Metareasoning and Path Planning for Autonomous Indoor Navigation

Susan L. Epstein and Raj Korpan
Hunter College and The Graduate Center of The City University of New York
695 Park Avenue
New York, NY 10065
susan.epstein@hunter.cuny.edu

Abstract

A planner for indoor navigation in an unknown space should anticipate the perils of sensor and actuator errors. The approach described here uses a spatial model whose elements generalize continuous affordances from discrete data to support robust hierarchical planning. A novel reactive controller intervenes to address lack of progress toward a target and to improve plans opportunistically, during their execution. A metareasoning architecture seamlessly integrates these components as hierarchical decision making. Empirical results demonstrate the flexibility of this approach and the role of the spatial model in three challenging real worlds.

Introduction

To navigate successfully to a specified location (*target*) in a complex indoor space, a robot needs a plan. A typical plan is a sequence of locations (*waypoints*) from the robot to the target. Given the likelihood of errors in a robot's sensors and actuators, however, such a plan often fails. The thesis of our work is that a robot can efficiently learn, robustly plan, and effectively navigate in challenging worlds without a map. Our approach learns a hierarchical spatial model of *freespace*, unobstructed area that supports hierarchical planning. So equipped, our system can ignore many details until execution time, can interrupt plan execution for failure to make progress, and can revise a plan opportunistically. The principal results reported here are a system that learns and exploits a continuous representation of its world from discretized data on relatively few targets, and a demonstration, in simulation, of its effective metareasoning in large, complex, indoor spaces (henceforward, *worlds*).

A *task* here is navigation to a target in a world like Figure 1 with a laser range finder as the primary sensor and without an input map. (Figures in this paper show walls and other fixed obstacles only for the reader's convenience. The robot has no access to such a map, yet still traverses these worlds in real time.) The range finder measures the distance to the nearest obstruction in multiple directions, as visualized in Figure 2(a). From its experience in a world, our system, *SemaFORR*, learns a spatial model of the freespace there, and improves its performance as it augments its model. Because robot sensors and actuators have limited precision, the outcome of a decision is somewhat uncertain. Thus the robot must repeatedly *localize*, that is, detect its precise location and orientation in the world. We assume here that the robot has accurate localization through, for example, visual-inertial odometry (Forster et al. 2017). Our experiments demonstrate that brief preliminary exploration can provide an initial model that empowers planning.

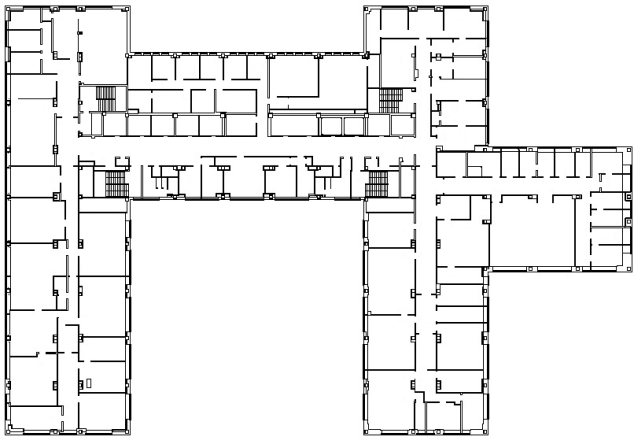


Figure 1: H10, a challenging world for a robot without a map

Although both indoor navigation and autonomous driving have a target as their goal, they present different challenges and opportunities. Driving is constrained to a tiny fraction of outdoor space, a network of clearly delineated, well-mapped roads. A complex indoor space, however, is mostly freespace, designed by an architect to support not only navigation but also security, aesthetics, or privacy. Moreover, the worlds studied here allow proportionally far more space that the robot can occupy than the space that is unavailable to it.

SemaFORR formulates a plan as a sequence of freespace elements rather than waypoints. This allows it to delay decisions about how to move between elements until the next plan step must be executed. Metareasoning relies on a reactive controller that monitors and can interrupt plan execution. The next section provides background and related work. Subsequent sections describe the model, the planners, and the architecture that integrates them, followed by an empirical demonstration, and a discussion of the results.

Background and Related Work

An autonomous robot navigator is an embodied computational system embedded in a world. It relies on a *controller* to make its decisions. SemaFORR is written for ROS, the international standard for robot operating systems, which allows modular construction and platform-specific implementations (Quigley et al. 2009).

Algorithms that represent space continuously soon become intractable in large, complex spaces. Most work in robotics, therefore, discretizes the robot's body (often, as we do here, to a point on a coordinate plane), and represents its perception of a continuous world as a finite set of values. Consider, for example, a robot with *pose* $\langle x, y, \theta \rangle$, where $\langle x, y \rangle$ is the robot's location and θ is its *orientation*. Figure 2(a) visualizes our robot's sensor data, with the robot at the arrow's base (x, y), oriented toward the arrow's head. The blue *rays* measure the distance to the nearest obstruction, here every $1/3^\circ$ along a 220° arc. This data is *local*, that is, only within the range limit of the sensor. It is also *partial* because the sensor lacks a 360° field of view, and because small obstacles between the rays can go undetected, particularly those farther from the robot.

Because real worlds are rampant with spatial irregularities (e.g., moldings, support columns, indentations or protrusions in walls) and sensors are vulnerable to atmospheric distortions and reflective surfaces, a robot's sensors may provide inaccurate data. This makes it difficult to guarantee the accuracy of any learned spatial model.

