Intelligent Exploration and Autonomous Navigation in Confined Spaces

Aliakbar Akbari, [1] Puneet S Chhabra,[2] Ujjar Bhandari [3] and Sara Bernardini [4]

***Abstract*— Autonomous navigation and exploration in confined spaces are currently setting new challenges for robots. The presence of narrow passages, flammable atmosphere, dust, smoke, and other hazards makes the mapping and navigation tasks extremely difficult. To tackle these challenges, robots need to make intelligent decisions, maximising information gain while maintaining the safety of the system and their surroundings. In this paper, we present a suite of reasoning mechanisms along with a software architecture for exploration tasks that can be used to underpin the behavior of a broad range of robots operating in confined spaces. We present an autonomous navigation module that allows the robot to safely traverse known areas of the environment and extract features of the unknown frontier regions. An exploration component, by reasoning about these frontiers, provides the robot with the ability to venture into new spaces. From low-level sensory input and contextual information, the robot incrementally builds a semantic network that represents known and unknown parts of the environment and then uses a logic-based, high-level reasoner to interrogate such a network and decide the best course of actions. We evaluate our approach against several mine-like challenging scenarios with different characteristics using a small drone. The experimental results indicate that our method allows the robot to make informed decisions on how to best explore the environment while preserving safety.**

I. Introduction

Confined spaces are characterized by two features: i) They are substantially (or fully) enclosed, with limited access of entry and egress; and ii) They present one of more risks for operators specified among fire and explosions, presence of gases, vapours, fumes, dust, and flowing solid, increase in body temperature and drowning. Examples are chambers, silos, tanks, mines, pits, pipes, sewer and similar spaces. The use of robotic systems to assist or remove human operators working in such restricted and hazardous environments can prove highly beneficial to minimize health & safety risks and make operations in such spaces more time and cost effective.

Several tasks that need to be performed in confined spaces require an initial exploration and mapping of them. Given the possible extreme conditions that can be encountered, robot navigation and exploration become challenging problems in such spaces. In maximising the acquired knowledge, robots need to take into account the conditions of the environment, e.g., presence of multiple and narrow junctions, toxic and flammable gases, and poor visibility (e.g., fog or dust) as well as their available resources to be able to safely leave the space when necessary. To tackle such issues, higher levels of autonomy along with advanced sensing are needed.

Fig. 1: (Left) A real-world example of confined space, an actual mine. Source: Network Rail, UK. (Right, Top) - A simulated environment with multiple entry and exit points. (Right, Bottom) - Inside view of the simulated scene used for experiments.

Intelligent exploration demands robots to sense, localize, map and navigate in a 3D space by making use of semantic and contextual knowledge. The 3D map of the environment can be constructed using sensors and robot localization software, e.g., simultaneous localization and mapping (SLAM) system. A SLAM system is capable of providing the information on free and occupied spaces or grids around the robot. *Autonomous navigation* finds collision-free paths by employing motion planning algorithms that take into account the 3D map and localization information. Finally, *semantic knowledge processing* deals with representing knowledge (e.g., connected regions and their state) and reasoning over it to enable the robots to take smart decisions while exploring.

To deal with the challenges posed by using robotic systems in confined spaces, this paper presents a semantic and intelligent exploration approach which aims to provide autonomous navigation and smart decision making for robots tasked with surveying unknown regions and confined spaces. The main contributions of this work are:

- *Advanced sensing*. We used a light-weight ($< 60gms$) 360° time-of-flight (ToF) imaging solution that produces infra-red (IR) and 3D point cloud data in a *single snap-shot*. We apply and test our algorithms on sparse point cloud data collected using the *Dragonfly* system.
- *Autonomous navigation*. Robot navigation in unknown environments involves continuous mapping, localisation and path planning. We propose a reasoning module that *selects goals intelligently*.
- *Semantic knowledge processing*. We use semantic net-

*This work is partially supported by the Innovate UK.

Aliakbar Akbari[1] and Sara Bernardini[4] are with Department of Computer Science, Royal Holloway University of London, Egham, UK. ali.akbari@rhul.ac.uk, sara.bernardini@rhul.ac.uk

Puneet S Chhabra[2] and Ujjar Bhandari[3] are with Headlight AI Ltd., London, UK. puneet@headlight.ai, ujjar@headlight.ai

works to represent the robot's knowledge. The network is incrementally built using the low-level information collected when a robot explores an area. A set of *logic-based reasoning process*, along with environmental knowledge, are used to explore the surroundings. Upon mission termination, this knowledge is then provided to the operator.

