Hierarchical Freespace Planning for Navigation in Unfamiliar Worlds

Raj Korpan[1] **and Susan L. Epstein**[1,2]
[1]The Graduate Center of The City University of New York
[2]Hunter College of The City University of New York
rkorpan@gradcenter.cuny.edu, susan.epstein@hunter.cuny.edu

Abstract

Autonomous navigation in a large, complex space requires a spatial model, but the construction of a detailed map is costly. This paper demonstrates how two kinds of exploration support an alternative to metric mapping, one that facilitates robust hierarchical path planning. High-level exploration builds a global spatial model whose connectivity supports an effective, efficient, freespace planner, while low-level, target-driven exploration addresses areas where the global model lacks knowledge. Empirical results demonstrate successful and efficient travel in three challenging worlds.

Introduction

Increasingly, robots deployed in large, complex, indoor spaces (*worlds*) are expected to navigate to specific locations autonomously. Path planning based on a spatial model proves necessary for such navigation. The problem addressed here is autonomous navigation in an unfamiliar world. Traditionally, a metric map would detail obstructions there and a planner would seek to avoid them. Instead of such a map, our planner uses models of the world's *freespace*, the areas that a navigator could occupy, and formulates hierarchical plans. The thesis of this work is that models of connected freespace based only on data from an onboard range finder support effective planning for navigation in an unfamiliar world. The principal contributions of this paper are two novel exploration algorithms, the models they produce, and a hierarchical planner, all focused on freespace. Extensive empirical simulation in challenging real worlds demonstrates the real-time effectiveness of this approach and its support for efficient planning and *robust* travel that is resilient to robotic error and plan failure.

A *target* here is a fixed location in the world. The robot's *task* is to visit a randomly selected sequence of targets without an input world model. A *plan* is a sequence of behaviors that, if executed flawlessly, reaches a target. While a traditional plan identifies a sequence of discrete points to visit, a freespace plan identifies a sequence of portions of freespace to visit. Freespace planning facilitates higher-level plans (e.g., "go to the end of this hallway") and provides considerable flexibility for an agent that operates under uncertainty and in realistic conditions.

Rather than explore exhaustively or wait while an adequate model gradually develops from experience, active learning initially explores an unfamiliar world to build two high-level models. The first model is static; it captures global connectivity as a network of extended stretches of freespace that could support long-distance travel. The second model is dynamic; initially it represents more local and detailed knowledge from the same exploration. Subsequent target-driven experience, however, augments the second model with local detail and may trigger additional, low-level exploration. The planner exploits both models.

While freespace planning proves empirically adequate for many tasks, sensor noise, actuator error, and lack of experience and knowledge challenge this approach. Any plan here is incomplete unless its freespace model includes the target. To reach such targets, we supplement high-level exploration with low-level exploration in the target's vicinity. Moreover, plans whose subgoals are continuous areas, rather than discrete points, allow for execution-time responses that compensate for errors and exploit unanticipated opportunities. The implementation with ROS (Quigley et al. 2009) is for a set of specific, industry-ready robots, but parameterized for other platforms. Our approach generalizes across worlds; the parameters described here were tuned to one world but worked well in others. The next section provides the context of this work. Subsequent sections describe the exploration algorithms, the models, and the hierarchical freespace planner. Finally, we describe and discuss our empirical results.

Related Work

This work addresses the classic AI trade-off between exploration for knowledge and exploitation of what is already known. Traditionally, robot controllers first use *mapping* to discover the precise metric location and position of all obstructions in an environment, and then construct paths to a target in the learned map. Other controllers learn a limited spatial representation and then plan within it. This section describes the traditional approach and then provides context for the alternatives.

SLAM (Simultaneous Localization and Mapping) both *localizes* (determines the robot's current location and orientation in the world) and builds a map at once (Durrant-Whyte and Bailey 2006). Popular SLAM approaches are probabilistic (Stachniss, Thrun, and Leonard 2016) and use

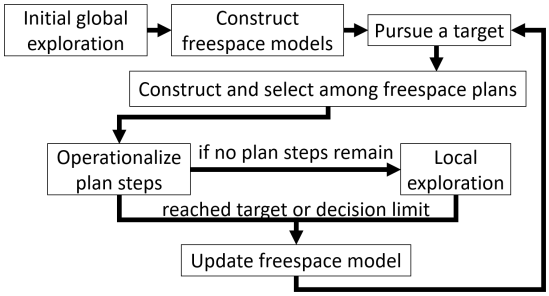


Figure 1: High-level diagram of our approach

a non-convex cost function (Lu and Milios 1997; Cadena et al. 2016). They require extensive parameter tuning, are not robust to outliers, and drive the robot to systematically visit the entire world.

Active SLAM probabilistically selects actions to reduce uncertainty in the map and to localize the robot (Leung, Huang, and Dissanayake 2006). This action selection is often intractable in practice without additional help, such as deep reinforcement learning trained on billions of samples (Wayne et al. 2018). Instead, our approach assumes near-perfect localization. This could be achieved with physical or visual landmarks (Se, Lowe, and Little 2002), visual-inertial odometry (Forster et al. 2017), and WiFi signals (Ocana et al. 2005), all of which perform well without mapping or offline learning from vast quantities of data.

Other approaches address uncertainty by exploration. Frontier-based exploration tracks the boundary between known and unknown space, and plans paths to that boundary to reduce the area of unknown space (Yamauchi 1997). Exploration based on information gain (Whaite and Ferrie 1997) or the next best view (González-Banos and Latombe 2002) greedily selects where to explore. More recently, deep reinforcement learning on office blueprints strategically selected the next area to explore (Zhu et al. 2018). It was limited, however, to worlds of the type on which it had learned (e.g., small offices with fewer than 10 rooms), while our approach requires no preliminary training or categorization. Rather than focus on reduced uncertainty in the entire map, we prioritize exploration for high-level connectivity.

Given some experience in a new world, a robot controller constructs a spatial model in which it can reason about how to get to a target. For example, an *occupancy grid* superimposes a uniform grid on the world's footprint and records which cells are obstructed. Probabilistic road maps (Kavraki et al. 1996) and rapidly-exploring random trees (*RRT*) (Lavalle 1998) randomly sample freespace to construct plans but depended on fortuitous, careful location sampling. Both sampling-based approaches and occupancy grids, however, require a map of the entire world, and result in planned sequences of discrete locations that could come uncomfortably close to obstacles. Other work combined frontier-based global exploration, next-best-view-based local exploration, and RRT-based planning (Selin et al. 2019). Although similar to our approach, the goal of that work was efficient mapping of a 3D world, whereas our work uses limited exploration to facilitate robust navigation.