A *decision point* $\langle x, y, \theta, V \rangle$ is the robot's pose and its *view* V at that location, the data reported by its range finder there. (Sensor data received while the robot is in motion is ignored.) SemaFORR provides a deliberately small *action repertoire*: 6 forward moves of different lengths, 12 clockwise or counterclockwise turns of different degrees, and a pause. At each decision point, SemaFORR senses locally, chooses one action from its repertoire, and executes that action. During a task, SemaFORR records the robot's *path*, a chronological list of its decision points. A *solution* is a path from the robot's initial pose to one within ϵ of its target.

World representation

IEEE currently recognizes three standard ways to represent a world with respect to a coordinate plane (IEEE RAS Map Data Representation Working Group 2015). An *occupancy grid* superimposes a grid of uniform square cells on the *footprint* of the world, the area delineated by its boundaries (e.g., the outer walls of a building). It indicates whether or not each cell is blocked (fully or partially) by obstructions. A *geometric map* lists individual points and line segments that delineate obstacles to support precise distance and angle computation. Finally, a *topological map* is a graph whose nodes represent locations, and whose edges denote pairs of locations reachable from one another, with labels for the known travel distance between them. If a map is not provided, the robot must construct one.

SLAM (Simultaneous Localization and Mapping) localizes the robot and maps the world at once (Durrant-Whyte and Bailey 2006). As the robot moves, SLAM uses proprioceptive and exteroceptive sensor data both to generate a map and to determine the robot's pose in it. Because V is incomplete and possibly inaccurate, and because different poses can generate the same V, modern SLAM methods are probabilistic (Thrun, Burgard, and Fox 2016; Stachniss, Leonard, and Thrun 2016). For now, the most popular probabilistic SLAM approaches remain those that identify the most likely map of the robot's world with maximum a posteriori likelihood (Lu and Milios 1997). They efficiently optimize and increment a global map of the world's static geometry to generate a likely floorplan as the robot moves. Probabilistic SLAM, however, is limited to static landmarks, requires extensive parameter tuning, and is not robust to outliers. Unlike SemaFORR, it does not make decisions or plans (Cadena et al. 2016).

Beyond the IEEE standard's topology graph, many other topological models are possible. They are designed to provide a high-level view of freespace and how portions of it are connected, one over which a controller could reason. Some approaches have learned and represented this topology in first-order logic (Joshi et al. 2012) or as polygons induced with a monocular camera (Stein et al. 2020). SemaFORR's spatial model is a topological graph whose nodes represent continuous freespace.

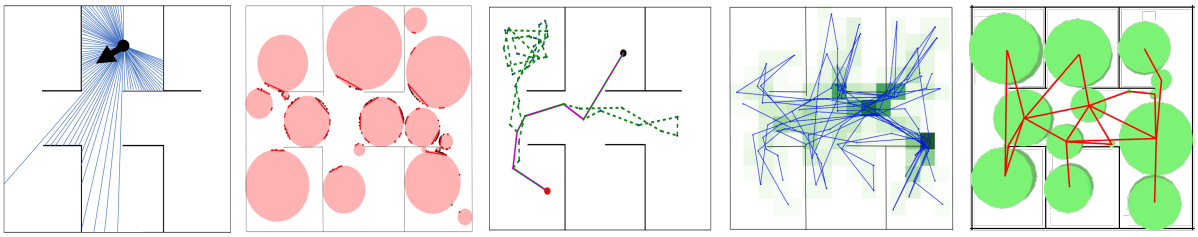
Robot control

Modern robot controllers are hybrids that blend reactivity and planning. A controller necessarily has some low-level mechanisms that take priority in decision making. These *reactive* procedures are immediate responses dictated by both the robot's task and its world. An autonomous car, for example, has provisions for emergency braking to avoid collisions and for lane following to obey driving restrictions.

The complexity of real-world spaces demands that a controller be more than purely reactive. A plan is *correct* if its flawless execution can be guaranteed to achieve a specified goal. For a robot, however, a planner depends upon what is likely an inaccurate world model, and no execution can be guaranteed flawless. A planner is complete if it is guaranteed to halt in finite time, to return a solution if one exists, and to otherwise indicate failure. Complete navigation planners have not scaled well, require perfect knowledge of obstacles, and are rarely implemented (Lavalle 2006). Instead, planning for robot navigation trades off among computational efficiency, optimality, and feasibility.

Modern navigation planners search discrete locations to find waypoints. A *graph planner* searches for paths between nodes that typically represent locations. When the edge label for a pair of mutually accessible nodes is the Euclidean distance between them, A* is an optimal (i.e., shortest path) algorithm (Hart, Nilsson, and Raphael 1968). Its optimality, however, requires that the label is a heuristic underestimator of the effort that remains to the goal. In that case, A* returns the shortest path as a sequence of node waypoints.

A *grid-based planner* builds a graph from an occupancy grid, with a node for each unobstructed cell and an edge for each pair of immediately adjacent cells with unimpeded access between their centers. Each edge is labeled with the Euclidean distance between those centers, so that a graph



(a) Rays cast by the range finder (b) Regions with doors and exits (c) A dotted path and its derived solid trail (d) Trails and their conveyors (e) Skeleton

Figure 2: Affordance detection in Home, a simple artificial world

search algorithm finds a sequence of cell-center waypoints. Grid-based planning is only resolution complete, that is, complete under the discretization it imposes on a world. Too coarse a grid may label too many cells obstructed; if the graph is thereby disconnected, some navigable paths can no longer be detected. An accurate representation of freespace requires a sufficiently fine-grained grid, one so well aligned with the world's architectural details that every cell is either fully in freespace or fully obstructed. Because graph search algorithms, including A*, have time and space complexity at least exponential in the size of the graph, an optimal plan is rarely found in real time.

