Using Language to Generate State Abstractions for Long-Range Planning in Outdoor Environments

Matthew Berg*‡, George Konidaris*, and Stefanie Tellex*

***Abstract*— Robots that process navigation instructions in large outdoor environments will need to operate at different levels of abstraction. For example, a land-surveying aerial robot receiving the instruction *“go to Boston and go through the state forest on the way”* must reason about a long-range goal like *“go to Boston”* while also processing a finer-grained constraint like *“go through the state forest.”* Existing approaches struggle to plan such commands because of the immense number of locations and constraints that can be expressed in language. We introduce a hierarchical representation of outdoor environments and a planning approach that dynamically compacts the robot’s state space to enable tractable planning in city and state-scale environments. Our approach leverages natural abstractions in real-world map data, coupled with abstractions generated from users’ instructions, to generate filtered environment views that accelerate planning while supporting a robot’s ability to obey complex temporal goals and constraints at different levels of abstraction. We evaluate our approach on seven templates of LTL$_f$ formulas and in an 80 kilometer-radius environment containing over 250,000 locations downloaded from OpenStreetMap. The results show our approach enables planning in seconds or minutes in a large outdoor environment while still satisfying the task specification.**

I. Introduction

Robots are increasingly deployed to outdoor domains for autonomous missions: fixed-wing drones are delivering critical medical goods [1], quadcopters are surveying infrastructure and land [2], and autonomous trucks are being tested on public roads [3]. These robots will have to be tasked by humans—for example, a pilot providing high-level navigation tasks to a drone delivery fleet, or an autonomous truck operator instructing the vehicle to detour towards better weather. As interactions with outdoor robots become more common, we require an interface that allows a human to naturally give commands to a robot, while exploiting all the knowledge that can be extracted from such commands.

Natural language offers an intuitive and expressive interface for human-robot interaction. There is a body of work on resolving natural language commands to plans for a robot [4]. Motivated by the complex temporal goals and constraints often expressed in natural language, recent work has focused on non-Markovian commands [5, 6, 7, 8]. These approaches leverage a sequential decision-making framework that supports non-Markovian tasks but is computationally expensive due to the necessary processing of extended state histories. For example, a robot receiving the command *“go to Boston and go through the state forest on the way”* must retain and evaluate its state history—a violation of the Markov property—to ensure a visit to a *“state forest”* then *“Boston.”* As the robot’s environment grows large, it becomes increasingly difficult to plan to satisfy these extended state sequences.

*Brown University, Providence, RI, 02912, USA. Emails: matthew_berg@alumni.brown.edu, {gdk, stefie10}@cs.brown.edu.
‡ Work done prior to joining Amazon.

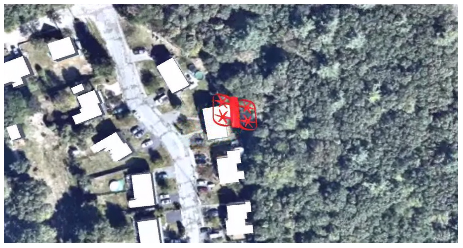


Fig. 1. A simulated Skydio R1 flying over Massachusetts, USA after being tasked to *“go to Boston and go through the state forest on the way.”* The quadcopter is operating in an 80-kilometer radius environment based on map data from OpenStreetMap. Our approach generates the environment representation and computes the path plan; existing language modules [6, 7] ground the natural language command to a structured form for planning.

We present a hierarchical representation of outdoor environments and a planning approach that leverages map and language-based abstractions to enable tractable planning of non-Markovian tasks in city and state-scale environments. The hierarchical map representation is constructed using data from OpenStreetMap [9]. Locations are organized into categories such as `landmarks` and `cities` and their geometries are separately stored for planning. We assume the hierarchical representation is combined with a structured encoding of a natural language command, computed with an existing model such as Berg et al. [7]. The planner uses work from Oh et al. [6] to decompose this encoding into subtasks that are solved with a sequential decision-making model. As the planner progresses through subtasks, it evaluates semantic similarities between a user’s location referring expressions and real-world locations to further reduce environment size.