Instead of planning in a graph of discrete locations, our robot controller, *SemaFORR*, represents freespace with a *topological map*, a weighted graph where a node represents a continuous area and an edge label estimates the distance between its endpoints. Other topological models have sought to provide a view of freespace connectivity. Some represented learned topology in first-order logic (Joshi et al. 2012) or as polygons induced with a monocular camera (Stein et al. 2020). Another approach learned a hierarchical representation from a map (Tomov et al. 2020), but lacked SemaFORR's active learning and task connectivity.

SemaFORR produces hierarchical plans from its world model. When an inaccurate model, actuator error, or noisy or incomplete sensor data results in plan failure, replanning and plan repair may be inadequate or expensive. Instead, *hierarchical planning* postpones action selection until a step is executed, and thereby leaves multiple ways to achieve it. To execute a step, the controller *operationalizes* it, that is, replaces it with one or more detailed actions (Kaelbling and Lozano-Pérez 2011). This delay allows the controller to recover from unpredictable sensor and actuator errors and capitalize on unanticipated opportunities.

The work that addresses a task most similar to our own navigates in campus floor plans (Stein, Bradley, and Roy 2018). It learned to encourage movement along hallways, but relied on an occupancy grid and chose actions based on lowest estimated cost. It was trained on thousands of targets, orders of magnitude more than SemaFORR requires, and gauged its performance only against a naïve planner that assumed all space was free until sensed otherwise.

High-level Exploration

In the following, the robot's *pose* is $\langle x, y, \theta \rangle$, where $\langle x, y \rangle$ is its location in a two-dimensional space and θ is its *orientation* with respect to some fixed reference frame. A *decision point* δ is the robot's pose and its *view* V, the set of rays from the robot's range finder that extend to the nearest obstruction in $|V|$ directions. This work simulates a commonly deployed 15GHz onboard laser range finder with a $25m$ range. It casts 660 rays that report V every $1/3^\circ$ across a 220° field of view. V is *local* (within the sensor's range) and *partial* (not a full 360° view). At any δ, SemaFORR chooses an action from the robot's deliberately small *action repertoire*: 6 forward moves (0.1, 0.2, 0.4, 0.8, 1.6, or 3.2*m*) and 12 rotations (5, 15, 30, 45, 60, or 90° clockwise and counterclockwise). Figure 1 shows a high-level diagram of our approach, which begins with global exploration and then pursues targets one at a time with its freespace plans.

Sensor and actuator limitations and errors, along with real-world spatial irregularities (e.g., indentations and protrusions in walls), make freespace models incomplete and imprecise. Moreover, algorithms that represent space continuously soon become intractable. SemaFORR discretizes the robot's body to a point on a coordinate plane, and its movement as a sequence of points in two-dimensional space.

HLE, SemaFORR's high-level exploration algorithm, assumes that the world intentionally facilitates travel and that it cues a navigator with long rays that might be useful to reach

$e(c_L)$ = mean $e(r) \forall r \in LeftFocus$
$e(c_R)$ = mean $e(r) \forall r \in RightFocus$
$r_1 \sim r_2$ iff distance $< 1m$, overlap $\geq (l(r_1) + l(r_2))/6$
Requirements for HLE candidacy
length to width ratio > 1
clear $c \equiv$ all of $s(c), m(c), e(c)$ are unobstructed and at most 1 has a passage number
c not similar to any $c' \in C \cup E$
not a large room $\equiv LR(c) = false$ based on mean, max, min, median, and σ of $l(r), \forall r \in V$
Requirements for LLE candidacy
$possible \equiv r \in (C \cup V^* \cup Q$ from path to $T)$
$l(r) \geq 2m$
r comes within d meters of T
$e(r) > .75m \; \forall \; r \in possible$

Table 1: Computations for exploration

distant locations. HLE's exploration is a preliminary, time-limited search for *passages*, long, relatively narrow extents of connected freespace. In this one-time active learning, the robot selects its own goals ("What's down that hall?") in a search for global connectivity. Essentially, HLE maintains a prioritized list of δ's that indicate potential passages and explores them one at a time.

HLE initializes its *passage grid* as freespace, but can later relabel cells as obstructed or with a passage number. HLE defines a long ray r as a line segment with length $l(r) \geq d$, from its start $s(r) = \langle x, y \rangle$ where it was sensed to its endpoint $e(r)$. The minimum length d is a parameter that can be tuned based on the size of the environment and the robot sensor's range. (Here, $d = 7m$.) To allow for sensor error and spatial irregularities, at each δ HLE forms representative cues c_L and c_R from narrow bundles of 41 rays (***LeftFocus*** and ***RightFocus***) to the robot's immediate left $(\theta - 90°)$ and right $(\theta + 90°)$. A cue starts at the robot's location and ends at the average endpoint of the rays in its bundle.

HLE maintains a list C of *candidates*, dissimilar cues likely to provide novel, useful information. C prioritizes cues where $l(c) \geq 2d$ over those where $d \leq l(c) < 2d$. To facilitate the robot's return for later investigation of a new candidate, HLE places it first in the section of C determined by the cue's length $l(c)$. E records explored candidates, whether or not their search generated a passage.

To become a candidate, a cue must satisfy all the conditions in Table 1. First, it must suggest an area whose length is greater than its estimated width. HLE estimates the freespace width of two wider bundles of 136 rays (***LeftOpen*** and ***RightOpen***) at the periphery of V. A cue c must be *clear*, that is, its start $s(c)$, midpoint $m(c)$, and endpoint $e(c)$ are unobstructed in the passage grid and at most one of them has a passage number. A cue must also differ from all candidates in C and E, as gauged by their overlap (Allen 1983). Finally, LR, a boolean classifier for large rooms, must return false on c. LR was developed offline from thousands of views in a world not used here. When those views were clustered with 2-means, one cluster had the natural label "large room." We learned a decision tree on the data labeled by cluster, and used the tree's top two rules to create LR.