Two families of navigation planners are probabilistic. A probabilistic road map (*PRM*) algorithm builds a graph on randomly generated locations and adds edges where it is possible to move directly between them (Kavraki et al. 1996). PRM planners control the size of the graph but depend on clever location sampling and fortuitous selection to find good routes. Another approach, Rapidly-exploring Random Trees (*RRT*), iteratively extends a graph from an initial node (the starting point) and then tries to connect randomly-selected locations to the nearest node in the existing graph (Lavalle 1998). RRT has a Voronoi bias that prefers extensions to unrepresented portions of the world. Both PRMs and RRT return a sequence of specific waypoints and have versions that are asymptotically complete, that is, complete in the limit of the number of nodes in their graphs (Karaman and Frazzoli 2011). Unlike our work, they do not exploit navigation experience or continuous portions of freespace.

Even with a tractable search algorithm for an accurate and adequately detailed map of the world, plans often fail. The robot still must be able to move through freespace from one waypoint to the next. If it has adequate actions in its repertoire and its actuators are precise, the plan will succeed. Too often, however, this is not the case. The model may be flawed or a distant obstacle may have gone unnoticed. The model may be incomplete because the true vista is beyond the sensor's limit or because the robot has no experience in some part of the world. If the robot's actuators err, it may not be able to reach, or even sense, its next waypoint.

To recover from plan failure, a controller can replan, but that is often computationally costly. Alternatively, it can attempt *plan repair*, that is, provide a subplan intended to address that problem (e.g., plan to move around an object left in the middle of a hallway). One way to repair a plan is with a *reactive planner*, which triggers in a particular situation and then provides one-step responses based on some strategy until it alleviates that situation and the controller resumes execution of the original plan. Neither replanning nor plan repair is guaranteed to alleviate the underlying problem.

Another approach to plan failure is *hierarchical planning*. It formulates a sequence of less precise plan steps (e.g., "go to the airport," "take the flight") and defers more detailed steps until execution time (Kaelbling and Lozano-Pérez 2011). Such plans are likely to be more robust because they leave the controller with multiple possible actions from which it can choose once it has completed the immediately preceding plan step. This transformation from general to more specific at execution time is called *operationalization*. For example, there may be multiple options to reach the airport, but the best choice may be clear only when it is time to go there. During execution, the controller only operationalizes a segment when it is about to execute it, and then selects actions to achieve it.

A recent Bayesian model of general hierarchical discovery proposed a system that plans online and learns concepts incrementally for use by its planner (Tomov et al. 2020). This approach could theoretically discover the concepts that underlie SemaFORR's model, but it was tested only on tiny, artificial worlds. SemaFORR too plans hierarchically online and increments its model when it achieves a goal, but it only instantiates input concepts and uses reactive planning to speed its adaptation to specific worlds.

A Freespace Model

SemaFORR's world model is a collection of *spatial affordances*, features that facilitate movement rather than obstruct it. Its model summarizes its experience, based only on the discrete decision points recorded in its path history. Nonetheless, all these affordances represent continuous lengths or areas in freespace. Further details on the algorithms sketched here are available in (Epstein et al. 2015; Epstein and Korpan 2019). We use an artificial world, Home, to illustrate the ideas in this section.

A *region* is a circle in freespace. Each time the robot is stationary, the controller enters a sense-decide-act loop, and the range finder reports the distances to the nearest obstructions. In Figure 2(a), the region defined by the decision point at the arrow's base is the circle whose center is the robot's location and whose radius is the minimum distance reported

by V. From some orientations, this local information will be an overestimate of freespace that changes when the robot turns in place. Accumulated contradictory or overlapping regions are resolved when a task terminates. A region records its radius and decision point, plus any additional range data detected if the robot assumed other poses in that region.

An *exit* for a region represents access to freespace, a point where the robot's path once crossed the region's perimeter. A *door* is an arc on the region's perimeter, a continuous generalization of finitely many, relatively close exits between its endpoints. A door assumes that its infinitely many, as-yet-unexperienced locations will also serve to transport the robot into or out of the region. Figure 2(b) is an example of learned regions with exits and doors (drawn for clarity as secants to their respective arcs). Although regions use freespace to approximate what appear to be rooms in the figure, they record only sensor data, not walls.

A *trail* is a refined version of the robot's path during a task, whether or not it reached its target. At the end of a task, the learning algorithm revises the robot's path heuristically, to smooth it and eliminate digressions. The remaining (usually far fewer) decision points are *trail markers*; they suffice to define the trail and preserve its views. As in Figure 2(c), the sequence of line segments defined by consecutive trail markers is typically more direct than the original (dotted) path, but the trail is not expected to be optimal.

A *conveyor* is a small square in freespace that frequently supports trails. To learn conveyors, SemaFORR superimposes a $2 \times 2m$ grid on the world's footprint, and tallies how often any trail passes through each cell. Conveyors are high-count cells that are presumed to facilitate target-independent navigation, but are not used in planning. Figure 2(d) is an example of learned trails and conveyors, with higher-count conveyors shaded more darkly.

The model's *skeleton* is a graph that represents how regions are connected. Each node there represents a region; an edge between two nodes indicates that the robot took some path directly from one region into the other, without passing through any intervening region. Each such path segment is reformulated as a *subtrail*, a sequence of trail markers derived from it. The length and trail markers of the shortest such subtrail become the label for the edge in the skeleton between those two regions. Figure 2(e) superimposes a skeleton SemaFORR learned for Home, where edges indicate only the existence of a subtrail, not its trail markers. With perfect knowledge, a region of degree one in a skeleton would be a dead-end.