We evaluate our approach in 30 and 80-kilometer radius environments located in the Northeast United States. The tests use 7 templates of LTL$_f$ formulas that specify straight-line start-to-goal distances of up to 80 kilometers. We additionally test the example task to demonstrate language-to-plan functionality with existing language modules [6, 7] and a simulated aerial robot (Figure 1). Our approach can compute plans on the order of seconds to minutes from an environment containing over 250,000 locations. The results show that the map and language-based abstractions enable tractable planning in large outdoor environments while generating similar or shorter-length paths compared to an existing baseline that takes up to hours to run.

II. Related Work

Route directions can be ambiguous and refer to an immense number of paths in the environment. Lovelace et al. [10] discuss a three-step model of generating route directions: spatial knowledge, choice of route, and translating the route into verbal instructions. A robot must reverse these steps in a way that satisfies the user's intent and is computationally tractable. There has been work on modeling large-scale spaces [11] and following instructions in spaces containing a variety of environment features [12, 13, 14]. Matuszek et al. [12] parse natural language route instructions into a formal path description language more easily interpreted by the robot's planner. To ensure tractable planning, they constrain the state space of potential paths using actions available to the robot. Kollar et al. [13] introduce the spatial description clause, a different formalism that allows a robot to probabilistically reason about spatial relationships in route directions. Their approach assumes route directions are given sequentially. However, directions can be non-Markovian.

A Markov Decision Processes (MDP) is a sequential decision-making framework previously used to solve non-Markovian natural language commands [5, 6, 7, 8]. MDPs are a convenient choice for the fully-observed planning problem because they are compatible with an intermediate, structured representation of a user's language. Planning in partially-observed environments is an active thread of research [15, 16, 17, 18], however, our work focuses on fully observable environments generated from geographic data. MDPs generally struggle to operate tractably in large state spaces. Oh et al. [6] introduce the Abstract-Product Markov Decision Process, a framework that uses abstraction to accelerate planning performance on non-Markovian tasks. AP-MDPs decompose a high-level task into multiple subproblems, each solved by recursively lower-level MDPs. Our approach uses an AP-MDP that is modified to support an environment containing thousands of real-world locations. We use deterministic path planning where possible, and filter the state space for MDP-based planning using semantic information in the map. Other work combines learning and planning to find hierarchical MDP representations [19], but is not suitable for tasks where the agent cannot repeatedly execute the task in order to learn.

There is a body of work on grounding natural language to structured representations [5, 6, 7, 8, 20, 21, 22, 23, 24]. One representation is Linear Temporal Logic, a first-order logic capable of expressing non-Markovian goals and constraints. LTL formulas can be translated into an equivalent automaton [25] that an MDP uses to satisfy a task's temporal objectives. Gopalan et al. [5] present a sequence-to-sequence model [26] for grounding natural language to a variant of LTL [27]. Berg et al. [7] introduce a model to ground commands with unrestricted location referring expressions; this work enables a robot to process commands in unseen environments but does not support a spatial hierarchy. Our work adapts their landmark resolution model to support a map representation and tasks defined at multiple levels of abstraction.

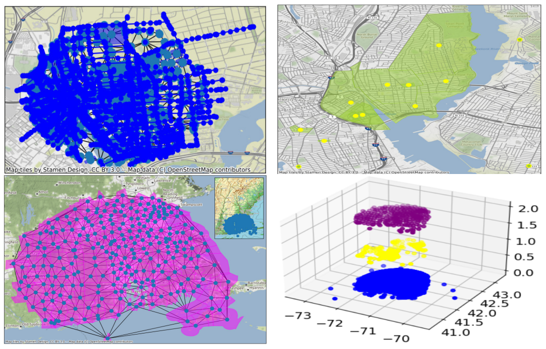


Fig. 2. 80km map. The center is in Brown University, Providence, RI, USA. The `landmark` (top left, partial view), `neighborhood` (top right, partial view), `city` (bottom left) levels and hierarchical map are shown.