Algorithm 1: Pseudocode for HLE exploration

Input: pose $\langle x, y, \theta \rangle$, current candidates C, explored candidates E, time τ, decision limit b
If $C = \emptyset$, then find first candidate
while *time τ remains and $C \neq \emptyset$* **do**
  Select a candidate $c = (s(c), e(c))$
  if *c is still a valid candidate* **then**
    Assign passage number to c
    Record c as explored in E
    Pursue(*c, pose, time τ,decision limit b*)
end
Build skeleton and highway graph

Pursue(c, pose, time τ, decision limit b)
$path \leftarrow$ path in history from $s(c)$ to $\langle x, y \rangle$
$trail \leftarrow$ refined version of $path$
Execute $reverse(trail)$ to reach $s(c)$
while *termination conditions not met* **do**
  Rotate toward $e(c)$ as necessary
  Move forward
  Observe long rays and screen for new candidates
  Update cell values in passage grid
end

Algorithm 1 is pseudocode for HLE. In the robot's initial pose in an experiment, HLE rotates the robot in place to find valid candidates. Then HLE takes candidates from C until it finds a c that is still clear in the passage grid and dissimilar from those in E. HLE assigns c a passage number $n(c)$, adds c to E, and begins to *pursue* it, that is, it moves the robot to $s(c)$, rotates it to face the endpoint $e(c)$, and moves the robot in uniform $0.8m$ steps toward $e(c)$ with at most b decisions. At each δ during pursuit, HLE saves new valid candidates, updates $e(c)$ if it can sense farther, and assigns the label $n(c)$ to the grid cell that contains δ if it is not already labeled.

HLE begins with the robot at the first candidate's start, but any other candidate c began at an earlier δ. To reach $s(c)$, HLE constructs a *trail*, a refined version of the path that eliminates extraneous steps, such as loops and detours, on the path it took from $s(c)$ to the robot's current location, and follows that trail backward. The trail-learning algorithm, in a single pass, retraces the decision points along a path backward from its endpoint to its start. Each time it can sense a decision point closer to the start, it eliminates any intervening ones. Those remaining, including the start and endpoint, are collected as trail markers. Because exploration forms a continuous path, the trail exists and is correct (reaches its endpoint if executed flawlessly).

Real-world candidate pursuit often encounters spatial irregularities in a passage. If the robot comes too close to an obstruction, HLE has it turn away from it, take a small step forward, and then continue toward $e(c)$. If a passage curves slightly or contains a protrusion, HLE centers $e(c)$ in the freespace directly in front of the robot. As a result, HLE may detect diagonal passages.

There are several termination conditions for candidate

pursuit. HLE stops pursuit of c if it makes more than b decisions to explore it, has reached the end of the passage (i.e., is within $0.5m$ of $e(c)$ or has only $0.1m$ of freespace directly before it) or has just made so hard a turn that it is considered to be in a different passage (i.e., its current orientation differs by more than $45°$ from its average orientation in the current passage thus far). The robot could also encounter an area too wide to be a passage, that is, more like a large room. Once the robot has moved at least half c's estimated length, pursuit of c also stops when HLE estimates the passage's length to be less than 1.5 times its width. Estimated length is how far HLE has pursued c plus how far the robot can sense along its current θ. Estimated width is the sum of the average length of the representative cues c_L and c_R at each δ in the current passage thus far. These termination conditions may produce an explored passage that is shorter than d in length.

Real-world passage walls rarely align neatly with grid-cell borders, so HLE uses occupancy mapping (Moravec and Elfes 1985) during pursuit to label some cells in the passage grid. Within *LeftFocus* and *RightFocus*, HLE relabels a free cell no more than $2m$ from the robot with the current passage number. It also relabels any obstructed cells there within $4m$ of the robot. These values are based on the typical width of hallways in built, indoor worlds. Figure 2(a) shows an example of a passage grid after HLE reached $\tau = 30$ minutes.

Low-level Exploration

LLE, SemaFORR's low-level, goal-driven exploration algorithm, searches for rays that may lead it to a target T. LLE triggers when SemaFORR has no plan or it has completed its plan but did not reach T, indications that the current model offers no further guidance to T. Like HLE, LLE assembles and explores candidates. LLE draws its candidate rays from those that remained when HLE terminated, from Q (the Vs at each δ in the path so far to the current target), and from the rays V^* saved in SemaFORR's model (detailed in the next section). Because LLE is target-driven, the conditions for candidacy are with respect to T, as shown in Table 1. LLE orders its candidates by their distance to T and stops search when it senses T.

Algorithm 2 is pseudocode for LLE. To explore candidate c, LLE constructs a plan to the start of c and then a sequence of 20 evenly spaced *waypoints* (locations) along it to keep them at most $1.25m$ apart based on our robot's maximum sensor range of $25m$. If execution reaches $e(c)$ or the robot *loses track* of c (cannot sense at least one of its next two waypoints on three consecutive decision cycles), LLE discards that candidate and considers the next one. While it is active, LLE continues to screen all rays from each δ for qualified new candidates.

Once no candidates remain, LLE expands its search with an *inclusion grid* that indicates which cells are covered by SemaFORR's learned spatial model. In increments of $1m$, LLE bins, by their distance to T, all rays in V with endpoints marked as unincluded in the grid. It selects one ray at random from the bin closest to T, discards all the others, explores it as if it were a candidate, and updates inclusion grid cells at each δ to avoid subsequent repetitive search. Meanwhile, LLE watches for valid candidates. If one arises, LLE abandons ray exploration and pursues its new candidate instead. Otherwise, after it explores a ray, the robot should be in a new location, from which ray exploration begins again. Finally, when all current rays are marked as covered, LLE formulates a plan to the included cell closest to T, where it may then sense new uncovered rays.

Algorithm 2: Pseudocode for LLE exploration

Input: current pose $\langle x, y, \theta \rangle$, view V, and target T, remaining HLE candidates C, rays V^* stored in the spatial model, views Q during path to T, distance threshold d

```
possible ← C ∪ V* ∪ Q
ValidRays ← candidates(possible, T, d)
Sort ValidRays by distance to T
Initialize inclusion grid
while LLE is active do
    if ValidRays ≠ ∅ then
        Select candidate c from ValidRays
        Explore(c, ValidRays, True)
    else if ValidRays = ∅ then
        Update inclusion grid
        ClosestBin ← ClosestRays(V,T)
        if ClosestBin ≠ ∅ then
            Select random r ∈ ClosestBin
            Explore(r, ValidRays, False)
        else
            Generate and follow plan to cell in
              inclusion grid with minimum dist. to T
end
```

