TIME CONSTRAINED EXPLORATION USING TOPOSEMANTIC SPATIAL MODELS: A REPRODUCIBLE APPROACH

Jose Luis Susa Rincon, Stefano Carpin
Department of Computer Science and Engineering
University of California, Merced

As robots become ubiquitous in our everyday lives, they are more and more often assigned complex tasks involving multiple objectives at once. Moreover, efficiency, here intended as the ability to complete more tasks in a given amount of time, is becoming increasingly important. At the same time, spurred by progress in machine learning, there is a tendency to explore novel designs in which robots rely more on visual sensors and less on traditional sensors like range finders. Starting from these premises, the objective of this paper is twofold. First, we revisit the classic exploration problem introducing temporal constraints in the task and embracing a toposemantic spatial representation that does not include any metric attribute. To assess strengths and weaknesses of the various exploration methods abstracting from the underlying technical implementation, we perform a set of massive simulations in ROS using its Gazebo simulation environment. The simulation-based approach leads to the second objective of this contribution, namely, presenting a set of findings that are reproducible by a third party. As *measurable robotics* gains more attention, this work represents a first attempt to present a reproducible study based on Gazebo.

1. MOTIVATION

Consider a service robot that must deliver an item in an unknown environment, say an office floor it never entered before, and for which it has no prior knowledge other than the fact that it is an office environment. For example, the robot may have to deliver an envelope in the copy room, or put a package on the desk in Mr. Chairman's office. Not having preliminary knowledge about where these places are located, the robot should then explore the environment until it recognizes it has reached the desired place. Then, it shall deposit the goods it is supposed to deliver, travel its way back to the entrance, and leave. Exploration is an integral component of this task, and if the robot is doing this job as part of a series of deliveries, it is important to complete the task quickly to increase the number of tasks completed in a set amount of time.

Exploration is one of the fundamental abilities for an autonomous robot operating in an unknown environment. In its most studied form, this means building a spatial model of the environment, and central to the task is the decision process determining "where to move next.'' A classic early solution to this problem is *frontier-based exploration* [1], an approach inherently tied to using occupancy grid maps to represent space. In this case, a frontier is defined as the boundary between explored and unexplored space, and the rationale is that by moving towards large frontiers, a robot will be quicker in discovering more unknown areas. Other possible approaches are based on random exploration, or variations of the frontier based approach. For example, distance to frontiers may be considered to break ties between equally large frontiers. In many of these approaches the temporal dimension is not explicitly considered. That is to say, that while heuristics are introduced to expedite the exploration process, time is most often not an explicit metric or constraint.

In this work, we deviate from existing literature by considering three modifications to the basic setting. First, we assume the robot does not build a metric model of the environment, like an occupancy grid map, but rather a topological model with semantic annotations. This is called *toposemantic* model and will be further expanded later. Second, the objective of the robot is not to build a spatial model per se, but rather to explore the unknown environment until a target location is discovered. The model is functional to this objective, e.g., to avoid revisiting areas already explored. To this end, we assume that the target location is provided in a format compatible with the sensorial capabilities of the robot, so that it can detect when the desired place has been reached. Finally, we introduce a temporal constraint, i.e., a time $\mathcal{T}$ such that the exploration task is considered not solved if after time $\mathcal{T}$ the robot has still not reached the location it is looking for.

Exploration is a fundamental ability in mobile robotics that has been extensively studied in the last three decades. With the current explosive growth in robotics applications, the area continues to grow, particularly in considering extensions and special cases of exploration not studied in the past. In this paper we consider the topic of *efficient* exploration using a semantic topological oriented map. Efficient, in this case, means that a robot is expected to complete its assigned task within a given temporal deadline. To accomplish this task, the robot does not build a metric map, but rather incrementally builds a topological model where the environment is represented by a graph. We opt for topological models for various reasons. First, it is known that humans make use of topological models for spatial awareness and navigation [2]. Second, there is an interest in moving away from representations such as occupancy grid maps that are tightly integrated with range finders. Finally, topological maps are more compact and use less memory. To handle temporal deadlines, there exist at least two approaches. First, one could combine multiple objectives (e.g., time, safety, explored area) into a single objective function, for example with a linear combination, and then plan using standard methods like Markov Decision Processes. Instead, we rely on our recent planning algorithms using constrained Markov decision processes (CMPDs). This approach allows optimizing with respect to one objective (say reaching a goal), while satisfying one or more constraints, like time to completion and probability of collision. This approach has the advantage of not requiring a combination of intrinsically heterogeneous quantities.

2. RELATED WORK

Exploration is one of the main challenges for robots facing a new environment and there is no consensus on the best strategy and representation. Ultimately, these are application specific, and are still an active research topic, as witnessed by continued publications in this area. Vidal et al. recently studied robotic underwater exploration [3] while Tung et al. propose a method using the visual saliency of objects and the environment to drive the robot towards salient objects using a 3D occupancy map [4]. Other authors have recently proposed to use a deep reinforcement learning and included topological and structural information about a building to improve exploration [5]. While metric maps and occupancy grid maps have been common in multiple applications, new approaches have been developed, and topological and semantic maps have shown the potential to be used in complex applications (see [6] for a relatively recent survey). Multiple labeling methods and scene recognition [7] have been proposed, with the main goal of getting a good understanding of the objects and their relationships [8] [9]. For indoor environments, Quattrini et al. use semantic information combined with geometric information to

improve the exploration of a map with privileged areas that are defined by humans as first goals for the robots [10]. Temporal deadlines in robotics have been extensively studied for scheduling and coordination tasks, for example, when multiple robots must coordinate their actions so that the relative temporal ordering becomes relevant [11] [12]. We recently studied a class of multi-objective planning problems where the temporal deadline becomes an additional constraint to be met while optimizing some other objective function, like for example, the probability of successfully reaching a location or gathering some data [13] [14] [15]. These methods are the foundation of the planning method we consider.