III. Planning for Non-Markovian Instructions in Large Outdoor Environments

Consider the non-Markovian task *"go to Boston and go through the state forest on the way."* The user is referring to locations that are potentially far apart and difficult to localize to without a map. Further, the reference to Boston is implicitly referring to hundreds or thousands of sub-locations, while the reference to a state forest is referencing a single location. As the environment grows large, the number of possible locations and sub-locations can become computationally intractable. A robot needs a structured representation of the environment that ideally leverages spatial abstractions to make the planning problem tractable.

Our approach constructs a hierarchical map representation using data from OpenStreetMap [9], a publicly available map service containing rich spatial hierarchies and semantic information. The map representation contains a variety of locations such as shops, streets, parks, forests, and more. This representation is combined with a structured encoding of the task and supplied to an Abstract-Product Markov Decision Process (AP-MDP)-based planner [6], which outputs a sequence of global coordinates satisfying the task's temporal and spatial objectives. Our approach focuses on computing higher-level path plans, and assumes the robot contains lower-level autonomous systems that respond to local obstacles and changes in the environment.

A. Linear Temporal Logic on Finite Traces

LTL is a domain-independent formalism that supports temporal conditions with an infinite time horizon and is capable of capturing non-Markovian goals and constraints. We consider finite-length tasks and use LTL on Finite Traces (LTL_f) [28]. This work follows the syntax:

$$\phi ::= \alpha \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \mathcal{F}\phi \mid \mathcal{G}\phi \mid \phi\,\mathcal{U}\psi$$

where $\alpha \in A$ is an atomic proposition, $\neg, \wedge, \vee$ are logical negation, conjunction, and disjunction, ϕ and ψ are LTL_f formulas, $\mathcal{F}$ denotes *finally*, $\mathcal{G}$ denotes *globally*, and $\mathcal{U}$ denotes *until*. See De Giacomo and Vardi [28] for semantic interpretations of LTL_f.

B. Map Generation

The hierarchical map representation (Figure 2) uses abstraction to create a compact yet semantically expressive environment for planning from geographic data. The map representation contains three levels: `landmarks` such as forests, buildings, and streets, `neighborhoods`, and `cities`. The map representation's data is downloaded from OpenStreetMap (OSM) [9]. OSM maps contain nodes, ways, and relations. A node is a single global coordinate, a way is a collection of nodes, and a relation is a collection of nodes, ways, and/or relations [29]. The hierarchy is based on the geometry and semantic data of these elements—we define `landmarks` as named nodes, `neighborhoods` as nodes and ways tagged as neighborhoods, and `cities` as relations at OSM's administrative level 8.

The map representation is defined as $MP = (G, H, T, Z)$:

1) G is the graph set, centered on a single (latitude, longitude) coordinate. $g^i \in G$ is the graph at abstraction level i. Level $i = 0$ corresponds to `landmarks` (Figure 2, top left), $i = 1$ to `neighborhoods` (top right), and $i = 2$ to `cities` (bottom left).
2) H is the hierarchical map composed of all $g^i \in G$. This graph unifies the `landmark`, `neighborhood`, and `city`-level graphs (Figure 2, bottom right).
3) T is the transition map composed from $t^i \in T$ Ball Trees [30]. The planner queries the transition map to move between levels of the hierarchy.
4) Z is the filtered graph set. Each $\zeta^i \in Z$ is a sub-graph of g^i containing the filtered state space for the subtask the planner is currently solving (Figure 5).