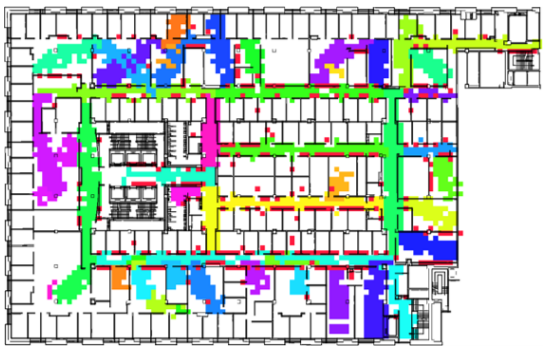
$Explore(c, ValidRays, IsCandidate)$

```
Generate waypoints for c
while able to sense waypoints do
    Visit next remaining waypoint
    Append valid candidates from δ to ValidRays
    if ¬IsCandidate and ValidRays ≠ ∅ then
        return
end
```

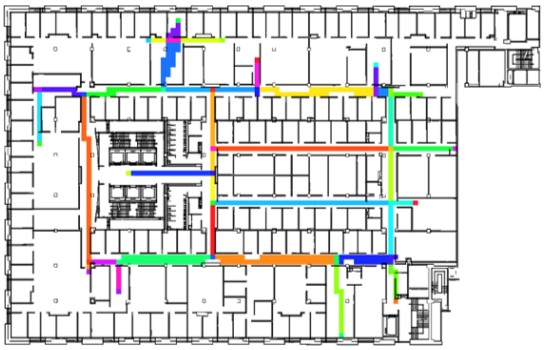
Models of Freespace

After HLE exhausts its candidates or its time limit, SemaFORR builds two models of the freespace in its environment. One is a highway graph, a static initial global model of long extents in freespace. The other is a skeleton, a dynamic model that delineates connectivity among smaller areas of freespace and changes during subsequent travel to targets. Both are described below, with an example in Figure 2.

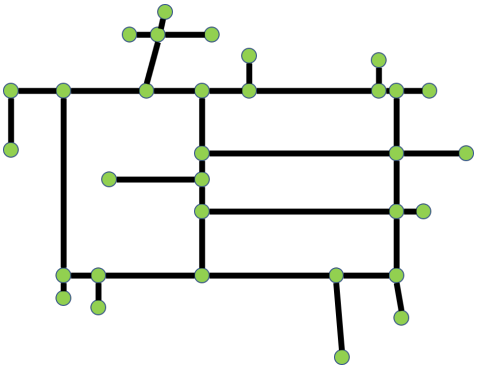
To detect global connectivity, SemaFORR begins with an unlabeled *freespace grid* and revisits the decision points along the continuous path P the robot took under HLE. As SemaFORR moves along the line segment between each pair of consecutive decision points in P, it labels as freespace each grid cell through which it passes. To smooth this record of visited freespace, SemaFORR also labels as freespace any unlabeled cell when at least three of the four cells that share an edge with it are labeled as freespace.



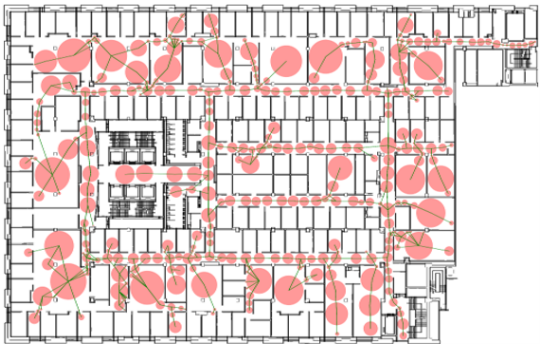
(a) Distinctly colored passages in the final passage grid



(b) Distinctly colored highways in the freespace grid



(c) Highway graph



(d) Skeleton after 40 targets

Figure 2: Freespace models created for G5 after 30 minutes of exploration. This space measures $110 \times 70m$.

Only a contiguous horizontal or vertical freespace extent at least d long in the freespace grid becomes a *highway*. (This may introduce discontinuities.) Grid cells in the (typically small) areas where a vertical and a horizontal highway overlap form an *intersection*. A highway bounded by only one intersection is a *spur*; its other endpoint is relabeled as an intersection. Spreading activation smooths each highway and intersection and labels them uniquely (e.g., Figure 2(b)).

From the freespace grid, SemaFORR then derives a *highway graph*, a connected, planar graph that represents intersections as nodes labeled with their centroids and represents highways as edges that connect their endpoints (e.g., Figure 2(c)). An edge's label is a trail learned from travel along that highway and its cost, the Euclidean distance between the centroids of its endpoints. By construction, one endpoint of a spur has degree one, that is, is a dead-end in the highway graph. If processing has disconnected the highway graph, SemaFORR retains only the connected component with the most nodes. For example, the horizontal passage at the top right in Figure 2(a) went uncovered in Figure 2(b) because its vertical connection to the freespace grid was too short. (The second freespace model, however, recovers it.)

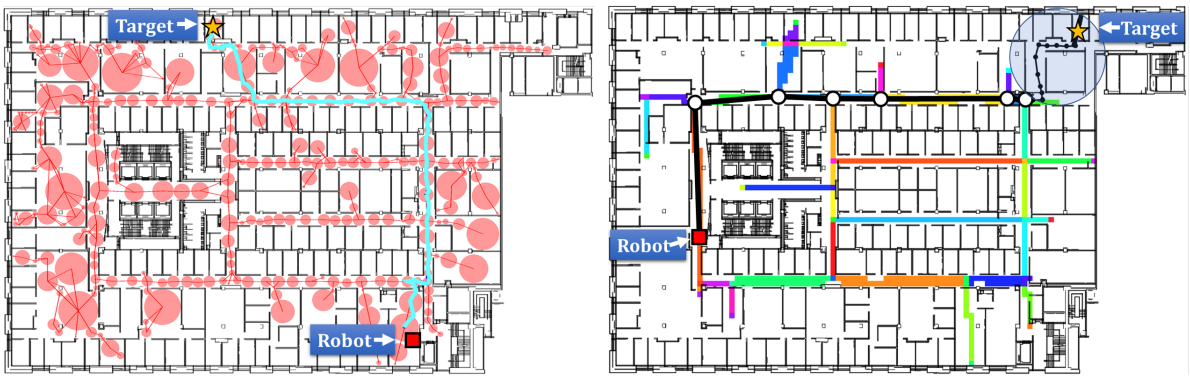
The building blocks for SemaFORR's more detailed spatial model are regions. At decision point $\delta = \langle x, y, \theta \rangle$ with view V, SemaFORR captures the local freespace around the robot as a *region*, a circle with center $\langle x, y \rangle$ and radius $min\{length(r) | r \in V\}$. From some orientations, this may overestimate freespace, and so may change when the robot turns in place. A region's 360° *visibility* V^* records the maximum distance sensed at 1° intervals from anywhere within it and the locations where those maxima occurred. Along with their V's and V^*'s, accumulated contradictory or overlapping regions are resolved after each target. Regions record only freespace, not obstructions.