- *High-level reasoner*. We propose a high-level reasoner that aims to make the best decisions for the robot. This module employs semantic knowledge processing using the latest environment conditions (measured using sensors) and the robot's contextual information, e.g., regions with fog or smoke. This kind of intelligent awareness enables the robots to explore confined spaces even in the presence of multiple, complex branches.

II. Related Works

Robot exploration has been studied in the context of various applications. Frontier exploration is one of the most popular approaches [1]. A frontier region is the boundary between explored and unexplored region. The frontier exploration algorithm drives the robot to visit the areas that have not been explored. Selin *et al.* [2] concentrate on exploring large areas, while Ravankar *et al.* [3] focus on exploration using low-cost sensors. The approach in Bachrach *et al.* [4] also employs a modified version of the frontier exploration method to choose possible poses in the free space where a robot should y to to observe previously unexplored regions and remain well-localized.

The work Vanegas *et al.* [5] present navigation in unknown and GPS-denied environments. Mainly, the approach combines the use of Simultaneous Localization and Mapping (SLAM) with Partially Observable Markov Decision Processes (POMDP) algorithms into a framework in which the navigation and exploration tasks are modeled as sequential decision problems under uncertainty. Shim et al [6] offer an autonomous exploration algorithm for urban navigation by building local obstacle maps and looking for conflict-free trajectories by using a model predictive control (MPC) framework. There are some other approaches, e.g., [7], [8], [9], that enhance exploration using different versions of perception and localization systems with path planning in confined spaces. In this direction, the work Thrun *et al.* [10] discuss the software architecture of an autonomous robotic system designed to explore and map abandoned mines. A set of software tools is presented, enabling robots to acquire maps of unprecedented size and accuracy.

Most approaches focus on exploring unknown regions with no high-level smart decision making capabilities or consideration of the robot's resources. However, there are some challenging applications in confined-space environments where a robot needs to be smart enough to take the effective decisions in the presence of multiple options, motion difficulties, fog, dust, and similar factors. This paper addresses these challenges by proposing an integrated approach to embrace autonomous navigation, semantic knowledge, and logic-based reasoning augmented with a context and vehicle health system that results in intelligent exploration.

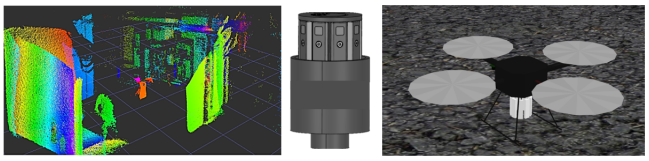

Fig. 2: (Center) A light-weight 360° time-of-flight (ToF) 3D camera system used to capture a confined space (a basement). (Left) An example point cloud generated using the *Dragonfly* system. (Right) A simulated robot in Gazebo carrying the 3D ToF camera system.

III. 3D Sensing and Localization

To create a 3D map and perform localization, a robot should be able to carry out SLAM while it is navigating. The 3D camera system *Dragonfly* (shown in Figure 2) that we use can generate point cloud data at 45fps, producing 15Mil points. However, in our experiments, we down-sample and capture data at 5fps, producing 2Mil points. Figure 2 illustrates how the *Dragonfly* system is integrated with a drone and an example point cloud data captured using the system.

The *Dragonfly* SDK allows the robot to localise itself in the world coordinates, in the absence of GPS. This is further used to produce a fused 360deg point cloud. The *Dragonfly* SDK also provides a 3D map along with the borders. The nodes that belong to the border regions are wrapped into *frontier regions*. Finally, the SDK identifies voxel regions, both free and occupied, along with their boundaries (see Figure 3).

We developed a reasoning mechanism based on the perception system to identify region boundaries in confined spaces. The boundaries could be barriers and entries that branch out a space to a new environment. Regions are classified into branches and voids. A ray casting technique is used to identify whether a robot is located within the barriers or not according to the sensors range. If a robot is surrounded by obstacles, the region is labelled as a *branch*; when it is obstacle free, the region is labelled as *void*. In addition, our perception system allows us to measure the closest distance between a frontier node and obstacles.

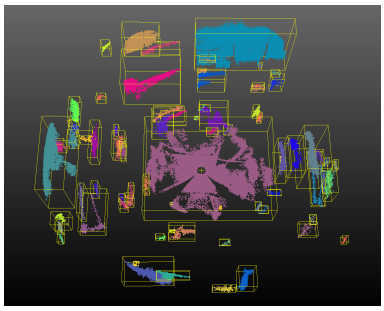

Fig. 3: An example point cloud (captured using the *Dragonfly* system), segmented into regions (various colors) and the region boundaries highlighted in yellow.