Planning

Because freespace is about where one could be, it naturally supports planning. In practice we have found that A* paths in fine grids for real worlds typically hug walls and corners tightly. As a result, when a small sensor or actuator error occurs during the construction of the occupancy grid or during navigation in the world itself, such plans often fail. Instead of waypoints, SemaFORR represents a plan as a sequence of *steps* (waypoints or regions) in freespace.

SemaFORR plans in the skeleton, which is built from continuous travel to a sequence of targets and is therefore connected. A step in a plan that requires movement from region A to region B is necessarily feasible, because some path once went from a location in A to a location in B to establish the graph's edge between them. Thus, these plans are correct up to actuator and sensor error. If the robot's initial position and target lie in regions, any complete graph search algorithm will make the planner complete.

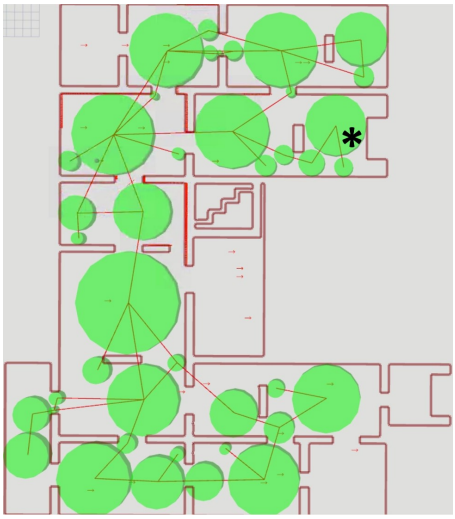


Figure 3: Skeleton for M5

Planning in a freespace graph offers two advantages: speed and flexibility. Planning is faster because a freespace graph is typically several orders of magnitude smaller than an occupancy grid for the same world. Consider, for example, M5, the $54 \times 62m$ fifth floor of MoMA, New York's Museum of Modern Art. Planning on a $1 \times 1m$ occupancy grid there requires a graph on hundreds of thousands of cells, versus the 21 nodes in Figure 3. Freespace plans are also more robust because movement in freespace offers so many possible ways to travel, for example, from one region to another.

SemaFORR's planner, SKELETONP, finds a shortest path in the skeleton with graph search. If the robot and the target each lie in (distinct) regions, SKELETONP generates a hierarchical plan, from the robot's initial pose R to the target location at T, as a sequence of regions in the skeleton. If they do not, SKELETONP identifies *surrogate* regions, $\widehat{R}$ and $\widehat{T}$ for R and T, respectively. To find a surrogate for a point p, SKELETONP takes the closest region to p whose recorded range data senses p. If there is none, then it scores each region with a weighted combination of its center's Euclidean distance to p and its degree in the skeleton, and selects a region with maximum score. Once the surrogates are selected, the plan is $\widehat{R} \rightarrow \ldots \rightarrow \widehat{T}$. Because we have no admissible heuristic h for our planner, we use Dijkstra's algorithm, which has higher complexity bounds than A* but is fast enough because the skeleton is small (Dijkstra 1959). How to reach the surrogates and how to operationalize a plan opportunistically are described in the next section.

SemaFORR is designed to learn its model online, as it pursues a sequence of tasks, but that leaves it at a consider-

Table 1: SemaFORR's Advisors and their rationales

Tier 1, in order of priority	
VICTORY	Go toward an unobstructed target
AVOIDOBSTACLES	Do not go within ε of an obstacle
NOTOPPOSITE	Do not return to last orientation
ENFORCER	Operationalize and execute plan
OUT	Leave a repeatedly confined area
Tier 2 planners	
SKELETONP	Find region sequence to target
Tier 3 heuristics	
Based on commonsense reasoning	
BIGSTEP	Take a long step
ELBOWROOM	Get far away from obstacles
NOVELTY	Go to unvisited locations
GOAROUND	Turn away from nearby obstacles
GREEDY	Go close to target
Based on the spatial model	
CONVEY	Go to frequent, distant conveyors
ENTER	Go into the target's region
EXIT	Go far from non-target regions
LEASTANGLE	Leave this region toward target
TRAILER	Use subtrail toward target
UNLIKELY	Avoid dead-ends in the skeleton

able disadvantage on its first task, when it has no model, and on many early tasks, when its model is quite limited. We therefore employ an exploration algorithm whose original purpose was to enhance the model with other affordances. Here, however, we use it only as an active learner to establish an initial model before the robot began its tasks. The exploration algorithm assumes that the world is intended to facilitate travel and that freespace cues the navigator with glimpses of long extents (e.g., down a hallway). Given a time limit, exploration recognizes and pursues selected long ($\geq 7m$) rays. (This work is currently under review.)

Integration by Metareasoning

SemaFORR is based on *FORR* (FOr the Right Reasons), a cognitive architecture for learning and problem solving (Epstein 1994). A FORR-based system makes decisions with a set of rationales, each implemented by an *Advisor*, a procedure that represents its behavior. A list of SemaFORR's Advisors appears in Table 1.

SemaFORR integrates the spatial model and all its Advisors in tiers that capture three reasoning mechanisms: reactivity, planning, and heuristic guidance. At each decision point, given its full experiment history, action repertoire, and current spatial model, SemaFORR cycles through its tiers of Advisors to select its next action. In a pose, the robot is said to sense a point if it lies within the area covered by its range finder, and to sense a region only if it can sense its center.