3. BACKGROUND ON SPATIAL REPRESENTATION AND PLANNING UNDER TEMPORAL CONSTRAINTS

Semantic Topological Oriented Maps

In this subsection we define *semantic topological oriented* maps (or toposematic maps, for brevity) as an extension to the classic definition of topological maps. Topological maps model space as a graph $G = (V, E)$ with vertices representing places and edges modeling the ability to move between the places represented by vertices. Our model builds upon two assumptions. First, indoor environments are normally subdivided into corridors and rooms arranged along orthogonal directions. This observation was already made and exploited in [10]. Second, we assume that the robot is equipped with a device providing absolute orientation (compass). Starting from these two hypotheses, without loss of generality, we assume that the walls and corridors of the building are arranged along the four cardinal directions that will be in the following abbreviated as N (north), S (south), E (east), and W (west). An oriented topological map exploits these assumptions to define the relations "to the right of" and "to the left of" between vertices in the graph. Vertices in a graph may have degree one (rooms, or corridor dead ends), two (corridors), three (T junction), or four (four way intersection). Thanks to the compass, each edge can then be labeled as E-W or N-S according to its direction. Accordingly, two adjacent vertices are said to be along the N-S (north-south) direction if the edge connecting them has the N-S label and we similarly define adjacent vertices along the E-W direction. To define the relationships "to the right/left of" for elements aligned along the N-S direction, we assume that the robot faces W, whereas for elements along the E-W direction we assume the robot faces N. Figure 1 illustrates this approach. The three vertices on the left are along the E-W direction. Based on our assumption that the robot points north to define left/right relationships, vertex $c1a$ is to the left of $c1b$ and $c1b$ is to the right of $c1a$. On the right, $c2b$ is on the right of $c2c$ and on the left of $c2a$. The following definition formalizes the structure of a map.

Definition 1 A *semantic topological oriented map* is defined as $\mathcal{M} = (V, E, \mathcal{L}, \mathcal{D}, \mathcal{S})$ where*:*

- (V, E) are the vertices and edges of a directed graph.
- $\mathcal{L}: V \to \lambda$ assigns a unique semantic label to each vertex, with λ being a finite set.
- $\mathcal{D}: E \to \{E - W, N - S\}$ is a function assigning a direction to each edge.
- $\mathcal{S}: E \to \{L, R\}$ is a function assigning a label L (to the left of) or R (to the right of) to each edge. The meaning is that the oriented edge is to the left/right of the vertex it originates from (see also Figure 2 for examples).
- If $e = (v_i, v_j) \in E$ is an edge from v_i to v_j and $\mathcal{S}(e) = L$, then $e' = (v_j, v_i) \in E$ and $\mathcal{S}(e') = R$. Likewise, $e(v_i, v_j) \in E \wedge \mathcal{S}(e) = R \Rightarrow e' = (v_j, v_i) \in E \wedge \mathcal{S}(e') = L$.
- For each couple of adjacent vertices, $\mathcal{D}(v_i, v_j) = \mathcal{D}(v_j, v_j)$.

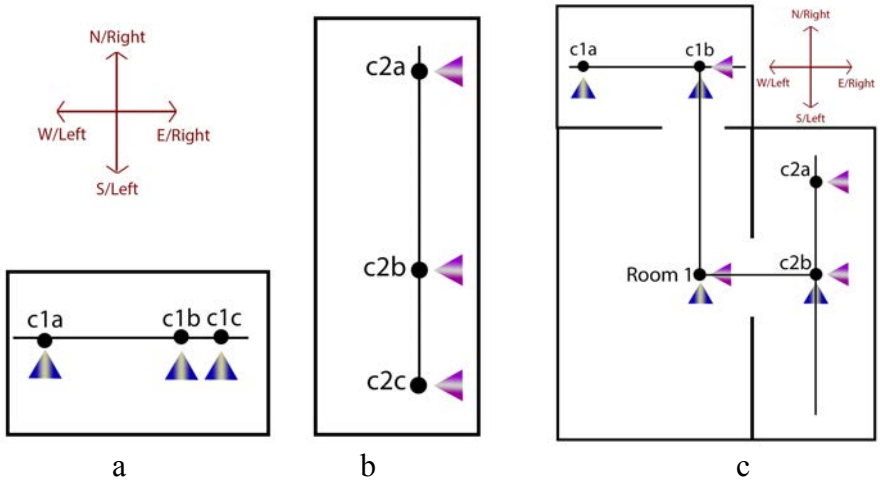

Figure 1: **a:** *three adjacent vertices along the E-W direction. c1a is on the left of c1b, and c1c is on the right of c1b.* **b:** *three vertices along the N-S direction with c2a on the right of c2b and c2c on the left of c2b.* **c:** *two adjacent vertices along the E-W direction, c1a is on the left of c1b. One room 1 on the left of c1b. Two adjacent vertices along N-S with c2a on the right of c2b, and room 1 to the left of c2b.*

The last two conditions establish two consistency constraints. If it is possible to go from v_i to v_j along one direction (say N-S), then it is possible to go from v_j to v_i along the same direction but opposite orientation (say S-N). Second, if there is an edge from v_i to v_j indicating that v_j is to the left of v_i, then there must also exist the opposite edge from v_j to v_i indicating that v_i is to the right of v_j.

Collectively, the algorithms providing $\mathcal{L}, \mathcal{S}$ and $\mathcal{D}$ will be in the following indicated as *Intersection Detection System* (IDS). Thanks to recent advances in perception [16], we assume that the IDS system can recognize if a detected intersection or corridor is new or revisited. Figure 2 shows how a toposemantic map can be used to represent the interior of a building.