IV. Semantic Knowledge Processing

We now show how the robot makes use of high-level knowledge to make intelligent decisions. Our approach consists of two main steps: i) knowledge representation; and ii) reasoning over that knowledge. Knowledge can be structured through a semantic network providing relations between different *concepts*. It is composed of *nodes*, which are either concepts or objects, and *links* which express the relations between them. The semantic network is incrementally constructed and expanded using low-level information. After a robot explores an area, sensor measurements and contextual information is extracted and semantically laid over the network. The main templates proposed for the nodes are:

- **Agent**: robots acting in the environment.
- **Region**: regions of explored and unexplored space.
- **Vision**: values of sensory information, e.g., clear, fog, dust.
- **Gas**: different gases that are in the environment.

In addition, the predicates corresponding to the links are:

- **Loc**: $loc\,(rob, rgn)$ links the current location of a robot rob with a region rgn.
- **Conn**: $conn\,(rgnX, rgnY)$ represents that region $rgnX$ is connected to another region $rgnY$ inside the environment.
- **Exp**: $exp\,(rgn, value)$ says that region rgn has exploration value $value$.
- **Gas**: $gas\,(rgn, value)$ shows that region rgn has a flammable gas value $value$, which could be either flammable or not.
- **Vis**: $vis\,(rgn, vision)$ links region rgn with the vision value $vision$, which can be dust, smoke, dirt, and clear.
- **Visit**: $visit\,(rgnX, rgnY)$ says that region $rgnX$ is the visited region for the region $rgnY$.
- **Pot**: $pot\,(rgnX, rgnY)$ says that region $rgnX$ is one of the candidate regions to be explored after region $rgnY$.
- **Best**: $best\,(rgnX, rgnY)$ says that region $rgnX$ is the best next region to explore after region $rgnY$.
- **BestNeigh**: $bestNeigh\,(rgnX, rgnY)$ says that region $rgnX$ is the best neighbouring region for region $rgnY$.
- **BestHome**: $bestHome\,(rgnX, rgnY)$ says that region $rgnX$ is the best region towards the robot home position for the region $rgnY$.

We propose a set of logic-based reasoning processes to extract semantic information from the knowledge base. This information is then fed into the exploration module. The reasoning process uses the syntax of first-order logic. This includes quantifiers, predicates, functions, variables, as well as constants. We show some of our formulas in what follows. To find the next feasible region to explore, the following reasoning is applied:

$$\begin{aligned}\varphi_1 : \exists x\, \exists y\, \{&conn(x,y) \wedge exp(x, notExplored) \\ &\wedge vis(x, clear) \wedge gas(x, notFlammable) \\ &\wedge fSize(x) > fDim(rob)) \implies pot(x,y)\}\end{aligned} \tag{1}$$

The functions $fSize()$ and $fDim()$ in (1) return the size of an entrance and the dimension of a robot, respectively. Formula (1) says that, if region y is connected with region x, and x is unexplored, has clear vision, is not flammable, and the size is greater than the dimension of a robot, x a potential candidate region that could be explored next.

After finding all the candidate regions to explore next, the robot decides which one is the best option using the following reasoning process:

$$\begin{aligned}\varphi_2 : \exists x_n \exists x' \exists y \{&pot(x_n, y) \\ &\wedge x' = fMinDist(x_n, rob) \implies best(x', y)\}\end{aligned} \tag{2}$$

The function $fMinDist$ in (2) gets the list of all candidate regions x_n and the current robot location, rob and returns the region x' which has the minimum distance to rob. The region x' becomes the best next region to explore.

If all the regions around the robot are visited, the robot should determine which one of the visited regions is the best one to traverse to perform further exploration. The following formula defines how the robot can do that:

$$\begin{aligned}\varphi_3 : \exists x_n\, \exists x'_n\, \exists y\, \exists z_n \{&visit\,(x_n, y) \wedge pot\,(x'_n, x_n) \\ &\wedge z_n = fMinNeigh\,(x'_n, rob) \\ &\implies bestNeigh(z_n, y)\}\end{aligned} \tag{3}$$

In (3), the predicate $visit$ lists all the explored regions x_n adjacent to a region y using the predicate exp. The function $fMinNeigh$ gets the potential regions x'_n and the current robot position and returns the list of regions z_n connecting the current location to the next best unexplored area through neighbouring regions with minimal cost. The best neighbouring regions for an area are returned by the predicate $bestNeigh$.

The following formula reasons about how a robot can select the shortest sequence of regions in order to get back to the home position from the current location:

$$\begin{aligned}\varphi_4 : \exists x \exists y_n \{&loc\,(rob, x) \wedge\ (y_n = fMinCost\,(x, rob)) \\ &\implies bestHome\,(y_n, x)\}\end{aligned} \tag{4}$$

In (4), the function $fMinCost$ lists all the regions that connect the robot's location to the home position. The formula stores in $bestHome$ the list of the regions with the minimum cost to home.