To maintain a compact map representation, each g^i is constructed from the Delaunay triangulation [31] of locations' centroids.[1] Locations' geometries are separately stored. They are used to compute hierarchical relationships during map generation and hierarchical transitions at plan-time. When the planner transitions to a new level of the hierarchy, it uses a Ball Tree $t^i \in T$ from the transition map to efficiently query for nearby map features. Each Ball Tree is constructed from the centroids of the corresponding g^i using the Haversine distance metric [32]. The map representation contains textual semantic data from OSM, such as building type, waterway type, and land use [33]. The filtered graph set Z leverages this semantic data to prune centroids and edges that are peripheral to the planner's current subtask. These graphs are discussed in Section III-D.

C. Planning

LTL$_f$ tasks can be planned with a Markov Decision Process (MDP), a model for sequential decision making in stochastic environments. We assume stochastic transitions are rare at the map representation's levels of abstraction and do not model them in our approach.

[1]The geometric centroid is used at the `landmark` and `neighborhood` level. At the `city` level, an administrative center (e.g., a town hall) is used when available in the map data; otherwise, the geometric centroid is used.

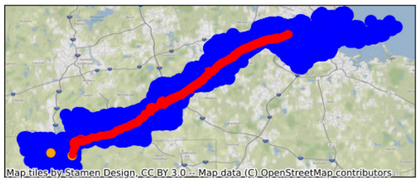

Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

Fig. 3. A* planning within a `landmark`-level projection of the `city`-level plan computed for subtask (q_1, q_2). Blue points show `landmark`-level centroids within the projection, red points show the `landmark`-level path, and orange points show where a `landmark-city` transition is made during planning. The path starts at Quaddick State Forest, Thompson, Connecticut and completes at a `landmark` in Boston, Massachusetts.

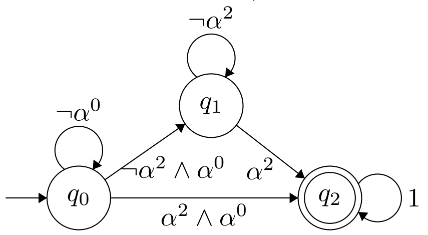


Fig. 4. DFA for the command *"go to Boston and go through the state forest on the way."* The LTL$_f$ formula is $\mathcal{F}(\alpha^0 \wedge \mathcal{F}(\alpha^2))$, where α^0, α^2 are atomic propositions at the `landmark` and `city` level. The transition from q_0 to q_2 is considered spatially infeasible and removed before planning.

Definition 1. Markov Decision Process (MDP): A deterministic MDP is defined by the tuple $M = (S, A, \gamma, T, R)$, where S is a set of states, A is a set of actions, γ is the discount factor, $T : S \times A \rightarrow S$ is the transition function, and $R : S \rightarrow \mathbb{R}$ is the reward function. The robot seeks to find a policy $\pi : S \rightarrow A$ that maximizes the sum of discounted rewards. An MDP contains $|A|^{|S|}$ policies and there exists at least one optimal policy π^*. Our approach uses Value Iteration [34] to find π^*. Convergence is guaranteed because the space of policies is finite and the robot iteratively improves its policy.

States are defined by mappings between LTL$_f$ propositions and locations' geometry. For example, the state *Boston* in $\mathcal{F}(\textit{state forest} \wedge \mathcal{F}(\textit{Boston}))$ is true when the robot is within Boston's polygon. Actions are defined by edges between locations, for example, the robot takes action a to travel from its current state to Boston. An MDP can be extended with a labeling function that maps locations to atomic propositions.

Definition 2. Abstract Labeled Markov Decision Process (AL-MDP) [6]: An AL-MDP is defined by the tuple $M^i = (S^i, A^i, T^i, s^i_0, AP, L^i, R^i)$, where i is the level of abstraction, S^i, A^i, T^i, s^i_0, and R^i are the MDP components at abstraction level i, AP is the set of atomic propositions, and $L^i : S^i \rightarrow 2^{AP}$ is the labeling function that maps states to the atomic propositions. Propositions can be defined at different levels of abstraction, but a state s^i is only assessed against propositions at level i or higher. A composition of AL-MDPs $\{M^0, ..., M^i, ..., M^l\}$ (l = the highest level of abstraction) supports fluid planning at different levels of abstraction.