The *skeleton* summarizes local freespace connectivity as a connected graph whose nodes are regions (Figure 2(d)). An edge there indicates that the robot moved directly between them with no other region intervening. An edge's label is a trail derived from the shortest such path, along with its length. With perfect knowledge, a region of degree one in a skeleton would be a dead-end. SemaFORR builds an initial skeleton from HLE's exploration path P and refines it after each target. A region that overlaps a highway or an intersection records that commonality for subsequent planning.

The highway graph abstracts the initial skeleton into longer contiguous chunks of freespace that support longer moves and faster planning. The skeleton is connected because it first arises from HLE's continuous exploration path and is then augmented by a sequence of targets, each of which begins where the previous one ended.

Planning in the Freespace Models

SemaFORR's hierarchical graph planner, *HP*, flexibly plans a path to target T in both the skeleton and the highway graph. The skeleton plan captures low-level spatial details



(a) Skeleton plan (b) Highway plan

Figure 3: In G5, from the robot (red square) to its target (yellow star) (a) a skeleton plan (blue) through the (pink) regions and (b) a highway plan (black) with a circled skeleton plan through regions that connects the highway plan to the target.

and incorporates knowledge gleaned during pursuit of previous targets. The highway plan exploits global connectivity and is focused on long-distance travel. By their construction, however, neither necessarily abides by the triangle inequality that A* requires to find a least-cost path. Instead, SemaFORR uses Dijkstra's algorithm to find shortest paths (Dijkstra 1959). This suffices because both graphs are at least an order of magnitude smaller than a usefully fine grid-based graph, with nodes typically of degree less than five.

For any location L, its *skeleton surrogate* is the region $\widehat{L}$, defined as the first of the following: the region that contains L, the closest region with visibility V^* that senses L, or a high-degree region near L. To plan in the skeleton for a robot at R with a target at T, HP finds $\widehat{R}$ and $\widehat{T}$ and builds a *skeleton plan* that is a shortest path sequence of regions:

$$\widehat{R} \rightarrow \ldots \rightarrow \widehat{T}$$

Similarly, for any location L, its *highway surrogate* is the intersection L^*, defined as the first of the following: an intersection that contains L, the closer endpoint of a highway that contains L, an intersection that overlaps $\widehat{L}$, or the closer endpoint of a highway that overlaps $\widehat{L}$. If neither L nor $\widehat{L}$ overlaps the highway graph, HP uses Dijkstra's algorithm to search the skeleton from $\widehat{L}$ for the shortest path to any region L^* that overlaps the highway graph. HP's *highway plan* takes the robot along a shortest path sequence of intersections from R^* through the highway graph to T^*. Unless $R^* = \widehat{R}$, the highway plan is preceded by a shortest skeleton path from $\widehat{R}$ to R^*, and unless $T^* = \widehat{T}$, it is followed by a shortest skeleton path from T^* to $\widehat{T}$:

$$\widehat{R} \rightarrow R^* \rightarrow \ldots \rightarrow T^* \rightarrow \widehat{T}$$

SemaFORR calls HP only once for a target. To prevent unnecessary highway travel, HP builds both a skeleton plan and a highway plan to reach T, and returns the shorter one. Figure 3 shows examples in G5.

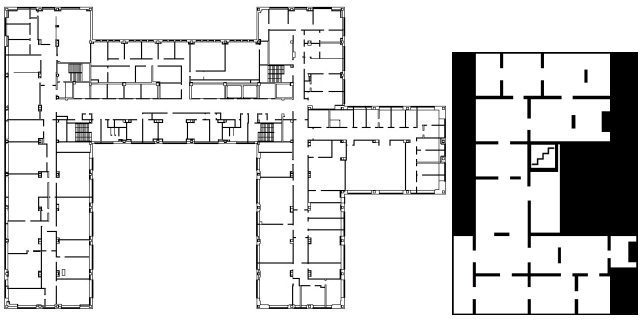
Operationalization

A hierarchical HP plan for navigation defers explicit action selection until execution time, when SemaFORR replaces its current high-level plan step with a sequence of lower-level steps. This operationalization allows the controller to decide opportunistically and to compensate robustly for sensor and actuator error. To operationalize a waypoint w, SemaFORR selects an action to *approach* it, either a move toward w directly or, based on one-step lookahead restricted to V, a turn intended to precede such a move. To operationalize a region G with center g, SemaFORR substitutes waypoints opportunistically. When the robot can sense g, SemaFORR replaces G with g. Otherwise, if the robot is in region R SemaFORR takes the known sequence of waypoints from the trail recorded on the edge from R to G, which is likely longer than a direct step.

To operationalize an intersection I with centroid i, SemaFORR treats i as a waypoint and approaches it. To operationalize a highway H between intersections I_A and I_B, first SemaFORR tries to identify the regions A and B that overlap H and are closest to I_A and I_B, respectively. If it can identify A and B and can find a skeleton path between them through a sequence of regions all of which overlap H, SemaFORR replaces H with that sequence. Otherwise, SemaFORR replaces H with the sequence of waypoints from the trail saved on the edge labels in the highway graph. The resultant sequences of regions or waypoints are readily operationalized, as described above.

HP's hierarchical plans allow SemaFORR to take unanticipated opportunities that arise as it executes a plan. To find novel shortcuts, it considers two plan steps at a time, including the one it is about to operationalize. If, for example, a plan lists the remaining sequence $A \rightarrow B \rightarrow C$ of regions, once the robot is in A it may be able to sense C's center. A novel shortcut would eliminate B from the plan and approach C instead, which SemaFORR does, whether or not the three regions' centers are colinear.

Hybrid controllers combine reactivity and planning. An autonomous car, for example, has an intended route, but also provisions for emergency braking and lane following. SemaFORR is such a hybrid. At each decision point, action selection combines reactivity, planning, and heuristics that exploit low-level information in the skeleton.