Planning Under Temporal Constraints

We shortly recap our planning approach based on CMDPs, but due to space limitations we refer to [17] [13] for more details. We assume that the reader is familiar with MDPs (Markov Decision Processes) and related terminology [18]. A finite, stationary, discrete time CMDP is defined as $(\mathbf{X}, A, \beta, c, \{c_i\}_{i=1}^{L}, \mathcal{P}, \{D_i\}_{i=1}^{L})$ where: $\mathbf{X}$ is a discrete set of states; A is a finite set of actions; β is the probability distribution of the initial state; $c: \mathbf{X} \times A \rightarrow \mathbb{R}_{\geq 0}$ is a primary non-negative cost function with $c(x, a)$ being the cost of executing action a in state x; $c_i: \mathbf{X} \times A \rightarrow \mathbb{R}_{\geq 0}$ are L secondary cost functions with $c_i(x, a)$ being the i-th cost incurred when executing action a in state x; $\mathcal{P}(x, a, y)$ is the transition probability, i.e., the probability of transitioning from state x to state y when action a is executed; and D_i are L non-negative constants. A policy $\pi(x)$ defines with which probability each action should be taken when in state x. Let $c(\pi)$ be the overall primary cost incurred when following policy π, and $c_i(\pi)$ be the overall L additional costs. The CMDP problem is therefore defined as follows

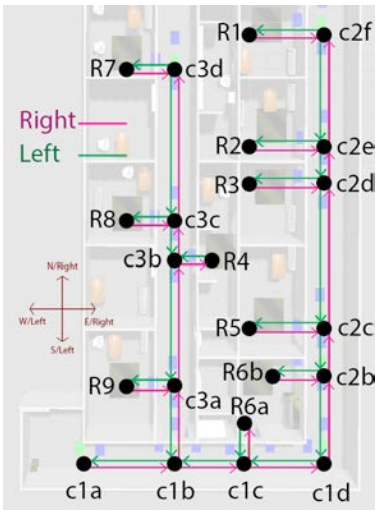

Figure 2: Toposemantic map: vertices with a label Rx are rooms, while corridors have labels of the type Cx. Edges with a R label are purple, while edges with a L label are green.

$$\pi^* = \arg_\pi \min c(\pi)$$
$$s.t. \quad c_i(\pi) \leq D_i \quad 1 \leq i \leq L$$

i.e., we aim at following a policy π^* that minimizes the primary cost subject to bounds on the L secondary costs. The primary and secondary costs are defined as:

$$c(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} c\big(X_t, \pi(X_t)\big)\right] \qquad c_i(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} c_i(X_t, \pi(X_t))\right]$$

To fulfill a navigation task with temporal deadlines, we use one primary cost and two secondary costs. The primary cost is set to 0 in the goal state and to 1 elsewhere. The two secondary costs are the global failure probability and the cumulative time to reach the target vertex. Accordingly, the problem is then to compute a policy reaching the goal state while limiting the time spent and the failure probability. To apply the CMDP planner to the exploration task, the state set X will be the set of vertices V, such that the policy can be used to navigate the robot to a vertex. The set of actions A is instead a finite set of maneuvers representing basic motion abilities. Each maneuver is characterized by an expected time spent to execute it, and a failure probability. In section 6 we describe the maneuvers and the associated parameters.

4. INCREMENTAL MAP CONSTRUCTION, NAVIGATION, AND EXPLORATION

We now describe how a toposemantic map $\mathcal{M}$ can be incrementally built and used for navigation by a robot exploring an unknown indoor environment while attempting to reach a destination by a given deadline. The IDS module determines when a robot has reached a vertex (either new or formerly visited) and how many outgoing edges are emanating from the vertex. The $createNewNode$ (sketched in Algorithm 1) is called every time IDS identifies a corridor intersection or a room that was not formerly visited. The function takes as parameters the new vertex v' to be added and the set of edges emanating from it, and it updates the set of vertices and edges assigning labels. Unvisited edges, i.e., edges towards parts of the graph still to be explored,

are of the type $(v', v_\emptyset)$, to indicate they lead to unexplored areas ($v_\emptyset$ is used to model unexplored space). In the function, labels are set to satisfy the constraints listed in definition 1.

1. **Algorithm** $createNewNode(v', edges)$
2. Let v be the vertex the robot came from
3. $V \leftarrow V \cup \{v'\}$
4. $E \leftarrow E \cup \{(v, v'), (v', v)\}$
5. Set $\mathcal{L}(v')$ as per IDS (room or corridor)
6. Set $\mathcal{D}(v', v)$ and $\mathcal{D}(v', v)$ as per compass
7. Set $\mathcal{S}(v', v)$ and $\mathcal{S}(v', v)$ as per compass
8. **for all** unvisited outgoing edges from v' **do**
9. Add edges of type $(v', v_\emptyset)$ and set labels $\mathcal{S}, \mathcal{D}$

Algorithm 1: adding a new node to the map

The algorithm to navigate from the current vertex v_c to a desired target vertex v is sketched in Algorithm 2. The algorithm takes two parameters, i.e., v and a temporal deadline t_d. Later on we discuss how these are selected as part of the exploration module. To navigate to the vertex, a CMDP is built considering the given temporal deadline and a probability of failure P_f (for simplicity assumed constant in the following). At each step, the policy π^* is used to determine the primitive a to be executed from the current vertex. Termination may happen because the robot reaches the desired local target v or the global target $\mathcal{G}$, but also if the global temporal deadline expires. In this case the robot returns failure.