V. Autonomous Navigation

Planning collision free paths is integral to autonomous navigation. *Motion Planning* deals with detecting collision-free paths and generating a set of way-points. It is mostly done in the configuration space [11]. The configuration space has as many dimensions as the degrees of freedom a robot has, and therefore each point represents a configuration (state) of the robot. The subspace corresponding to collision-free configurations is called the *free space* and the subspace corresponding to collision configurations is called the *collision space*. Motion planning consists in finding a path in the *free space* between two robot configurations.

Recent research is centered around sampling-based motion planning to provide efficient solutions to path planning

by avoiding the need to compute the whole configuration space. In this work, we employ the *RRT-Connect* motion planner [12]. Two trees are rooted at the *start* and *goal* configurations, requiring to meet. One tree is extended and attempts to connect to the closest node of the other tree in each iteration. Next, the procedure is reversed by swapping both trees. A greedy heuristic method is applied to traverse the two trees.

In an unknown environment, with limited or incomplete information about the free and collision space, a robot should be integrated with re-planning capability in order to traverse the space safely. Via navigation, more information can be perceived and knowledge added through the robot vision system. This allows the robot to apply collision checking for the rest of configurations found in the path. Once any collision is detected, re-planning is applied to look for a new path from the current configuration. Algorithm 1 details the steps of the proposed autonomous navigation module. First, the type of region is returned using EvalRgn. It can be an entry region, a branch or a void. Frontier nodes are then laid on it [line 2]. The goal configuration is chosen by SetGoal, taking into account the region type [line 3]. If the type is void, the function selects the frontier node nearest to the obstacles. A safety distance from them is maintained to stabilize the robot within the barriers (obstacles or walls). If the type is a branch (multiple entry/exit points), we select the best frontier node (from a list of all the nodes as far away as possible from the current robot location and the obstacles). Finally, in case of entry/exit region, the mean value (in meters) of the entry/exit is computed and returned. The function SetMPQuery is in charge of setting and returning all motion planning parameters containing *planning time*, *goal state*, *initial state* (using the SDK), *step size*, etc [line 4].

Motion planning is then called by CallMP and a collision-free path is reported [line 5]. If a path is found, the status of the robot power is determined by CheckPower to check that there is sufficient power to follow the path [line 9]. Each configuration of the path is evaluated against collisions with the environment by CheckCollision. If it is collision free, a robot moves to the configuration using MoveToConf and the map $\mathcal{M}$ is updated from the current position using the SDK; otherwise, the function Navigate is again called and re-planning takes place from the current state [line 15].

Algorithm 1: Navigate($\mathcal{M}$, Rgn, $\mathcal{G}$)

```
1  Res ← ∅
2  T ← EvalRgn(Rgn)
3  G ← SetGoal(Rgn, T)
4  P ← SetMPQuery(G)
5  Q ← CallMP(M, P)
6  if Q = ∅ then
7      Res = PathNotFound
8      return {Res, M}
9  if CheckPower(G, Q) then
10     foreach { q ∈ Q } do
11         if !CheckCollision(q, M) then
12             MoveToConf(q)
13             M ← ExpandMap()
14         else
15             return Navigate(M, Rgn, G)
16     Res = Executed
17 else
18     Res = BatteryLimit
19 return {Res, M}
```

VI. Intelligent Exploration

The robotics exploration problem deals with maximizing knowledge of an unknown environment and is highly challenging when applied to confined spaces. We have developed an intelligent exploration planner which creates and exploits semantic knowledge of the unknown regions. It employs the proposed autonomous navigation technique enabling a robot to smartly define a goal configuration and avoid obstacles while exploring.

Algorithm 2 depicts the proposed exploration function that is mainly responsible to explore an unknown region and propagate knowledge to the semantic network. It gets the current occupancy map $\mathcal{M}$ and the semantic map knowledge $\mathcal{G}$ and returns the status of exploration, the updated knowledge (semantic graph) and the map. The algorithm looks for frontier regions by FindFrontier [line 4]. To make this process efficient, a bounding box is considered around the current location of the robot and frontier nodes are then computed within the box. The new nodes are appended to the list of frontiers computed at the previous stage. Using this information, a Navigate function is called, which reports the new map as well as the status of the navigation module [line 5]. If navigation is *executed* and a new entry region is identified, such a region will become part of the high-level knowledge. If there is no path, a new frontier region is again computed excluding the infeasible frontier nodes. Navigate can fail due to the limited robot power, in which case the Explore returns the corresponding values, i.e., the current state of exploration. The RegionExplored function evaluates whether the current region is completely explored or not [line 15]. If there is no frontier regions left in the space, the function returns true, and eventually the map is updated and set to explored using the predicate exp by the function UpdateGraph [line 16].