A separate formalism ensures the planner obeys temporal goals and constraints (Figure 4):

Definition 3. Deterministic Finite Automaton (DFA): Every LTL$_f$ formula can be translated into an equivalent DFA

[28]. A DFA is defined by the tuple $\mathcal{B} = (Q, \Sigma, \delta, q_0, F)$, where Q is the set of states, Σ is the alphabet, $\delta : Q \times \Sigma \rightarrow Q$ is the transition function, q_0 is the initial state, and F is the acceptance condition.

Taking the product of a DFA and AL-MDPs enables the planner to satisfy temporal and spatial objectives. This construction is inspired by an Abstract Product Markov Decision Process (AP-MDP) [6], the sequential decision-making framework we use in our approach. During planning, an LTL$_f$ state transition is analogous to a DFA transition, which we refer to as a subtask and denote (q_k, q_{k+1}). Each subtask is solved using an AL-MDP at the lowest level of abstraction across the stay condition (constraint on $q_k \rightarrow q_k$) and goal condition (constraint on $q_k \rightarrow q_{k+1}$). The action-minimizing path satisfying the DFA's acceptance condition is the solution. See Oh et al. [6] for complete details.

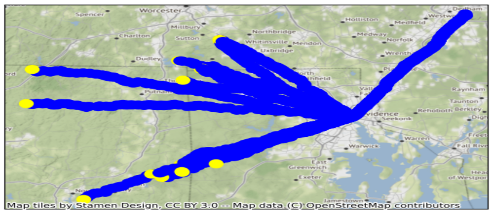
Discontiguous connections—for example, an edge between `city` A and `city` B that intersects `city` C—can exist because locations' centroids are connected independent of their geometry. We adapt the AP-MDP reward function to handle these connections. Our approach validates the stay condition is satisfied along all states from the current state, s_p^i, to the next state, $s_p^{i'}$. If the stay condition holds, a reward is assigned based on $s_p^{i'}$. Let a^i be the current action, e be the edge traversed to complete action a^i, q_k be the current state in the DFA, and V be the set of states mapping to the stay and goal constraints' atomic propositions that intersect with e. The reward r is assigned as:

$$r = \begin{cases} 100, & \text{if } \textbf{i. } T^i(s^i, a^i, s^{i'}) = s^{i'} \textit{ and} \\ & \quad \textbf{ii. } \delta(q_k, L(v)) = q_k \ \forall v \in V - s^{i'} \textit{ and} \\ & \quad \bullet\ \delta(q_k, L(s^{i'})) = q_{k+1} \\ -1, & \text{if } \textbf{(i)} \textit{ and } \textbf{(ii)} \textit{ and} \\ & \quad \bullet\ \delta(q_k, L(s^{i'})) = q_k \\ -100, & \text{otherwise.} \end{cases}$$

In our approach, AL-MDPs exclusively operate at the highest-possible level of abstraction, forgoing planning with recursively lower-level MDPs in favor of fast A* path planning. MDP-computed plans are therefore not guaranteed to resolve to the base `landmark` level of abstraction, for example, the AL-MDP for subtask (q_1, q_2) plans a sequence of `cities` to Boston. A `landmark`-level path is computed by planning within a lower-level projection of the higher-level states (Figure 3). This projection is constructed using two R-Trees [35, 36]—one storing edges, the other storing locations' geometries—that efficiently identify `landmark`-level edges within higher-level locations. The edges and their vertices form a `landmark`-level subgraph where A* commences.