(a) H10 (b) M5

Figure 4: H10 and M5 have configurations different from G5

SemaFORR incorporates several reactive procedures to contend with anticipated situations. They go toward a sensed target, do not take actions that collide with obstacles or repeatedly rotate in place, realign to get to a sensed target or waypoint, turn to face a nearby waypoint, and leave a confined space. When SemaFORR cannot execute its operationalized plan and no other procedure takes control, it relies on heuristics to select an action. (For more details on these other procedures, see (Epstein and Korpan 2020).) Finally, when SemaFORR completes HP's plan and has still not reached the target, it relies on LLE to try to find a way.

Empirical Evaluation

Privacy and security concerns discourage a shared dataset on which to benchmark real-world indoor navigation. Instead, we evaluate SemaFORR on three large, complex environments. G5 is a 107-year-old building about the size of an entire city block and was last renovated in 2009. Figure 4 shows the other two worlds. H10 is the tenth floor of a repeatedly renovated 1937 building, and M5 is the fifth floor of New York's Museum of Modern Art. For realistic assessment, our simulation introduces small random errors into the sensor data and into action execution by small changes in the time allocated to the robot's motors.

For each world, we randomly generated 5 ordered target sets of 40 freespace locations. The robot's next target is usually outside the robot's 25m sensor range; the average Euclidean distance across all target sets between consecutive targets (which ignores obstacles) was $40.9m$ for G5, $40.0m$ for H10, and $45.8m$ for M5. SemaFORR pursues a target until it *succeeds* (comes within $\epsilon = 1m$ of it) or *fails* (exceeds a prespecified step limit s per target). After each target, SemaFORR updates its regions and skeleton from its experience. In a *run* the robot explores with HLE and then visits one target set within the decision-step limit. Based on preliminary testing in G5, all constants, including $d = 7m$ and $b = 750$ steps per exploration candidate, were set once by hand and applied uniformly to all 3 worlds. All grid cells were $1 \times 1m$.

Simulations ran SemaFORR and ROS Indigo on a Dell Precision Tower 7910 with 16 GB memory and an Intel Xeon(R) 2.40GHz x 16 CPU. After extensive experiments with a range of times for global exploration (10 to 60 minutes) and step limits (50 to 1000 steps), we balanced satisfactory performance and reasonable time for real-world exploration on M5 with $\tau = 30$ minutes and $s = 500$ steps and on H10 and G5 with $\tau = 30$ and $s = 750$.

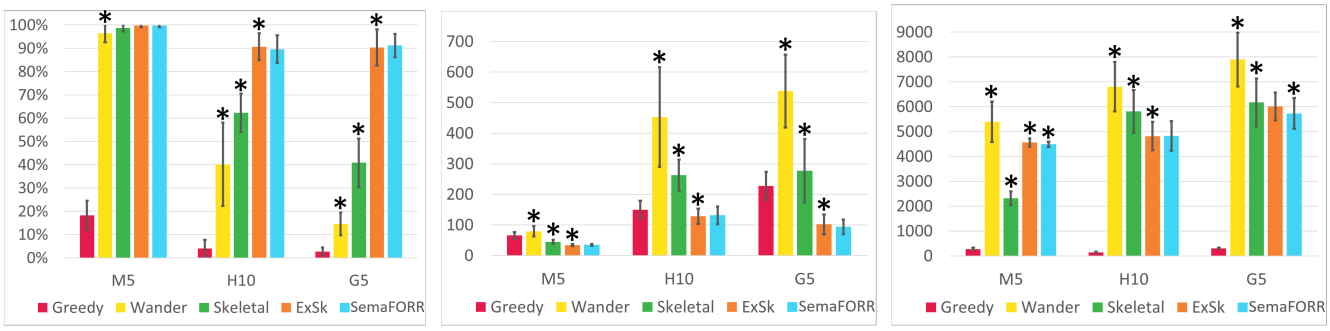
Performance metrics were success rate (fraction of targets reached), total wall clock time (including exploration, model construction, planning, decisions, and movement) and time per target in seconds, distance traveled in meters (including any exploration), and *coverage* (fraction of freespace identified in the model). Table 2 reports SemaFORR's performance. Despite the variability in time and distance across the five target sets, shown by their standard deviations, success rate and coverage were relatively consistent.

Figure 5 compares performance in 40 runs of SemaFORR (8 runs with each target set) to several alternative approaches. GREEDY, like the baseline in (Stein, Bradley, and Roy 2018), has a simple navigation strategy: take actions that move in the direction of the target but avoid obstacles and do not plan. WANDER builds an initial skeleton with HLE but does not plan. SKELETAL develops and plans in SemaFORR's skeleton only during tasks on random targets, without any exploration or highway graph. EXSK uses HLE and LLE but plans only in the skeleton.

WANDER significantly improves on GREEDY's meager success rate, and SKELETAL succeeds even more often than WANDER in H10 and G5. SKELETAL is also faster per target and travels less distance to reach its targets than WANDER. This supports our thesis that freespace planning supports satisfactory navigation in large, complex spaces, without a costly detailed map of the world or training on thousands of targets. Both EXSK and SemaFORR succeed more often and reach targets faster than SKELETAL. Although HP typically finds shorter plans in the highway graph than in the skeleton, both models stem from the same exploration path, and a highway plan is ultimately operationalized as a sequence of regions. As a result, EXSK and SemaFORR succeed about equally often, but SemaFORR travels significantly less dis-

World	Size(m)	Rooms	Free(m^2)	Dec. s	HLE τ	Success	Time(*sec*)	Time/target	Dist.(m)	Coverage
M5	54×62	14	1585	500	30	99.8%	3193.3	34.8	4485.7	94.1%
						(0.3%)	(75.7)	(1.9)	(67.0)	(0.5%)
H10	89×58	75	2627	750	30	89.6%	6771.6	131.5	4820.1	43.9%
						(3.7%)	(833.6)	(22.3)	(506.7)	(1.9%)
G5	110×70	180	4021	750	30	91.2%	5564.2	94.1	5730.0	41.8%
						(2.4%)	(550.7)	(13.8)	(400.2)	(1.1%)

Table 2: SemaFORR experiments in three challenging worlds. Standard deviations over the five target sets are in parentheses.



(a) On success rate (b) On time per target (*seconds*) (c) On total distance (*meters*)

Figure 5: The impact of reasoning and planning. An asterisk denotes better performance ($p \leq 0.05$) than the value to the left.