Example in Figure 4 illustrates how a semantic graph is created. After exploration, region *Rgn1* is marked *explored* using the function UpdateGraph. While the region is being scanned, the robot identifies two more regions, *Rgn2* and *Rgn3*, connected to the current one. The corresponding vision values are also added to the graph. All the new nodes and edges are added by the functions AddNode and AddEdge, respectively.

Algorithm 2: Explore($\mathcal{M}, \mathcal{G}$)

```
1  RgnExplored ← False
2  Res ← ∅
3  while !RgnExplored do
4      F ← FindFrontier(M)
5      {ResN, M'} = Navigate(M, F , G)
6      if ResN = Executed then
7          if NewEntry(M', S) then
8              G ← AddNode(M', S)
9              G ← AddEdge(M', S)
10     else if ResN = PathNotFound then
11         Continue
12     else
13         return {ResN, M', G}
14     M ← M'
15     if RegionExplored(M) then
16         G ← UpdateGraph(M))
17         Res = Explored
18         RgnExplored ← True
19 return {Res, M, G}
```

VII. High-Level Reasoner

We use a high-level reasoner module to guide the robot exploration mission. It takes the best decisions using the proposed reasoning processes as described in Algorithm 3. In particular, it gets all the point clouds PC published by the sensors and returns the computed 3D occupancy map $\mathcal{M}$ along with a semantic graph $\mathcal{G}$. All the point clouds are fused together by the function FusePC [line 1] and the initial 3D Map is constructed using GetInitMap [line 2] using the SDK. The semantic graph is then initialized by the function InitGraph [line 3]. A robot implements the mission exploring all the feasible areas [lines 5 - 19]. At each iteration, the function Explore [line 6] is called to compute unexplored areas.

Next, the robot tries to identify the best region(s) to explore using the functions NextBestRgn [lines 8], defined as follows:

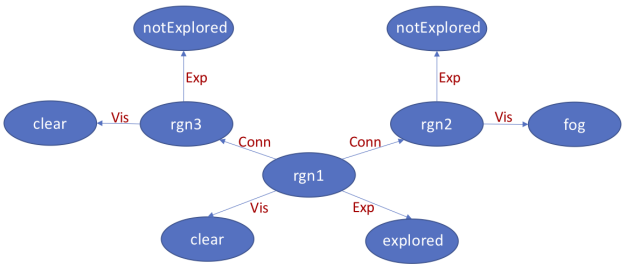

Fig. 4: An example to represent some parts of the semantic graph.

$$Result(NextBestRgn, \langle \mathcal{G} \rangle) = \begin{cases} BR & iff\ \varphi_2\ holds; \\ NR & iff\ \varphi_3\ holds; \\ \emptyset & otherwise; \end{cases}$$

To find the next best region(s), the formula φ_2 is first called to identify the best neighbouring unexplored region. If it does not hold, the formula φ_3 is called to determine the next best regions through all the neighbouring visited regions. If both φ_2 and φ_3 do not hold, the function returns null. In the latter case, the robot tries to find all the regions that are at an intermediate distance between the current one and the initial one using the function BestRgnHome [line 10]. The function GoHome [line 11] uses the formula φ_4 in order to get back to the home position. If instead the next region to explore is identified, the function Navigate is called [line 14] and the robot goes to that region.

If exploration does not terminate due to battery limits or other failures, the robot tries to identify the best regions to traverse to get back to the initial region and then moves through them [lines 20-22].

Algorithm 3: High-level reasoner

```
   inputs : PC
   outputs: M, G
1  FPC ← FusePC(PC)
2  M ← GetInitMap(FPC)
3  G ← InitGraph(M)
4  Mission ← False
5  while !Mission do
6      {ResE, M', G} = Explore(M, G)
7      if ResE = Explored then
8          N ← NextBestRgn(G)
9          if N = ∅ then
10             N' ← BestRgnHome(G)
11             GoHome(N', M)
12             Mission ← True
13         else
14             {ResN, M'} = Navigate(M, N, G)
15             M ← M'
16             if ResN = BatteryLimit then
17                 break
18     else
19         break
20 if !Mission then
21     N' ← BestRgnHome(G)
22     GoHome(N', M)
23 return {M, G};
```