1. **Algorithm** $NavigateToVertex(v, t_d)$
2. $CMDP \leftarrow BuildCMDP(\mathcal{M}, v, t_d, P_f)$
3. $\pi^* \leftarrow Solve(CMDP)$
4. **while** $time < \mathcal{T}$
5. $v_c \leftarrow$ current vertex
6. **if** $v_c = \mathcal{G}$ **then**
7. **return** Global Success
8. **if** $v_c = v$ **then**
9. **return** Local Success
10. $a \leftarrow \pi^*(v_c)$
11. execute a
12. **return** Failure

Algorithm 2: navigation algorithm

Since the robot starts with an empty map, it initially randomly moves around until it has K vertices in the graph. In all our experiments described in the following, K is fixed to 3. Finally, in *Algorithm 3* we show how the overall exploration task is solved. After creating the initial map $\mathcal{M}$, the robot enters a loop that continues until either the global temporal deadline $\mathcal{T}$ expires, or the robot succeeds in reaching $\mathcal{G}$. At every iteration, the function $ExplorationNext$ returns the vertex v to move to and a temporal deadline t_d, i.e., how much time the robot should spend to reach v. To determine these two quantities, $ExplorationNext$ uses the current map, as well as the variable $time$, indicating how much time passed since the task started. As $time$ grows, more

stringent temporal deadlines t_d will be returned. The robot then attempts to navigate to the assigned vertex v and once there, it randomly picks an outgoing edge and follows it until IDS indicates that a new vertex has been reached.

```
1.  Algorithm ExplorationTask(𝒯, 𝒢)
2.  ℳ ← InitializeGraph()
3.  while time < 𝒯 do
4.      v, t_d ← ExplorationNext(ℳ, time)
5.      flag ← NavigateToVertex(v, t_d)
6.      if flag = Local Success then
7.          Pick random edge of type (v, v_∅) out of v
8.          Follow edge e of type (v, v_∅) and find new node v'
9.          CreateNewNode(v', edges)
10.     else
11.         return flag
12. return Failure
```

Algorithm 3: global exploration algorithm

5. EXPLORATION ALGORITHMS

We discuss five different strategies that can be used to guide the robot through its exploration task. Each of these strategies represents a different way to select the vertex v returned by the function $ExplorationNext$ used in *Algorithm 3*. At the end of the section we also discuss how we set the temporal deadline t_d.

Random Strategy Exploration

The random exploration strategy is our baseline approach. It returns a random vertex among all vertices with one or more outgoing unvisited edges, i.e., edges towards $v_\emptyset$. If there is more than one vertex with unvisited edges, a uniform probability distribution is used to make a choice. It is easy to prove that this strategy will eventually explore the whole environment, but it is likely to be inefficient.

Topological Frontier

Topological frontier is the analogue of the well-known frontier based exploration algorithm. In frontier-based exploration, frontiers are defined as the regions on the boundary between explored and unexplored space, and the robot then moves to frontier to expand the explored space. In our topological domain, the boundary between known and unknown parts of the environment is identified by unexplored edges. Accordingly, the topological frontier algorithm sorts all nodes by the number of outgoing unexplored edges, and it randomly selects one among those with the highest number of outgoing edges. In this latter case a uniform distribution among all candidates is used.

Topological Frontier with Normalized Distances

In the topological case the cost to move to a vertex in the graph is set to the number of edges in the path connecting the current vertex to a target location. To consider distances in the topological frontier algorithm, we choose a closer vertex when one or more equivalent ones are present. To combine heterogeneous quantities (number of outgoing unexplored edges and

distances), we use a linear combination of normalized quantities (see e.g., [19]) to assign a value to each vertex in the map. We then select the vertex maximizing this metric. To be specific, let $\mathcal{V} \subset V$ be the set of vertices with one or more outgoing unexplored edge. For each vertex $v \in \mathcal{V}$ we compute the following quantity

$$S(v) = \gamma \frac{\deg(v)}{\max_{v' \in \mathcal{V}} \deg(v')} - (1-\gamma) \frac{d(v)}{\max_{v' \in \mathcal{V}} d(v')}$$

where $deg(v)$ is the number of outgoing, unexplored edges and $d(v)$ is the distance in the topological map, defined as the number of edges in the shortest path in the graph. The function then returns the node in $\mathcal{V}$ with the highest value for $S(v)$. In the experiments presented in the following section we set $\gamma = 0.5$. This approach then tries to drive the robot to frontiers that are large and near.

Semantic: Explore Corridors First

The first semantic exploration strategy relies on the labels attributed to vertices in the graph. In particular, it exploits the assumption that we can distinguish between rooms and corridors. Inspired by the work by Quattrini et al. , the robot prioritizes corridors when selecting where to go next [10]. That is to say that if among the vertices with unexplored outgoing edges there are both corridors and rooms, the robot always selects corridors first. Rooms are selected only when no corridor vertices can be selected.

Semantic Complete Corridors First

Finally in our second semantic strategy, before moving to a different corridor the robot finishes exploring the corridor it is located in. This is complementary to the previous one. To accomplish this, the semantic label $\mathcal{L}$ of a vertex and the directions of its adjacent edges are considered. Here, a corridor is defined as a path along the graph $G = (V, E)$ such that all vertices are labeled as corridor vertices by the function $\mathcal{L}$, and all edges connecting the vertices have the same label $\mathcal{D}$, i.e., they are all along the N-S or E-W direction. For example, in figure 2 $c1a - c1b - c1c - c1d$ is a corridor along the E-W direction that cannot be further extended because other vertices of type corridor can only be reached traversing edges along the N-S direction. The robot enters a new corridor only once the previous corridor can no longer be extended. This approach further capitalizes one aspect in *Algorithm 3*, i.e., once a vertex is reached a random outgoing edge is picked and traversed, too. This has the effect of exploring rooms connected to the corridor while this is being explored.