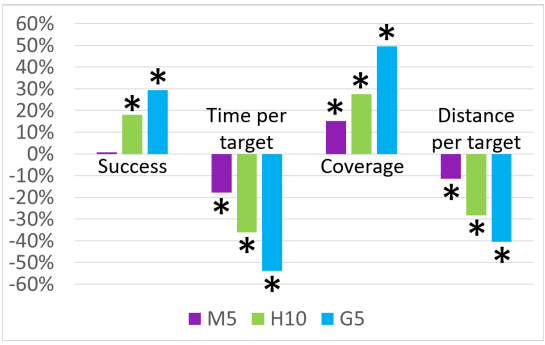


Figure 6: Performance improvements when SemaFORR explores with HLE instead of frontier exploration.

tance than ExSk to reach its targets in M5 and G5.

A second experiment compared HLE to frontier exploration (Juliá, Gil, and Reinoso 2012). We implemented and tested frontier exploration with the same step and time limits as HLE. The results in Figure 6 support our hypothesis that HLE's strategic focus on high-level connectivity is useful. SemaFORR with HLE outperformed frontier exploration; it succeeded more often in H10 and G5, and in all three worlds was faster, traveled less, and provided greater coverage.

SemaFORR navigates in real time; its average decision cycle is 0.5 seconds. Because the freespace models are an order of magnitude smaller than a usefully fine occupancy grid, HP is significantly faster than A*. In H10, for example, planning averaged 0.5 seconds per target, versus 15 seconds for A* in an occupancy grid (Epstein and Korpan 2019).

Discussion

SemaFORR's ability to balance exploration for knowledge with exploitation of its model to reach targets generalizes across different worlds. While hierarchical freespace planning suffices for simpler unknown worlds like M5, exploration is key in G5 and H10, the most challenging floors in their buildings. Indeed, navigation in G5 is so difficult that its occupants eventually color-coded the walls and hung art for guidance. Current work includes evaluation on additional worlds. For example, SemaFORR achieved 91% success given the same parameter values in another large ($130 \times 81m$) university building with 187 rooms, a space also notorious for its ability to confound its visitors.

This work highlights significant differences between indoor navigation and autonomous driving. Most of an indoor world is freespace that permits novel shortcuts; an autonomous vehicle is rigorously constrained to a small fraction of its environment. Our robot is expected to learn models of its world; an autonomous vehicle has a presumed-accurate map. Design of indoor worlds may address security or aesthetics as well as efficiency; vehicle highways are intended to expedite travel. Instead of road signs, we assumed near-perfect localization, that is, the robot knew exactly where it and the target were.

SemaFORR was inspired by work that found neural correlates for hierarchical planning and operationalization (Balaguer et al. 2016) and for chemical reinforcement of active spatial learning not motivated by escape (Mun et al. 2015). HLE was inspired by psychological evidence that young humans have an innate proclivity to explore long extents (de Hevia et al. 2014) and that active learning results in more expert navigation (Chrastil and Warren 2013). The freespace models were inspired by sketches after active exploration in complex virtual worlds (Chrastil and Warren 2013).

Although this paper focuses on indoor, built spaces, at its core is the discovery and exploitation of a hierarchy of spatial affordances that provide high-level connectivity within an environment. SemaFORR could be generalized for other spaces (e.g., outdoor, aerial, or underwater) with an expanded definition of connectivity and freespace. For example, an outdoor robot faced with different terrains could interpret "freespace" as "easy to traverse," so that exploration sought connectivity among long stretches of such terrain. An aerial robot in a building could similarly interpret "freespace" so that plans would avoid areas with strong drafts or fans. A plan would be a sequence of steps in such redefined freespace where SemaFORR could flexibly select actions to transition between those areas.

Exploration provides more knowledge, as measured by coverage. The highway graph's global knowledge prepares for unknown targets, and LLE expands upon it. For example,

after HLE on G5, initial coverage averaged 31.6% but rose to 41.8% by the end of a run. This improved SemaFORR's performance on subsequent targets (e.g., it was more likely to reach the last 5 targets in G5 than the earlier ones). SemaFORR summarizes this knowledge efficiently; it builds the passage grid, freespace grid, and highway graph only once and then discards the grids. This construction is fast; in G5, for example, it averaged 6.8 seconds.

Although evaluation here only compared against frontier exploration while the rest of the navigation architecture was held unchanged (Figure 6), comparison with other navigation architectures in the same large-scale environments is generally prohibitive because many are not available with open-source ROS implementations. Future work will compare SemaFORR with Active SLAM with RRT and topology-based navigation. Additionally, a highway-biased version of RRT instead of a sequence of regions could allow flexibility in plan construction.

The spatial models not only speed planning (compared to A* on an occupancy grid) but also allow the planner to defer operationalization on broadly-defined subgoals (e.g., movement between regions) until execution time. As a result, plans are robust to realistic actuator or sensor errors as well as to previously undetected obstructions (e.g., the small circles in Figure 3(b) that represent support columns). Hierarchical planning in the two models also allows the controller to recognize and exploit novel shortcuts and to support intervention when the robot fails to make progress. HP exploits the spatial models, and thereby reaches more targets in G5 and H10 faster and/or with shorter paths than SKELETAL, which lacks exploration and the highway graph.

The highway grid is static, but SemaFORR continues to modify the skeleton after each target. It adds highway-like chains of regions and dead-ends (rooms) to the original skeleton. Although planning may become somewhat slower as the skeleton grows, it also supports more informed decisions and addresses targets in previously unvisited areas.

Current work evaluates performance on larger task sets and repetition on the same targets. Based on trials with other parameter settings, however, neither more HLE time nor a higher step limit is likely to reach every target in H10 and G5. We believe metareasoning and reactive planners could help. For example, metareasoning would trigger periodically to restart HLE when a new passage is detected. Current work addresses these ideas and dynamic time allocation for exploration based on coverage and $|C|$. Current work also evaluates SemaFORR with imperfect localization, where our simulator adds uniform noise to the robot's pose.

SemaFORR readily reasons over finite action sets to produce the trajectories required. In preliminary work, evaluation with much larger action sets did not significantly change performance. Although a larger action set could be necessary for three-dimensional freespace or other domains that require precision, a limited action set did not prevent nuanced trajectories in two-dimensional freespace.