VIII. Implementation and Results

We now describe the main tools that we employed to implement the proposed framework. For motion planning, we use the MoveIt software[1], one of the most widely utilized software for path planning in robotics. It is well integrated with ROS (Robot Operating System) [2] and includes the

[1]http://moveit.ros.org
[2]https://www.ros.org/

Problem	Geometric and Execution Time (s)			Region Information			Explored Volume (m^3)			Exploration Time (s)
	Frontier	MP	Execution	Explored	Infeasible	Discovered	Free	Occupied	Total	
Problem-1	46	31	450	5	2	7	1273	147	1420	571
Problem-2	37	20	368	4	2	7	1219	126	1345	464
Problem-3	17	15	269	2	1	4	1162	130	1292	333
Problem-4	7	8	190	2	0	3	866	92	958	231
Problem-5	24	20	299	6	2	8	559	76	635	379

TABLE I: The system performance of different problems in terms of geometric reasoning and execution time, regions information, and exploration information. MP stands for motion planning.

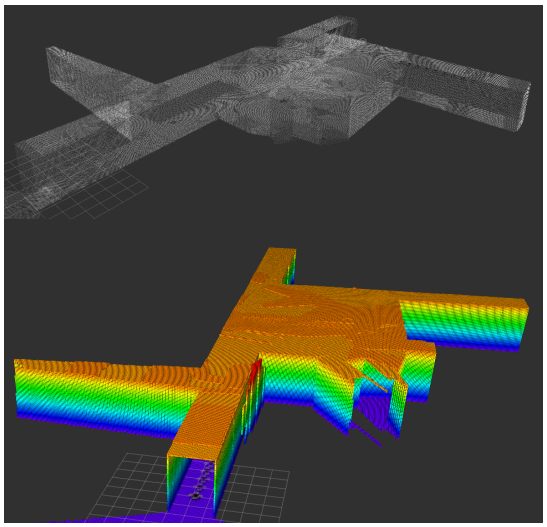

Fig. 5: Accessible explored areas. Top and bottom pictures show the fused point cloud and the occupancy 3D Map respectively.

Open Motion Planning Library (OMPL), an open source path planning library that implements state-of-the-art path planning algorithms. We adopt MoveIt for our robotic system for point cloud fusion, geometric reasoning (to define a goal) and the re-planning capability.

As for the robot model and localization, we use the Hector Quadrotor tools, which are also available online[3]. We employ the *Dragonfly* system (shown in Figure 2), with a 360deg field-of-view (FoV), to sense and gather information around the robot. We integrated the Octomap library [13] into our system for the construction of the 3D Map.

We implemented the semantic knowledge template and the reasoning processes by using the *Prolog* language, which provides rule-based logical queries, with the SWI-Prolog library[4]. We defined an interface for communication between the geometric reasoner and the high-level information.

The proposed approach has been tested against several scenarios with different characteristics. Figure 5 represents the fused point cloud as well as the 3D Map of the feasible areas of the environment in Figure 1. We consider problems with different exploration times and complexities in our experimentation: *Problem-1*: unlimited time; *Problem-2*: 470 (s); *Problem-3*: 340 (s); *Problem-4*: 240 (s); and *Problem-5*: exploration of areas with multiple loops and infeasible areas with unlimited time (see Figure 6). The performance of the

[3]https://github.com/tahsinkose/hector-moveit

[4]https://www.swi-prolog.org/

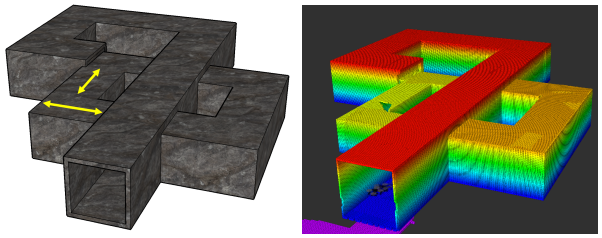

Fig. 6: The result of the explored areas. Left and right pictures show the simulated world (yellow arrows show the infeasible regions) and the occupancy 3D Map respectively.

robot for each of the above problems has been reported in Table I. We detail geometric reasoning and execution time, region information, explored volume, and exploration time.

As we expected, the solutions to *Problems 2* to *4* indicate that, under battery constraints, the robot explores smaller spaces. The table (*Problem-1* and *Problem-5*) also shows that if the robot has unlimited exploration time, it is able to explore all the feasible areas. In *Problem-5*, the robot needs to identify different loops, which are captured by the reasoning processes, to avoid to explore the visited region.