Temporal Deadline

Finally, in this subsection we describe one possible way to pick the temporal deadline assigned to reach a given vertex. The rationale is that as the global deadline $\mathcal{T}$ is approaching, the robot should speedup. To this end, we opt for a very simple schedule. Once a vertex v is selected, we define a preliminary deadline B equal to the number of edges between the current vertex v_c and v multiplied by a constant. Then, if the time available to complete the task is larger than $0.8\mathcal{T}$, we set $t_d = B$. If instead the time to complete the task is between $0.5\mathcal{T}$ and $0.8\mathcal{T}$ we set $t_d = 0.5B$. Finally, if the time to complete the task is less than $0.5\mathcal{T}$, we set $t_d = 0.3B$. The objective of this simple approach is to exemplify the effectiveness of our temporal aware planning strategy, and we leave to future work the study of how to implement more sophisticated approaches.

6. EXPERIMENTAL VALIDATION

Setup

As a preliminary step towards evaluating strengths and weaknesses of the proposed approach, we perform an extensive set of tests using ROS and Gazebo. The simulated environment is a faithful replica of one of the engineering buildings of the University of California, Merced, and its model in Gazebo was built from its architectural CAD design (see Figure 3).

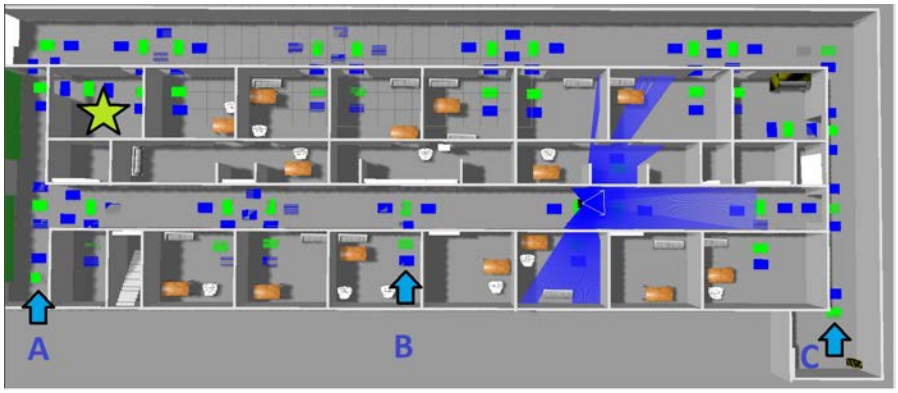

Figure 3: Three starting points (A, B, C) and one target (star).

The simulated robot is a Pioneer 3AT with limited sensing capabilities compatible with our former assumptions. In particular, the robot is equipped only with a compass, a laser range finder to avoid obstacles, and a logical camera to detect features in the environment. The logical camera is a ROS plugin that returns the position and orientation of any object in its cone of vision with respect to the robot.[1] The logical camera abstracts the implementation of the IDS system based on computer vision. This is obtained by embedding unique features in the environment that can be detected by the camera (these are displayed as green and blue tags in Fig. 3). It shall be noted that even though the simulated robot is equipped with a laser range finder, it is only used for obstacle avoidance when moving along corridors or entering doors. To test our navigation system under realistic conditions, a 25% error is added to linear and angular velocity commands, and a 5% error is added to orientation readings. In all our tests the robot starts without any preliminary information about the environment it is exploring.

Maneuvers

To navigate the environment, the action set A for the CMPD includes six maneuvers. Specifically, there are three elementary motions, and each can be executed fast or slow. The first maneuver is *go through a corridor*. This maneuver moves the robot forward (i.e., keeping the same orientation) while trying to keep an equal distance from the walls on either side. The second maneuver is *go through door on the right*. This will turn the robot to its right (relative to its current location and orientation) and move through a door, as identified by the IDS system. The third maneuver is the symmetric *go thorough door on the left*.

[1] http://wiki.ros.org/ariac/Tutorials/SensorInterface

Results

Here we compare the five exploration strategies on an exploration task where we vary the complexity of the navigation task, defined as the distance between the robot start location and the target location, and the assigned temporal deadline. The environment is shown in Figure 3, where the goal location G is the room marked with the green star, whereas the three different places marked A, B and C identify the different start points considered. Throughout the simulation, the probability of failure P_f used when computing the CMDP policy was set to 0.99. For each starting point we consider four different temporal deadlines, and we then execute 50 trials. A trial is considered a failure if the robot has not reached the target location by the given deadline, or collides with the environment while navigating. Overall, 3000 independent tests were executed to evaluate the five exploration strategies. Table 1 summarizes the performance for the five exploration methods we propose. The columns give the starting location, temporal deadline expressed in seconds, Time Spent (s), Success (%) for each type of exploration: Random (Ra), Frontier (Fr), Normalized Frontier (NF), Semantic Explore Corridors First (S1), and Semantic Complete Corridors First (S2). For each combination of start location and temporal deadline, we provide the success rate, the average time spent to reach the target. Note that the average time is given for successful runs only, because unsuccessful runs may result from exceeding the temporal deadline or because of collisions with the environment and therefore it would not be meaningful to average over unsuccessful runs, too. Table 2, instead, analyzes in greater detail the causes of the failures for each strategy.

To ease the comparison, the results are also visually compared in Figures 4 and 5. Unsurprisingly, the random strategy is the most effective when starting from location A, except when the shortest deadline is enforced. This is somewhat expected, given that other strategies tend instead to expand the map in a principled way that may push them far from the target location that is relatively close to A. Moving farther from the target increases the time to complete the mission, and ultimately the failure rate, as this translates increased chances to miss the temporal deadline or to collide with the environment. However, in the other cases this strategy is less effective, and performs very poorly for the most challenging case C. All things considered, the topological frontier with normalized distances appears to be most effective when the temporal deadline is not too demanding. However, as the deadline becomes more stringent its advantage seems to vanish and it becomes more or less comparable to the topological frontier. The semantic strategies, on the contrary, appear to have a performance that is less dependent on the start location. One final comment should be made regarding the success rates, as they may appear to be on the low end. This is due to the fact that the temporal deadlines are strict, and this boosts failure rates for all algorithms. A strict temporal deadline is not only harder to meet, but it also forces the CMDP planner to utilize more aggressive maneuvers, thus possibly also increasing the number of failures due to collisions with the environment.