D. Semantic Filtering

Goals and constraints are often planned at the base level of abstraction, for example, the planner operates at the `landmark` level to ensure its path reaches a state forest. However, planning with a `landmark`-level MDP is computationally intractable as the environment grows large. We approach this problem by filtering the map representation to paths that reach semantically significant locations. We assume a language module resolves all atomic propositions to salient locations; by extension, subtasks' stay and goal conditions resolve to salient locations. We use these locations to create subtask-unique state spaces for planning.



Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors

Fig. 5. Filtered graph for subtask (q_0, q_1). The blue points show the `landmark`-level paths and the yellow points show salient map locations satisfying the goal condition (state forests).

Our novel semantic filter generates a set of paths $\Omega_{q_k,q_{k+1}}$ based on the stay and goal conditions' salient map locations $\Upsilon_{q_k,q_{k+1}}$. These paths are generated at the lowest level of abstraction specified by the stay/goal conditions. For example, Ω_{q_0,q_1} contains `landmark`-level paths to state forests and Boston, while Ω_{q_1,q_2} contains a `city`-level path to Boston. The filter will additionally generate paths that avoid other salient locations when the stay/goal conditions contain multiple atomic propositions. All vertices and edges in the path set are used to create the filtered graph $\zeta^i_{q_k,q_{k+1}}$ at level i in the map representation (see Figure 5). The filtered graph is then supplied to the MDP planner, which finds the action-minimizing path satisfying the LTL$_f$ formula. We outline the semantic filtering procedure in Algorithm 1.

Algorithm 1: Semantic Filtering

```
Input: $g^i$, environment graph at level $i$
       $H$, merged map (for hierarchy transitions)
       $T$, transition map (for hierarchy transitions)
       $v_0$, start location in $g^i$
       $AP_{q_k,q_{k+1}}$, atomic propositions in the goal
       and stay conditions of subtask $(q_k, q_{k+1})$
Output: $\zeta^i_{q_k,q_{k+1}}$, the filtered graph for $(q_k, q_{k+1})$
1  Initialize $\Omega_{q_k,q_{k+1}}$
2  $\Upsilon \leftarrow$ retrieve_map_features($AP_{q_k,q_{k+1}}$)
3  for $\alpha$ in $AP_{q_k,q_{k+1}}$ do
4      for $v$ in $\Upsilon_\alpha$ do
5          $\omega_{to} \leftarrow$ astar($v_0$, $v$, $g^i$, $H$, $T$)
6          add($\Omega_{q_k,q_{k+1}}$, $\omega_{to}$)
7          if $|\Upsilon \setminus \Upsilon_\alpha| > 0$ then
8              avoids $\leftarrow \Upsilon \setminus \Upsilon_\alpha$
9              $\omega_{avoid} \leftarrow$ astar($v_0$, $v$, $g^i$, $H$, $T$, avoids)
10             add($\Omega_{q_k,q_{k+1}}$, $\omega_{avoid}$)
11         end
12     end
13 end
14 $\zeta^i_{q_k,q_{k+1}} \leftarrow$ subgraph($g^i$, $\Omega_{q_k,q_{k+1}}$)
15 return $\zeta^i_{q_k,q_{k+1}}$
```

IV. Results

We test planning performance, efficiency, and spatial efficiency on LTL$_f$ formulas specifying long-range tasks. We also benchmark performance on the example task with different location reference resolution thresholds and show a simulated aerial robot following the plan computed by our approach.

A. Environment

We use two environments centered on Providence, RI, USA. Each environment was generated by querying OpenStreetMap for geographic data within a 30 or 80 kilometer radius of the center point. The `landmark` and `city`-level queries returned locations with centroids past the query radius; we kept these locations. There are 44,822 locations composed from 44,615 landmarks, 151 neighborhoods, and 56 cities in the 30-kilometer environment and 251,184 states composed from 250,502 landmarks, 408 neighborhoods, and 274 cities in the 80-kilometer environment.