Figure 7 compares the performance of our approach with a simplified one for the scenarios represented in Figure 1 (Example 1) and Figure 6 (Example 2). *App1* indicates our full approach, while *App2* represents the same approach, but we deactivate our smart reasoning process for selecting frontier nodes to show the efficiency of the proposed reasoner. In particular, in *App2*, we disable the following components:

- The functions EvalRgn and SetGoal in Algorithm 1.
- The function NextBestRgn using the formulas φ_2 and φ_3 in Algorithm 3.

App2 also computes frontier nodes throughout the occupancy 3D Map. In consequence, it is only able to identify infeasible regions and navigate randomly to any frontier nodes using the motion planner. Geometric reasoning and execution time in *App2* increase. The main reason is that more calls to the motion planner are required to complete the mission. Figure 7 shows that the robot requires to spend significant amount of time to explore the whole environment with *App2*, whereas, in *App1*, the robot is able to perform the exploration task efficiently.

Figure 8 represents the total volume of the explored areas for *Problems 1* to *5* and the two different approaches. The problems consider battery constraints by limiting the exploration time. All the cases demonstrate that *App1* is able to cover more space compared to *App2*.

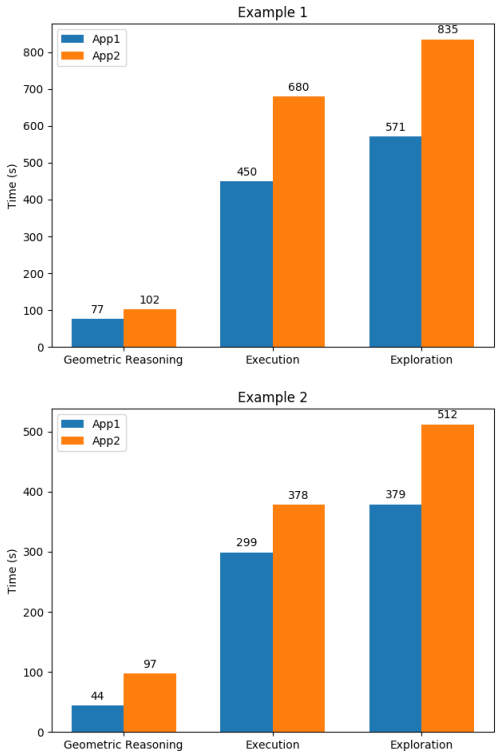

Fig. 7: Performance comparison of approaches *App1* and *App2*, which shows the significance of the proposed reasoning processes in terms of time efficiency. *App1* is the proposed approach and *App2* is the same one with no smart reasoning processes over frontier nodes.

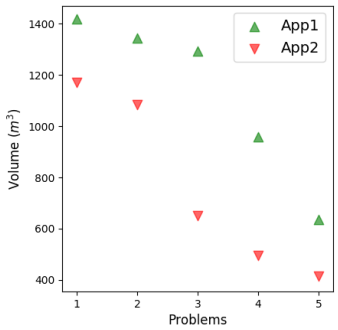

Fig. 8: The comparison of the approaches in terms of covering the space of explored areas.

The proposed approach is capable to recognise and avoid infeasible regions as well as to explore the confined spaces smartly using the embedded reasoning processes. Once the robot has explored a region, it decides to go to explore the next best region near to the current location. This makes the exploration process efficient as the robot avoids to navigate on the visited regions as much as possible. For instance, Figure 9 shows the robot is currently located at Region 6 (*Rgn6*) after exploring Regions 1 and 3 and discovering other regions. The key question is which region is the best to explore next. When making this decision, the robot knows that the adjacent region (*Rgn3*) has been explored and looks