The reactive Advisors in *tier 1* take priority, because reactivity saves cognitive effort and prevents foolish mistakes and wasteful behavior. The first three provide initial navigational common sense, in the following order. If the robot senses its target, VICTORY orients the robot toward the target or moves directly forward toward it; this becomes the decision and the cycle ends. Otherwise, AVOIDOBSTACLES eliminates from consideration all forward moves that would go beyond the distance it senses directly in front of it, and NOTOPPOSITE eliminates consecutive repetitive rotations in place. The remaining actions are *viable*. If, after any tier-1 Advisor executes, only one viable action remains, it becomes the decision and the cycle ends. Otherwise, the viable actions are forwarded to tier 2.

Because no plan should ignore reliable reactive behaviors, the two plan-related reactive Advisors that perform metareasoning are last in tier 1. ENFORCER tries to follow the current hierarchical plan. It operationalizes one step at a time, and then evaluates the remaining possible actions in the context of that step. Operationalization depends on a step's type.

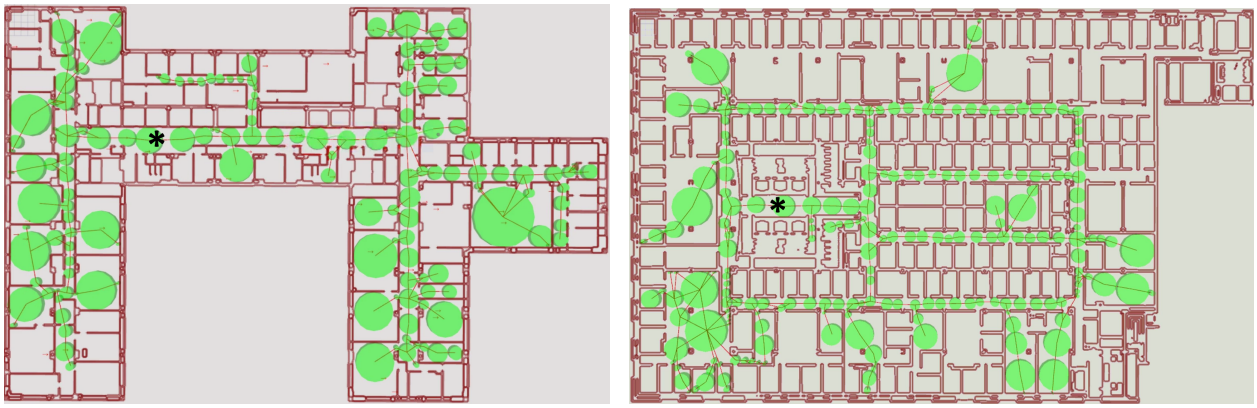
To operationalize a waypoint w, ENFORCER *approaches* it, that is, it selects a move that should bring the robot closer to w or, based on *one-step lookahead*, a turn that should do so. This lookahead does not actually turn the robot; it merely transposes the range data to consider it from the incomplete ($< 220°$) perspective it can project after the turn, and applies its rationale from that perspective. If there is any viable such action, ENFORCER selects as the decision the one that it predicts will bring it closest to w, and the decision cycle ends.

To operationalize the next region in a plan whose next steps are a sequence of regions $A \rightarrow B \rightarrow C$, ENFORCER considers the current decision point. If the robot is not in A, ENFORCER makes A's center the first step in the plan. Otherwise, for a robot in A, there are three possibilities:

- If the robot can sense C, ENFORCER ignores B and replaces C with C's center. This is what psychologists call a "novel shortcut," where the navigator has only ever traversed two legs of a triangle but now takes the third, a proclivity well-documented in many animals.
- If the robot can sense B but not C, ENFORCER replaces B with B's center. Although the skeleton stores a guaranteed path from A to B, it is only a sequence of subtrail markers. If the robot can sense B's center, there is a clear and likely more direct move into B. SKELETONP may not have been able to predict it, but the subtrail would now constitute a less efficient detour given the robot's current pose.
- Otherwise, ENFORCER replaces B with the trail markers recorded in the skeleton for the subtrail from A to B.

If ENFORCER alters the plan, the cycle ends. Otherwise, if it selects no action, it passes the viable actions to its reactive planner OUT.

OUT addresses repeated confinement in a small area (e.g., a room or the space around a narrow corner). In a grid superimposed on the footprint of the world, SemaFORR tallies how often each cell has been sensed as freespace. When at least 75% of the cells in the last 10 V's have been counted at least 4 times, and the current decision point identifies no new freespace, OUT triggers. Its strategy is first to have the robot turn to cover a full $360°$. If this identifies any new freespace, OUT halts, tier 1 ends, and control passes to tier 2. Otherwise, as it examines the history of the full experiment backwards, it collects decision points until it finds one, p, whose location it cannot sense from its current location. OUT refines that subpath, from the robot's current location



(a) H10 (b) G5

Figure 4: Skeletons after 30 minutes of exploration that began from *

backward to p, into a subtrail. To repair the current plan, OUT places that subtrail's markers as a sequence of waypoints at its front.

In *tier 2*, if planning has never been attempted for this task and SKELETONP can formulate a plan, it sends that plan to ENFORCER, and the cycle ends. A decision is relegated to tier 3 only if the plan has reached $\widehat{T}$ but not T, or when no action would satisfactorily operationalize the next step in the current plan. In that case, control passes to tier 3 along with the viable actions.