B. Long Range Planning

We created a dataset of 112 LTL$_f$ formulas to test in the 80-kilometer environment.[2] They are equally distributed over 7 templates that are defined with 1 to 4 atomic propositions. Each proposition refers to a unique location; the a proposition is varied and used to represent the goal state while the b, c, and d are constant and represent constraints. Increasing the number of propositions and LTL$_f$ operators can increase the number of states and transition complexity in the corresponding DFA. We analyze planning performance relative to this 'temporal complexity.' To test flat and hierarchical planners on the same tasks, all propositions are defined at the `city` (highest) level. The flat planners plan `city`-level tasks at the `landmark` level by using hierarchical relationships (e.g., `landmark` X is in `city` Y). We test with 42 formulas that can be satisfied in the 30 kilometer environment, and all 112 formulas in the 80 kilometer environment.

Results are shown in Figure 6. We test 3 planning configurations: no abstract planning and semantic filtering (NA/S), abstract planning and semantic filtering (A/S), and abstract planning without semantic filtering (A/NS). For each configuration, we evaluate computing time (performance) and Bellman backups (planning efficiency) as a function of the LTL$_f$ template. Computing time includes reference to location resolution,[3] Value Iteration, semantic filtering (NA/S and A/S), and base-level path resolution (A/S and A/NS). We additionally compare planned path lengths to evaluate spatial efficiency. In the 30-kilometer environment, we compare to a baseline that does not use abstraction nor semantic filtering (NA/NS) and is essentially a flat MDP processing the entire `landmark`-level state space. An unfiltered state space lets the planner consider more paths but causes orders of magnitude slower planning time. NA/NS is not benchmarked in the 80 kilometer environment because the number of locations makes it computationally intractable.

The abstract planning configurations sustain the highest performance as temporal complexity increases. The small performance difference between A/S and A/NS suggests semantic filtering provides low performance gains in smaller state spaces like the `city` level. However, semantic filtering achieves larger performance gains with `landmark`-level planning. NA/S tractably plans at the `landmark` level across all tested templates, while in the 30-kilometer environment, the baseline (NA/NS) cannot compute a plan for any template in tractable time. NA/NS exhibits larger performance and efficiency variations due to variability in the `landmark` level's size. Some paths go through urban areas with many locations, while others go rural areas with less locations. Variation can also be attributed to implementation-level modifiers like the complexity of polygon checks along a path. There are also differences between plans in the 30 and 80-kilometer environments because the environments contain slightly different map data. Last, we note temporally correct paths were computed for all tested formulas. These results show the combination of the hierarchical map representation and semantic filtering is able to efficiently and effectively find plans in both smaller and large environments.

Our approach aims to compute spatially efficient paths by leveraging A* when possible. Using a spatial heuristic helps to avoid planning paths that take fewer actions but cover longer distances. We compare planned path lengths on temporally simple tasks ($\mathcal{F}(a)$) in the 30-kilometer environment and observe cases where planning without a spatial heuristic (NA/NS and A/NS) plan longer paths (Figure 6, upper right), such as paths that traverse long edges at the map representation's boundary. We additionally compare the number of times each configuration plans the shortest path in the 80-kilometer environment (Figure 6, lower right) and observe NA/S tends to compute the shortest paths. A/S computes shorter paths for 'go to' tasks—$\mathcal{F}(a)$, $\neg b \mathcal{U} a$, $\mathcal{F}(a) \wedge \mathcal{G}(\neg b)$—while A/NS tends to compute shorter paths for 'go through' tasks——$\mathcal{F}(b \wedge \mathcal{F}(a))$ and variants. A/S struggles with 'go through' tasks because semantic filtering computes paths to state boundaries without knowing the most efficient boundary for the global plan. When filtering at the `landmark` level, states are small and boundary selection is not a significant issue, but at the `city` level, the size of map locations can create a more difficult planning problem. We expect there is some loss in spatial efficiency because semantic filtering is aggressively greedy: Taking the shortest `landmark`-level path between city A to city B can force the planner to take a far longer path to city C. An important future direction is quantifying the loss of optimally alongside a survey of different filtering heuristics.