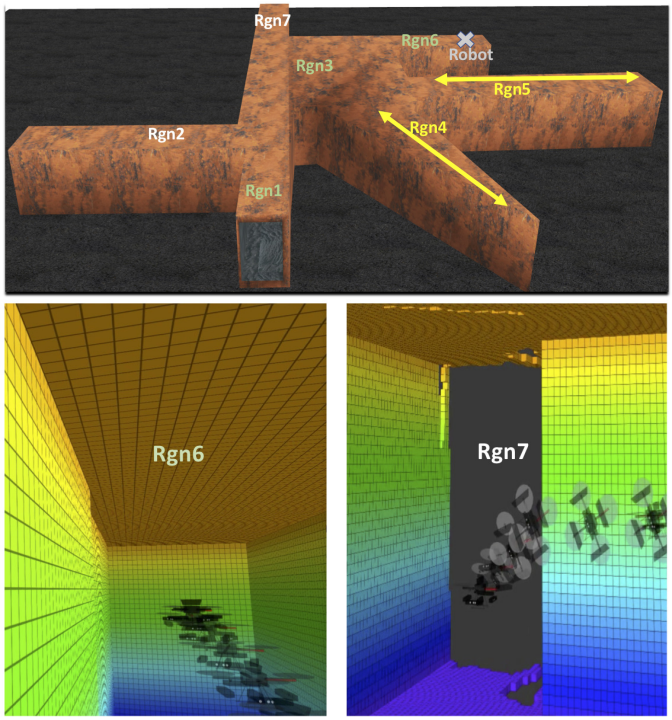

Fig. 9: An example of the robot's intelligent decision making to find and explore the next best region. Top picture shows the current location of the robot. Yellow arrows specify the infeasible regions. Regions in green color are the explored ones. Regions in white color are unexplored ones. Bottom picture shows the perception and navigation of the robot from *Rgn6* to *Rgn7*.

for another candidate in the semantic graph. It finds that *Rgn5* is the next nearest region, but it is an infeasible region. In consequence, it chooses *Rgn7* for exploration. This process makes the robot capable of exploring the environments efficiently, while avoiding hazardous areas. Figure 10 also shows our results with real sensory data and autonomous navigation planning where the robot explores a corridor in a basement.

IX. Conclusion

We present an intelligent exploration approach for applications in confined spaces. The proposed approach consists of *advanced sensing*, *autonomous navigation*, *semantic knowledge processing*, and *high-level reasoning*. Our approach enables the robot to maximize the information acquired

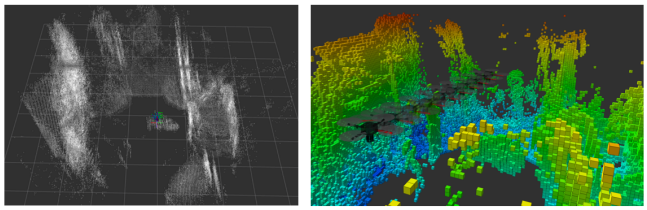

Fig. 10: We use real-world scans captured using the *Dragonfly* system. This data was then imported in Gazebo and used for drone navigation (shown in Figure 2). (Left) Original point cloud. (Right) Voxelized visualisation of obstacles and drone path.

during exploration while, at the same time, considering environmental factors that can affect its fitness and resources available. We evaluate our approach against several mine-like applications with various characteristics and present our results in terms of system performance for the mapped environments. We, moreover, compared the proposed approach, against the basic version of our planning to show the significance and efficiency of the presented approach in terms of making intelligent decisions as well as saving exploration time. Future work will focus on defining probability distribution over the semantic network and considering knowledge-based reasoning under uncertainty.

REFERENCES

[1] B. Yamauchi, "A frontier-based approach for autonomous exploration." in *cira*, vol. 97, 1997, p. 146.

[2] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, "Efficient autonomous exploration planning of large-scale 3-d environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1699–1706, 2019.

[3] A. A. Ravankar, A. Ravankar, Y. Kobayashi, and T. Emaru, "Autonomous mapping and exploration with unmanned aerial vehicles using low cost sensors," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 4, no. 1, 2018, p. 44.

[4] A. Bachrach, S. Prentice, R. He, and N. Roy, "Range–robust autonomous navigation in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 5, pp. 644–666, 2011.

[5] F. Vanegas, K. J. Gaston, J. Roberts, and F. Gonzalez, "A framework for uav navigation and exploration in gps-denied environments," in *2019 IEEE Aerospace Conference*. IEEE, 2019, pp. 1–6.

[6] D. Shim, H. Chung, H. J. Kim, and S. Sastry, "Autonomous exploration in unknown urban environments for unmanned aerial vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005, p. 6478.

[7] C. Papachristos, S. Khattak, and K. Alexis, "Autonomous exploration of visually-degraded environments using aerial robots," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 775–780.

[8] F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "A multi-modal mapping unit for autonomous exploration and mapping of underground tunnels," in *2018 IEEE aerospace conference*. IEEE, 2018, pp. 1–7.

[9] C. Papachristos, S. Khattak, F. Mascarich, and K. Alexis, "Autonomous navigation and mapping in underground mines using aerial robots," in *2019 IEEE Aerospace Conference*. IEEE, 2019, pp. 1–8.

[10] S. Thrun, S. Thayer, W. Whittaker, C. Baker, W. Burgard, D. Ferguson, D. Hahnel, D. Montemerlo, A. Morris, Z. Omohundro, *et al.*, "Autonomous exploration and mapping of abandoned mines," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 79–91, 2004.

[11] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE transactions on computers*, vol. C-32, pp. 108–120, 1983.

[12] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[13] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com. [Online]. Available: http://octomap.github.com