Tier-3 Advisors are single-focus, deliberately simplistic heuristic procedures that express action preferences. Five of the 11 tier-3 Advisors in Table 1 express navigational common sense; the others exploit the spatial model. Whenever a tier-3 Advisor has appropriate knowledge (e.g., a history of its previous decision points for NOVELTY or a relevant affordance for a spatial heuristic), it comments on every viable action with some *strength* $s \in [0, 10]$, from strong opposition to strong preference. Comments are on moves, and on turns with one-step lookahead if θ is within ϵ of the location the action addresses. Given the comments of all applicable tier-3 Advisors on the still-viable actions, SemaFORR's decision is the one with maximum total comment strength. Although these comments typically conflict, tier 3's power lies in the synergy among the rationales that underlie them. For example, a long forward move is likely to be better if it is also in the direction of the target and avoids a dead-end. Table 2 provides an example of a tier-3 decision.

Table 2: Comment strengths of Advisors during a typical tier-3 decision. The top 4 viable actions were two moves (m_1 and m_2), pause p, and a small left turn l. Each was supported with maximum strength by some Advisor. Despite some enthusiasm for m_2, l was the decisions.

Advisor	m_1	m_2	p	l
BIGSTEP	5.0	10.0	0.0	5.0
ELBOWROOM	4.1	10.0	0.0	2.9
NOVELTY	4.7	10.0	1.8	5.9
GOAROUND	0.0	0.0	0.0	10.0
GREEDY	5.0	0.0	10.0	5.2
TRAILER	10.0	0.0	5.1	6.9
Total	28.8	30.0	16.9	35.8

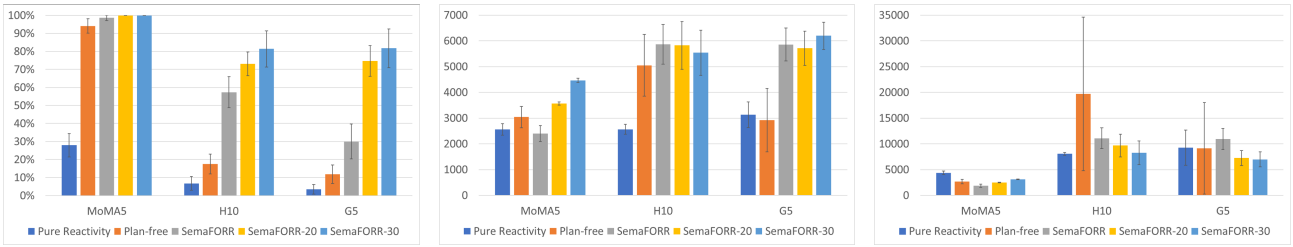
Experimental Design and Results

In an extensive suite of experiments, we tested SemaFORR in three challenging worlds. M5 was designed to attract, but not force, visitors along some path that allows them to view all the temporary exhibits there. H10 is an $89 \times 58m$ floor in a repeatedly renovated building. G5 is a $100 \times 70m$ floor known for its ability to perplex human navigators, despite color-coded walls and art introduced as landmarks by thoughtful employees. The other floors in the same buildings are both different and easier.

For each world, we randomly generated 5 different sets of 40 locations in freespace. A *run* began a target set with the robot at the real-world's entrance (the stars in Figures 3 and 4), and attempted to reach each target in that set within a *decision limit* (maximum number of actions selected). For M5, the robot began facing left with a 500-decision limit for each task. For H10 and G5, it began facing right, with a 750-decision limit. Travel to a target after the first one began in the location where the previous search ended.

To examine the interplay among reactivity, planning, heuristics, and the spatial model, we also tested two ablated versions. The first was a purely reactive baseline whose only Advisors were VICTORY, AVOIDOBSTACLES, and NOTOPPOSITE in tier 1. The other ablated version learned and used the spatial model but did not plan; it had the same three tier-1 Advisors and all the tier-3 Advisors. To demonstrate the power of the spatial model, we also tested SemaFORR preceded by 20 or 30 minutes of exploration.

All experiments simulated Fetch Robotics' Freight, an industry-ready robot whose laser range scanner reports 15 times per second the distance to the nearest obstruction within $25m$ of the robot, in 660 directions along a $220°$ arc. The robot's action repertoire was 6 forward moves (with motor activation to move 0.1, 0.2, 0.4, 0.8, 1.6, or $3.2m$), 12



(a) Success rate (b) Distance (meters) (c) Time (seconds)

Figure 5: Performance on three metrics, grouped by world. Within a world, from left to right, each histogram reports on the purely reactive baseline, the version without planning, SemaFORR, and SemaFORR preceded by 20 minutes and then 30 minutes of exploration.

rotations (with motor activation to turn clockwise or counterclockwise 5, 15, 30, 45, 60 or 90 degrees), and a pause.

MengeROS managed the simulation (Aroor, Epstein, and Korpan 2017). It deliberately introduced small, random errors into both the sensor data and into action execution as changes to motor time. MengeROS calculates V from an input geometric map to which SemaFORR has no access. To avoid collisions due to inaccurate sensor data, the maximum permissible forward move was no more than the minimum length within $\theta \pm 30°$. To avoid small, distant obstructions that could go undetected, the ability to sense a point requires confirmation within $20m$ and support within a multi-ray area. Precision ϵ with respect to arrival at a point in all experiments was $1m$ for targets, but $0.5m$ for subtrail markers, and $0.75m$ for regions' centers, both of which rely on V and so require greater care. All experiments ran on a Dell Precision Tower 7910 with 16 GB memory and an Intel Xeon(R) CPU ES-26-30 v3 @ 2.40GHz x 16, that ran ROS Indigo in Ubuntu 14.04.

The principal performance metric is *success rate*, the fraction of targets reached within the decision limit. We also report travel distance in meters and elapsed wall-clock time (model construction, planning, decision making, and exploration) in seconds. Data is reported over 10 runs, 2 for each set. Cited differences are statistically significant ($p = 0.05$).