7. REPRODUCING OUR RESEARCH

The final contribution of this submission is in ensuring that our research can be reproduced, as this is a topic of increasing importance in robotics. In [20], we find that "*A study is reproducible if you can take the original data and the computer code used to analyze the data and reproduce*

		Time Spent (Seconds)					Success Rate %				
S	TD	Rand	Frontier	Norm. Frontier	Sem 1	Sem 2	Rand	Frontier	Norm. Frontier	Sem 1	Sem 2
A	680	146.542	199.429	47.45	246	71.9	96	56	40	40	40
A	477	139.106	54.9333	76.087	53.9	48.4	94	30	46	30	34
A	272	99.825	31.125	39.1	35.1	75.8	80	32	40	36	36
A	68	42.5385	35.6471	34.6	31.2	48.4	26	34	30	26	30
B	2899	1006.36	345.3	883.607	347	656	50	60	56	54	58
B	2030	1026.9	373.923	522	361	543	42	52	56	60	54
B	1160	681.148	391.581	373.412	342	580	54	62	34	58	46
B	290	0	218.824	199.1	222	219	0	34	20	22	24
C	5628	1495	336.926	648.71	295	336	8	54	62	50	46
C	3940	1810.43	363.448	599.25	322	347	14	58	48	54	76
C	2251	1363.33	364.607	507.552	331	548	6	56	58	56	48
C	563	0	330.72	368.353	265	396	0	50	34	46	20

Table 1: Performance of the 5 exploration strategies.

		Fails because of Collision %					Fails because of DL not met %				
S	TD	Rand	Frontier	Norm. Frontier	Sem 1	Sem 2	Rand	Frontier	Norm. Frontier	Sem 1	Sem 2
A	680	4	36	24	40	28	0	8	36	20	32
A	477	4	36	12	36	16	2	34	42	34	50
A	272	2	24	10	16	2	18	44	50	48	62
A	68	8	2	8	10	2	66	64	62	64	68
B	2899	50	38	32	30	34	0	2	12	14	8
B	2030	58	48	36	36	28	0	0	8	4	18
B	1160	46	38	44	28	44	0	0	22	14	10
B	290	16	16	44	18	24	84	50	36	60	52
C	5628	92	46	28	38	50	0	0	10	12	4
C	3940	86	40	36	36	20	0	2	16	10	4
C	2251	94	40	36	42	36	0	4	6	2	16
C	563	100	40	30	40	40	0	10	36	14	40

Table 2: Analysis of failures for the 5 exploration strategies.

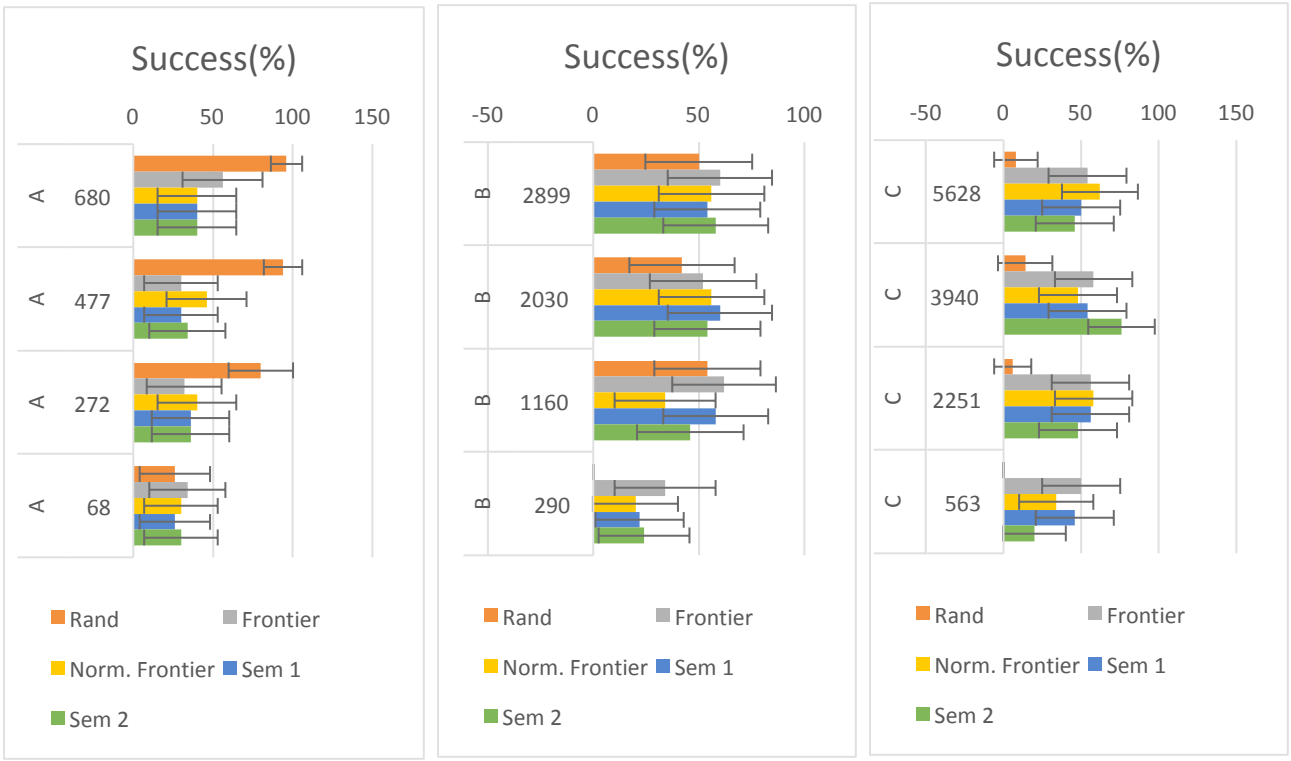

Figure 4: Percentage of Success for all the exploration strategies.