Several issues remain. HLE only explores if it has at least one candidate. Like architects, we have our navigator enter a world where it can detect a candidate (e.g., at the elevators). Search for a first candidate instead is a topic for future work. While H10 is smaller than G5, HLE does not always capture all H10's highways because H10 lacks the cycles that make it easier in G5 to double back from another direction. Reordering HLE's candidates to drive to uncovered areas could address that, but would also require more time. HLE captures some diagonal passages but, as formulated, makes a diagonal highway less likely. Of course, only exhaustive exploration can guarantee perfect knowledge of an environment. Without it, navigation and plans for it may be less than ideal. Although a thoughtful teacher familiar with a world could supplement HLE with an instructive sequence of targets, that approach would sacrifice autonomy.

In this work, a target is a fixed location in the environment, selected randomly in advance during experimental design. In previous work, SemaFORR navigated successfully to a moving target that had a tracking device (Aroor, Epstein, and Korpan 2018). SemaFORR could be extended to use its freespace model to predict probable locations of a moving target and use LLE to explore the area where the target was last sensed. In that earlier work, we also extensively evaluated navigation in environments crowded with other agents. Given a map, we have modeled crowds with various properties and developed appropriate and successful probabilistic learning mechanisms for them. SemaFORR was robust to the addition of these dynamic obstacles. Future work will adapt HLE, LLE, and HP for unfamiliar worlds populated with human crowds.

SemaFORR has multiple potential applications. Architects could use it during design, to analyze spatial connectivity. The models, particularly the highway graph, could predict connectivity on other floors of the same building. Rescue teams could use the models to find alternate routes when floor plans become invalid.

Meanwhile, SemaFORR's ability is noteworthy. It learns freespace models quickly and navigates with them efficiently. Its high-level exploration provides a global framework for learning, one that low-level exploration augments when a target lies in unknown territory. Hierarchical freespace planning then exploits this knowledge to support effective, robust navigation in complex worlds.

Acknowledgments

This work was supported in part by The National Science Foundation under CNS-1625843 and CF-1231216. The authors thank Matthew Wilson for discussions on the neuroscience of navigation, Anoop Aroor for Menge ROS, Gil Dekel for his pioneering work on the spatial model, and the anonymous referees for their many constructive suggestions.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11): 832–843.

Aroor, A.; Epstein, S. L.; and Korpan, R. 2018. Online learning for crowd-sensitive path planning. In *Int. Conf. on Autonomous Agents and MultiAgent Systems*, 1702–1710.

Balaguer, J.; Spiers, H.; Hassabis, D.; and Summerfield, C. 2016. Neural mechanisms of hierarchical planning in a virtual subway network. *Neuron* 90(4): 893–903.

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.

Chrastil, E. R.; and Warren, W. H. 2013. Active and passive spatial learning in human navigation: Acquisition of survey knowledge. *Journal of experimental psychology: learning, memory, and cognition* 39(5): 1520.

de Hevia, M. D.; Izard, V.; Coubart, A.; Spelke, E. S.; and Streri, A. 2014. Representations of space, time, and number in neonates. *Proceedings of the National Academy of Sciences* 111(13): 4809–4813.

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*, volume 142 of *LIPIcs*, 22:1–22:20.

Epstein, S. L.; and Korpan, R. 2020. Metareasoning and Path Planning for Autonomous Indoor Navigation. In *ICAPS 2020 Workshop on Integrated Execution / Goal Reasoning*.

Forster, C.; Carlone, L.; Dellaert, F.; and Scaramuzza, D. 2017. On-Manifold Preintegration for Real-Time Visual-Inertial Odometry. *IEEE Trans. Robotics* 33(1): 1–21.

González-Banos, H. H.; and Latombe, J.-C. 2002. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research* 21(10-11): 829–848.

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.

Juliá, M.; Gil, A.; and Reinoso, O. 2012. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots* 33(4): 427–444.

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

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

Lavalle, S. M. 1998. Rapidly-exploring random trees : a new tool for path planning. Technical Report TR 98-11, Iowa State.

Leung, C.; Huang, S.; and Dissanayake, G. 2006. Active SLAM using Model Predictive Control and Attractor based Exploration. In *IROS*, 5026–5031. IEEE.

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.

Moravec, H.; and Elfes, A. 1985. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, 116–121.

Mun, H.-S.; Saab, B.; Ng, E.; McGirr, A.; Lipina, T.; Gondo, Y.; Georgiou, J.; and Roder, J. 2015. Self-directed exploration provides a Ncs1-dependent learning bonus. *Scientific Reports* 5(1): 1–13.

Ocana, M.; Bergasa, L.; Sotelo, M.; Nuevo, J.; and Flores, R. 2005. Indoor robot localization system using WiFi signal measure and minimizing calibration effort. In *International Symposium on Industrial Electronics*, volume 4, 1545–1550.

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*, volume 3.2, 5.

Se, S.; Lowe, D.; and Little, J. 2002. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The International Journal of robotics Research* 21(8): 735–758.

Selin, M.; Tiger, M.; Duberg, D.; Heintz, F.; and Jensfelt, P. 2019. Efficient autonomous exploration planning of large-scale 3-D environments. *IEEE Robotics and Automation Letters* 4(2): 1699–1706.

Stachniss, C.; Thrun, S.; and Leonard, J. 2016. *Simultaneous Localization and Mapping*, chapter 46, 1153–1176. Springer, 2 edition.

Stein, G. J.; Bradley, C.; Preston, V.; and Roy, N. 2020. Enabling topological planning with monocular vision. In *International Conference on Robotics and Automation*, 1667–1673. IEEE.

Stein, G. J.; Bradley, C.; and Roy, N. 2018. Learning over Subgoals for Efficient Navigation of Structured, Unknown Environments. In *CoRL*, volume 87 of *Proceedings of Machine Learning Research*, 213–222. PMLR.

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.

Wayne, G.; Hung, C.-C.; Amos, D.; Mirza, M.; Ahuja, A.; Grabska-Barwinska, A.; Rae, J.; Mirowski, P.; Leibo, J. Z.; Santoro, A.; et al. 2018. Unsupervised predictive memory in a goal-directed agent. ArXiv preprint arXiv:1803.10760.

Whaite, P.; and Ferrie, F. P. 1997. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(3): 193–205.

Yamauchi, B. 1997. A frontier-based approach for autonomous exploration. In *International Symposium on Computational Intelligence in Robotics and Automation*, 146–151. IEEE.

Zhu, D.; Li, T.; Ho, D.; Wang, C.; and Meng, M. Q.-H. 2018. Deep reinforcement learning supervised autonomous exploration in office environments. In *IEEE International Conference on Robotics and Automation*, 7548–7555. IEEE.