SemaFORR performs in real time. For example, after 20 minutes of exploration, decisions averaged 0.402 seconds in M5, 0.216 in H10, and 0.238 in G5. Across all 3 worlds, model updates after a task averaged 19.5 seconds and planning in the skeleton only 0.17 seconds per task.

In M5, the purely reactive version struggled (27.9% success rate) but the plan-free version was surprisingly successful (94.1%), which we attribute to the openness of the space. In the two larger worlds, however, neither ablation did well. Planning improved the success rate in M5, and reduced time and distance. In H10, planning tripled the success rate, with little change in distance, and time was nearly halved, which suggests more efficient travel. In G5, planning doubled the success rate but also nearly doubled distance while time remained steady. G5 is larger, with many hallways that allow more long moves. As a result, the robot often traveled further to reach its targets, and thus took more time to execute under the same decision limit.

The spatial model supports planning. Inspection indicates that M5's few failures came on early tasks. Twenty minutes of exploration was enough to address most of the long rays the algorithm considered worthy of investigation, and produced a skeleton that supported success on every task. Thirty minutes of exploration maintained 100% success but incurred additional time and distance. In H10, 20 minutes of exploration significantly improved the success rate, and 30 minutes did so again, while time and distance remained unchanged. In G5, 20 minutes of exploration dramatically improved the success rate and reduced the time, while 30 minutes raised G5's success rate to that of H10.

Ideally, as a navigator learned more about its world, it would reach its targets faster and cover less distance. With SemaFORR, however, successes rise with planning and exploration. As a result, operationalized plans replace tier-3's tentative compromises with larger, more purposeful actions that may increase time and distance but reach more targets within the same decision limits.

Heuristics supported decision making. Table 3 reports on 255,975 tier-3 decisions during 10 runs in M5 with 20 minutes of exploration, and in H10 and G5 with 30 minutes of exploration. The commonsense Advisors almost always commented; the spatial model Advisors required relevant affordances. Only BIGSTEP and CONVEY had their strongest preference selected more often than not. GREEDY was the most frequently outvoted Advisor (41.2%).

Table 3: Frequency of tier-3 Advisors' comments, and how often their comments' strength s supported the chosen action

Tier 3	Freq.	$s = 10$	$s > 5$
BIGSTEP	100.0%	91.6%	94.6%
ELBOWROOM	100.0%	28.8%	72.2%
NOVELTY	99.2%	24.4%	73.6%
GOAROUND	98.0%	14.5%	67.1%
GREEDY	100.0%	18.7%	58.8%
CONVEY	29.0%	72.2%	92.9%
ENTER	3.0%	39.4%	73.2%
EXIT	60.1%	44.2%	88.2%
LEASTANGLE	12.6%	38.6%	71.1%
TRAILER	36.1%	44.8%	84.5%
UNLIKELY	14.5%	19.4%	87.8%

Discussion

SemaFORR demonstrates that a robot equipped only with a range finder can learn to navigate in a large, complex world without a map. Mapping is typically a computationally costly undertaking. To the best of our knowledge, no other controller has tackled this problem in worlds as large and complex as these.

Planning is clearly essential but not every difficulty can be anticipated. When a situation repeatedly arises, that suggests the need for reactive intervention. What motivated OUT, for example, was failure to make progress, particularly in rooms where the robot was on the wrong side of a wall deceptively close to the target. OUT triggered on average during 18.25% of the tasks in H10 and G5 after 30 minutes of exploration.

Metareasoning includes introspective monitoring of reasoning, and actions taken in response to it (Cox and Raja 2011). In SemaFORR, ENFORCER monitors plan execution with the reactive planner OUT to recognize and address lack of progress. ENFORCER also improves on SKELETONP's traditional approach with operationalization that introduces novel shortcuts and alternate paths into regions. While traditional navigation planners attempt to repair or replan in the face of failure, we believe that SemaFORR is the first to improve its plans opportunistically, during execution.

Reliance on randomly selected targets to develop SemaFORR's model is clearly less effective than principled exploration. The exploration-exploitation trade-off is a classic AI issue: whether an agent should apply what it knows and act to solve its current problem, or should act in a way that may ultimately increase its ability to solve this and other problems. This motivated the tier-3 Advisor NOVELTY, which simulated local curiosity but lacked a global perspective. Unless global exploration occurs before the robot's first task in a new world, SemaFORR has no spatial model, cannot plan, and is likely to fail. Even after its first few tasks, the model is likely to be inadequate for many targets.

The purpose of initial exploration is to provide a better world model in which to plan. Exploration offers SemaFORR a global perspective, a skeleton for a new world acquired through active learning with self-determined targets. An initial 30-minute investment in exploration, included in Figure 5(c), reduced task-solving time per target set by 75.5 minutes in H10, and by 95.8 minutes in G5.

An incomplete model of the world, particularly near the target, caused the remaining failures in H10 and G5. If the robot had to rely on heuristics to reach its first plan step $\widehat{R}$ from its current location, or to reach the target from its last plan step $\widehat{T}$, it was because it lacked knowledge. A metric on SemaFORR's spatial model is *coverage*, the fraction of freespace it includes. Table 4 reports on the model's coverage after 40 tasks. Across all our experiments, average coverage after the first task was significantly correlated with the success rate (0.92 for G5, 0.81 for H10, and 0.54 for M5).

Exploration assured perfect success in M5, a relatively open space. Despite the remarkable skeletons in Figure 4, however, Table 4 makes clear that in H10 and G5, with about 75 and 180 rooms, respectively, more exploration is not likely to improve SemaFORR further. What would? Not more tasks, which rely too much on serendipitous targets to extend the model. Not a higher decision limit, which would simply allot more resources to heuristics that have proved 80% adequate. Instead we have identified, and have under development, several promising additional reactive planners and spatial affordances.