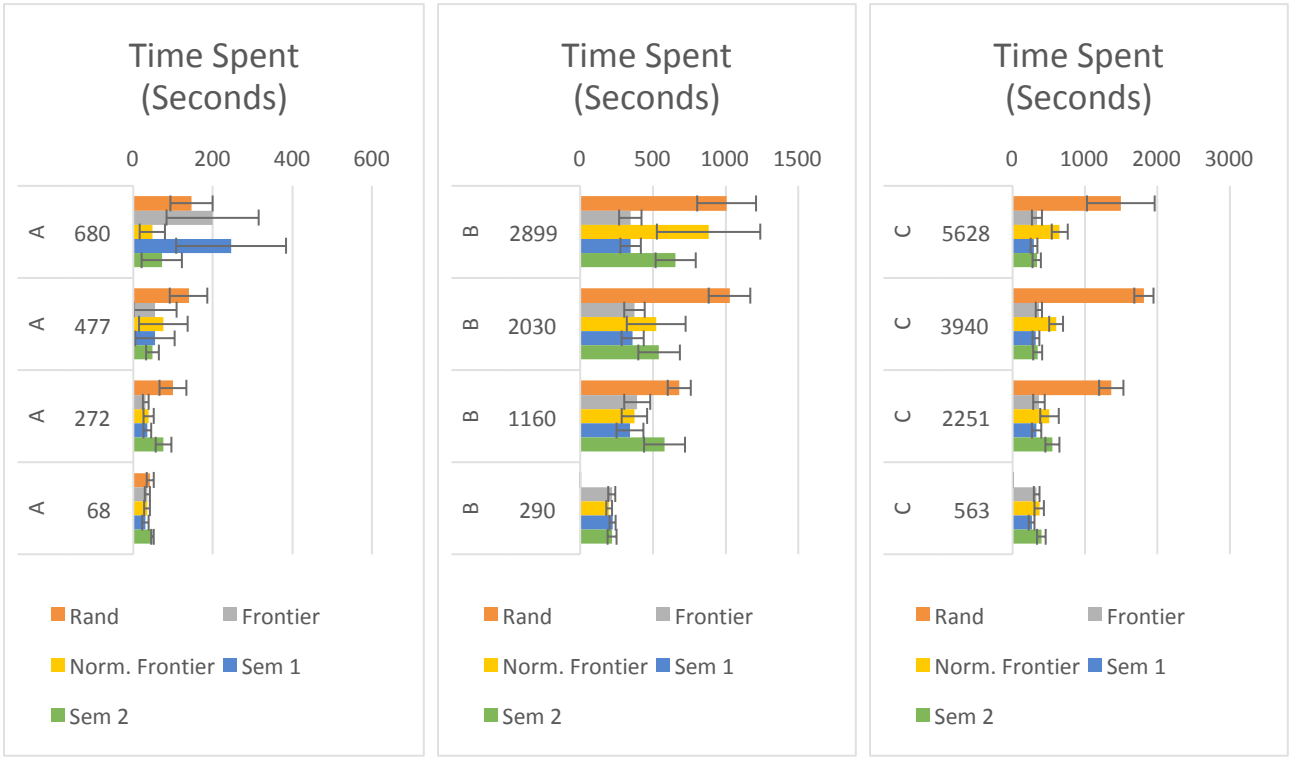

Figure 5: Time spent for all the exploration strategies.

all of the numerical findings from the study". Accordingly, the resources we provide in the following aim at making our study is reproducible. However, due to the stochastic nature of the experiments we performed, reproducing the same numerical findings is not a foregone conclusion, in particular if the user relies on a system less powerful than the one used for the experiments presented in section 6. A more realistic objective is aiming at reaching the same conclusions in terms of relative value of the exploration strategies we presented. In [21] a set of guidelines for *good experimental methodologies* in robotics is given. According to such guidelines, this work should be characterized as ``experimental'' and shall be furthermore classified as research in ``Autonomy/Cognitive Tasks.'' The experimental part of this paper is based on ROS/Gazebo and is therefore particularly suited for being reproduced by a third party. This would be much more challenging if results were obtained on a physical P3AT robot, because of the countless variables influencing the final results. While the robotics community is becoming increasingly aware of the importance of this aspect, *best practices* are still being defined, and standard tools to promote code reproducibility (e.g., Code Ocean) are not necessarily best suited or ready yet for all robotics research. This is in particularly relevant when considering an end-to-end system relying on multiple external libraries to perform heterogenous operations like mathematical computation, navigation, and more. Considering these challenges, in the supplementary materials section of this work, we have provided an image for a virtual Linux machine including our code, all libraries necessary to run it, and a set of scripts to re-run all the experiments to produce the results presented in this paper. The virtual machine image is based on VirtualBox, a free software available for a variety of operating systems, including Windows, OsX, and Linux[2]. In addition, we have also developed a technical document with step-by-step instructions to download and boot the virtual image, and run all experiments described in this paper. It shall be noted that other approaches could be viable, too, like Docker. With these premises, the experimental process that led to the results presented before is reproducible by a third party. As mentioned above, however, a third-party will not necessarily obtain the same numerical outcomes because of two sources of randomness. First, the algorithms themselves feature various steps relying on randomized choices, e.g., to break ties. Second, the simulation environment is influenced by the underlying platform and will therefore not produce exactly the same results. Moreover, by running the code through a virtual image, an inevitable slowdown will affect the results, and some computer systems may be unable to run the system altogether. To mitigate this problem, we have also added a complete set of instructions on the various packages to install on a native Linux system to enable reproducibility even without the VirtualBox image. Nevertheless, due to continuous updates in software packages, as well as discontinuations, this second approach may not be stable in the long run. To the best of our knowledge, this is the first example of a fully reproducible study of exploration algorithms.

