semantic segmentation with transformers," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[40] O. Alama, A. Bhattacharya, H. He, S. Kim, Y. Qiu, W. Wang, C. Ho, N. Keetha, and S. Scherer, "Rayfronts: Open-set semantic ray frontiers for online scene understanding and exploration," 2025. [Online]. Available: https://arxiv.org/abs/2504.06994

[41] V. K. Viswanathan, M. A. V. Saucedo, S. G. Satpute, C. Kanellakis, and G. Nikolakopoulos, "An actionable hierarchical scene representation enhancing autonomous inspection missions in unknown environments," 2024. [Online]. Available: https://arxiv.org/abs/2412.19582

[42] T. Zaenker, F. Verdoja, and V. Kyrki, "Hypermap mapping framework and its application to autonomous semantic exploration," in *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2020, pp. 133–139.

[43] S. A. Kay, S. Julier, and V. M. Pawar, "Semantically informed next best view planning for autonomous aerial 3d reconstruction," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3125–3130.

[44] C. Simons, A. Samanta, A. K. Roy-Chowdhury, and K. Karydis, "Segue: Semantic guided exploration for mobile robots," 2025. [Online]. Available: https://arxiv.org/abs/2504.03629

[45] R. Ashour, T. Taha, J. M. M. Dias, L. Seneviratne, and N. Almoosa, "Exploration for object mapping guided by environmental semantics using uavs," *Remote Sensing*, vol. 12, no. 5, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/5/891

[46] X. Yu and C.-W. Chen, "Semantic-aware next-best-view for multi-dofs mobile system in search-and-acquisition based visual perception," 2024. [Online]. Available: https://arxiv.org/abs/2404.16507

[47] A. Milas, A. Ivanovic, and T. Petrovic, "Asep: An autonomous semantic exploration planner with object labeling," *IEEE Access*, vol. 11, pp. 107 169–107 183, 2023.

[48] L. Lu, Y. Zhang, P. Zhou, J. Qi, Y. Pan, C. Fu, and J. Pan, "Semantics-aware receding horizon planner for object-centric active mapping," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3838–3845, 2024.

[49] F. Stache, J. Westheider, F. Magistri, C. Stachniss, and M. Popović, "Adaptive path planning for uavs for multi-resolution semantic segmentation," *Robotics and Autonomous Systems*, vol. 159, p. 104288, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889022001774

[50] J. Rückin, F. Magistri, C. Stachniss, and M. Popović, "An informative path planning framework for active learning in uav-based semantic mapping," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4279–4296, 2023.

[51] X. Liu, A. Prabhu, F. Cladera, I. D. Miller, L. Zhou, C. J. Taylor, and V. Kumar, "Active metric-semantic mapping by multiple aerial robots," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 3282–3288.

[52] I. D. Miller, F. Cladera, T. Smith, C. J. Taylor, and V. Kumar, "Air-ground collaboration with spomp: Semantic panoramic online mapping and planning," *IEEE Transactions on Field Robotics*, pp. 1–1, 2024.

[53] F. Cladera, I. D. Miller, Z. Ravichandran, V. Murali, J. Hughes, M. A. Hsieh, C. J. Taylor, and V. Kumar, "Challenges and opportunities for large-scale exploration with air-ground teams using semantics," 2024. [Online]. Available: https://arxiv.org/abs/2405.07169

[54] L. Bartolomei, L. Teixeira, and M. Chli, "Semantic-aware active perception for uavs using deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3101–3108.

[55] D. Hu and V. J. Gan, "Semantic navigation for automated robotic inspection and indoor environment quality monitoring," *Automation in Construction*, vol. 170, p. 105949, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092658052400685X

[56] D. Chen, M. Zhuang, X. Zhong, W. Wu, and Q. Liu, "Rspmp: real-time semantic perception and motion planning for autonomous navigation of unmanned ground vehicle in off-road environments." *Applied Intelligence*, 2023.

[57] J. Sun, J. Wu, Z. Ji, and Y.-K. Lai, "Rsmpnet: Relationship guided semantic map prediction," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 302–311.

[58] H. Du, X. Yu, and L. Zheng, "Learning object relation graph and tentative policy for visual navigation," in *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020, pp. 19–34.

[59] Y. Liang, B. Chen, and S. Song, "Sscnav: Confidence-aware semantic scene completion for visual semantic navigation," in *2021 IEEE International Conference in Robotics and Automation (ICRA)*, 2021.

[60] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," in *Proceedings of (ICLR) International Conference on Learning Representations*, May 2019.

[61] Y. Qiu, A. Pal, and H. I. Christensen, "Learning hierarchical relationships for object-goal navigation," in *2020 Conference on Robot Learning (CoRL)*, 2020.

[62] Z. Ravichandran, V. Murali, M. Tzes, G. J. Pappas, and V. Kumar, "Spine: Online semantic planning for missions with incomplete natural language specifications in unstructured environments," 2025. [Online]. Available: https://arxiv.org/abs/2410.03035

[63] R. Shrestha, F.-P. Tian, W. Feng, P. Tan, and R. Vaughan, "Learned map prediction for enhanced mobile robot exploration," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1197–1204.

[64] Y. Tao, Y. Wu, B. Li, F. Cladera, A. Zhou, D. Thakur, and V. Kumar, "Seer: Safe efficient exploration for aerial robots using learning to predict information gain," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1235–1241.

[65] D. P. Ström, F. Nenci, and C. Stachniss, "Predictive exploration considering previously mapped environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2761–2766.

[66] L. Roldão, R. de Charette, and A. Verroust-Blondet, "3d semantic scene completion: A survey," *International Journal of Computer Vision*, 2022.

[67] L. Holder, D. Cook, J. Gonzalez, and I. Jonyer, *Structural Pattern Recognition in Graphs*. Boston, MA: Springer US, 2002, pp. 255–279. [Online]. Available: https://doi.org/10.1007/978-1-4613-0231-5_10

[68] M. Dharmadhikari, P. De Petris, H. Nguyen, M. Kulkarni, N. Khedekar, and K. Alexis, "Manhole detection and traversal for exploration of ballast water tanks using micro aerial vehicles," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2023, pp. 103–109.

[69] K. Helsgaun, "An effective implementation of the lin–kernighan traveling salesman heuristic," *European journal of operational research*, vol. 126, no. 1, pp. 106–130, 2000.

[70] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," 2010.

[71] M. Dharmadhikari, P. De Petris, M. Kulkarni, N. Khedekar, H. Nguyen, A. E. Stene, E. Sjøvold, K. Solheim, B. Gussiaas, and

K. Alexis, "Autonomous exploration and general visual inspection of ship ballast water tanks using aerial robots," in *2023 21st International Conference on Advanced Robotics (ICAR)*, 2023, pp. 409–416.

[72] B. Zhou, Y. Zhang, X. Chen, and S. Shen, "Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 779–786, 2021.

[73] S. Khattak, H. Nguyen, F. Mascarich, T. Dang, and K. Alexis, "Complementary multi–modal sensor fusion for resilient robot pose estimation in subterranean environments," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 1024–1029.

[74] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using ros," *Springer Book on Robot Operating System (ROS)*.