Table 4: Freespace, coverage, and exploration time

World	Free(m^2)	0 min.	20 min.	30 min.
M5	1585	80.5%	93.2%	93.7%
H10	2627	28.3%	37.3%	44.0%
G5	4021	15.3%	34.7%	40.7%

We envision several applications for SemaFORR. Architects could apply it to buildings still in design, to predict how they might be navigated or improved. Rescue teams could apply it to damaged buildings, to find alternate routes for personnel. SemaFORR could also accompany another controller to provide information on global connectivity. The constants in the descriptions here are parameterized in our code, to make SemaFORR readily applicable to other platforms. (Grid cells, for example, are $1 \times 1m$ because that comfortably fits our robot.) In earlier work, we developed simple templates that enabled SemaFORR to respond to a restricted list of questions (e.g., "why did you do that?") in clear natural language (Epstein and Korpan 2019). Current work adapts that to the planner described here, so that SemaFORR might express its confidence as well as justify its behavior. Future work includes automatically learning OUT's trigger parameters; the current values were selected to generalize well across different worlds.

SemaFORR's key features are worthy of emphasis. Its affordances inductively generalize continuous space from only discrete measurements. It learns both actively, when it chooses what to explore, and online, at the end of each task. Its hierarchical plans defer operationalization and thus allow the controller to correct for actuator error and for obstructions that previously went unnoticed by its sensors. Its plan execution accepts intervention both to extricate it from difficulty and to detect and exploit novel shortcuts. Its decision limit allows it to augment all but the conveyors in its model, whether or not it reaches a target, so it learns from both success and from failure. It treats historical behavior (subtrails) as a recommendation rather than a requirement.

In two worlds deliberately selected for their difficulty, SemaFORR navigates autonomously to about 80% of its targets. We look forward to its continued improvement. Meanwhile, SemaFORR's metareasoning interleaves hierarchical planning and reactive planning to address and exploit different situations as they arise during target-driven navigation.

Acknowledgments. This work was supported in part by The National Science Foundation under CNS-1625843. The authors also thank Gil Dekel and Matthew Evanusa for their pioneering work on the spatial model, and Anoop Aroor for MengeROS and his support in SemaFORR's development.

References

Aroor, A.; Epstein, S. L.; and Korpan, R. 2017. MengeROS: A crowd simulation tool for autonomous robot navigation. In *2017 AAAI Fall Symposium on Human-Agent Groups*.

Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; and Leonard, J. J. 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics* 32(6):1309–1332.

Cox, M., and Raja, A. 2011. *Metareasoning: An Introduction*. MIT Press. 3–14.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1):269–271.

Durrant-Whyte, H., and Bailey, T. 2006. Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine* 13(2):99–110.

Epstein, S. L., and Korpan, R. 2019. Planning and explanations with a learned spatial model. In Timpf, S.; Schlieder, C.; Kattenbeck, M.; Ludwig, B.; and Stewart, K., eds., *COSIT-2019*, volume 142 of *LIPICS*, 22:1–22:20. Schloss Dagstuhl.

Epstein, S. L.; Aroor, A.; Evanusa, M.; Sklar, E. I.; and Parsons, S. 2015. Learning spatial models for navigation. In Fabrikant, S. I.; Raubal, M.; Bertolotto, M.; Davies, C.; Freundschuh, S. M.; and Bell, S., eds., *COSIT*, volume 9368 of *Lecture Notes in Computer Science*, 403–425. Springer.

Epstein, S. L. 1994. For the right reasons: The FORR architecture for learning in a skill domain. *Cognitive Sforscience* 18(3):479–511.

Forster, C.; Carlone, L.; Dellaert, F.; and Scaramuzza, D. 2017. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Transactions on Robotics* 33(1):1–21.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

IEEE RAS Map Data Representation Working Group. 2015. IEEE Standard for Robot Map Data Representations for Navigation.

Joshi, S.; Schermerhorn, P.; Khardon, R.; and Scheutz, M. 2012. Abstract planning for reactive robots. In *2012 IEEE International Conference on Robotics and Automation*, 4379–4384. IEEE.

Kaelbling, L. P., and Lozano-Pérez, T. 2011. Hierarchical task and motion planning in the now. In *ICRA*, 1470–1477. IEEE.

Karaman, S., and Frazzoli, E. 2011. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research* 30(7):846–894.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4):566–580.

Lavalle, S. 1998. Rapidly-exploring random trees : a new tool for path planning. Technical Report TR98-11, Iowa State.

Lavalle, S. M. 2006. *Planning Algorithms*. Cambridge University Press.

Lu, F., and Milios, E. E. 1997. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems* 18(3):249–275.

Quigley, M.; Conley, K.; Gerkey, B. P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. Y. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

Stachniss, C.; Leonard, J. J.; and Thrun, S. 2016. Simultaneous localization and mapping. In *Springer Handbook of Robotics*. Springer. 1153–1176.

Stein, G. J.; Bradley, C.; Preston, V.; and Roy, N. 2020. Enabling topological planning with monocular vision. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Thrun, S.; Burgard, W.; and Fox, D. 2016. *Probabilistic Robotics: Intelligent Robotics and Autonomous Agents*. MIT Press.

Tomov, M. S.; Yagati, S.; Kumar, A.; Yang, W.; and Gershman, S. J. 2020. Discovery of hierarchical representations for efficient planning. *PLoS Computational Biology* 16(4):e1007594.