8. CONCLUSIONS AND FUTURE WORK

In this paper have studied an exploration task with temporal constraints. The objective for the robot is to enter an unexplored area and reach a target location within a given temporal deadline. Our system incrementally builds a spatial model called *semantic topological oriented map*. The model enhances classic topological maps by adding semantic labels (corridor/room) and

[2] www.virtualbox.org

relationships of the type to-the-right-of and to-the-left-of. Key to our approach is the assumption that walls in the environment are aligned along orthogonal directions. This assumption is very common in most buildings. The proposed spatial model has been coupled with our recently proposed planner based on CMDPs and on top of this we proposed and analyzed five different exploration techniques exploiting the proposed model. Through thousands of simulated runs, it appears that our method dubbed *topological frontier with normalized distances* works best. In their current formulation, methods on semantic information are slightly less competitive, although it ought to be acknowledged that only very limited semantic information was used. Finally, we have provided all resources to ensure a third party to fully reproduce our results -- a first in the area of reproducible robotics. With respect to this aspect, we believe it will be important for the robotics community to reach a consensus on the tools and infrastructure needed to disseminate reproducible research.

REFERENCES

[1] B. Yamauchi, "A Frontier-Based Approach for Autonomous Exploration," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, 1997.

[2] B. Kuipers, "The spatial semantic hierarchy," *Artificial Intelligence,* no. 119, p. 191–233, 2000.

[3] E. Vidal, J. D. Hern and K. Isteniˇ, "Optimized environment exploration for autonomous underwater vehicles," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 6409-6416, 2018.

[4] T. Dang, C. Papachristos and A. Kostas, "Visual Saliency–aware Receding Horizon Autonomous Exploration with Application to Aerial Robotics," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 2526-2533, 2018.

[5] D. Zhu, T. Li, D. Ho, C. Wang and M. Q. Meng, "Deep Reinforcement Learning Supervised Autonomous Exploration in Office Environments," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 7548-7555, 2018.

[6] K. Antonios and G. Ioannis, "Semantic mapping for mobile robotics tasks: A survey.," *Robotics and Autonomous Systems,* no. 66, p. 86–103, 2015.

[7] S. Garg, N. Suenderhauf and M. Milford, "Don't Look Back: Robustifying Place Categorization for Viewpoint- and Condition-Invariant Place Recognition," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 3645-3652, 2018.

[8] M. Durner, M. Brucker, A. Wendt, P. Jensfelt, K. O. Arras and R. Triebel, "Semantic Labeling of Indoor Environments from 3D RGB Maps," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1871-1878, 2018.

[9] L. F. Posada, A. Velasquez-lopez, F. Hoffmann and T. Bertram, "Semantic Mapping with Omnidirectional Vision," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1901-1907, 2018.

[10] A. Quattrini Li, R. Cipolleschi, M. Giusto and F. Amigoni, "A semantically-informed

multirobot system for exploration of relevant areas in search and rescue settings," *Autonomous Robots,* vol. 40, no. 4, p. 581–597, 2016.

[11] M. C. Gombolay, R. J. Wilcox and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Transactions on Robotics,* vol. 1, no. 34, p. 220–239, 2018.

[12] J. Guerrero and G. Oliver, "Swarm-like Methodologies for Executing Tasks with Deadlines," *Journal of Intelligent & Robotic Systems,* vol. 68, no. 1, pp. 3-19, 15 9 2012.

[13] Y.-L. Chow, M. Pavone, B. Sadler and S. Carpin, "Trading safety versus performance: Rapid deployment of robotic swarms with robust performance constraints," *ASME Journal of Dynamical Systems, Measurements and Control,* vol. 3, no. 137, p. 031005, 2015.

[14] S. Carpin and S. Feyzabadi, "Planning using hierarchical constrained markov decision processes," *Autonomous Robots,* vol. 8, no. 41, p. 1589–1607, 2017.

[15] J. L. Susa Rincon, P. Tokekar, V. Kumar and S. Carpin, "Rapid deployment of mobile robots under temporal, performance, perception, and resource constraints.," *IEEE Robotics and Automation Letters,* vol. 4, no. 2, p. 2016–2023, 2017.

[16] E. Garcia-Fidalgo and A. Ortiz, "Vision-based topological mapping and localization methods: A survey," *Robotics and Autonomous Systems,* vol. 64, pp. 1-20, 2015.

[17] E. Altman, "Constrained Markov decision processes with total cost criteria: Occupation measures and primal LP," *Mathematical methods,* no. 43, p. 45–72, 1996.

[18] D. P. Bertsekas, "Dynamic Programming & Optimal Control," *Athena Scientific,,* vol. 1 and 2, 2005.

[19] S. Carpin, N. Basilico, D. Burch, T. Chung and M. K¨olsch., "Variable resolution search with quadrotors: theory and practice," *Journal of Field Robotics,* vol. 5, no. 301, p. 685–701, 2013.

[20] R. Peng, "A Simple Explanation for the Replication Crisis in Science · Simply Statistics," 2016. [Online]. Available: https://simplystatistics.org/2016/08/24/replication-crisis/. [Accessed 07 05 2019].

[21] F. Bonsignorio, J. Hallam and A. del Pobil, "GEM Guidelines," 2008.