C. Planning with a language command

We test our approach with the command *"go to Boston and go through the state forest on the way"* using the language-to-LTL model from Oh et al. [6] and the landmark resolution approach that we adapt from previous work [7].

[2]`https://github.com/matthewberg/LongRangeLTLTasks`

[3]This procedure is handled by the language module and is effectively a search over the hierarchical map. Its performance depends on the map construction and we therefore include it in the results.

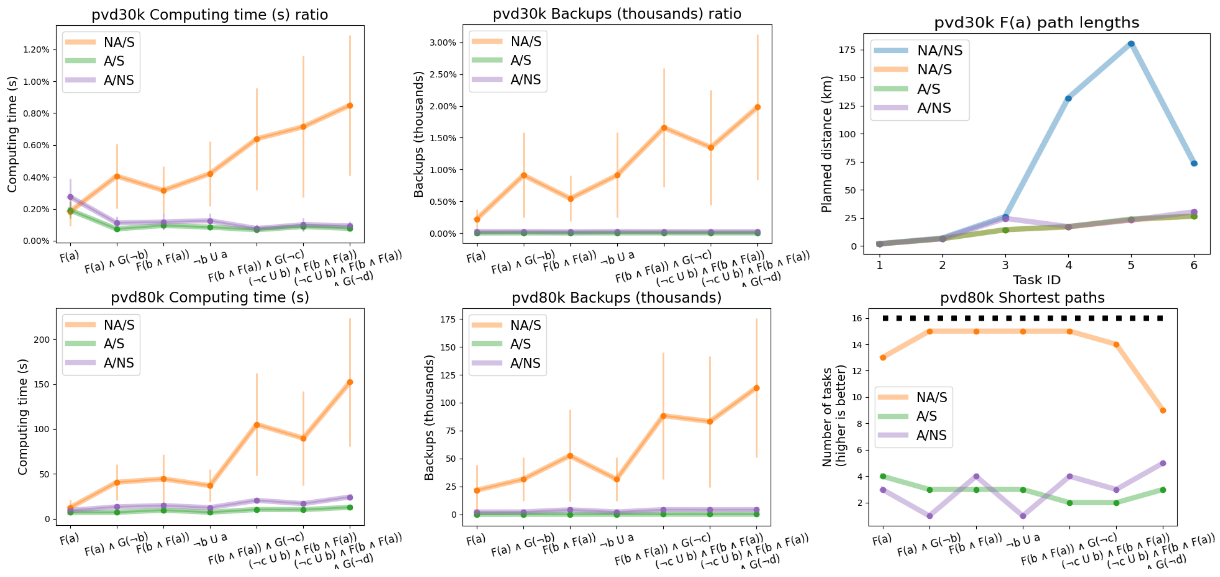


Fig. 6. Long Range Planning Results. The 30-kilometer environment benchmarks for performance and planning efficiency (top row, left and center) show the proportion of time and backups compared to baseline. The 80-kilometer environment benchmarks (bottom row, left and center) only compare planning configurations from our approach. Spatial efficiency benchmarks (right) show planned path lengths for $\mathcal{F}(a)$ tasks in the 30-kilometer environment and the number of shortest-length paths for tasks over all LTL$_f$ templates in the 80-kilometer environment.

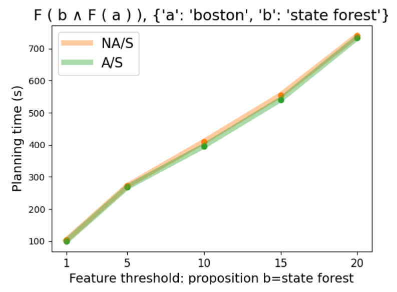


Fig. 7. Planning results for the example task *"go to Boston and go through the state forest on the way"*. The horizontal axis shows the location resolution threshold for proposition b, the vertical axis shows planning time.

The language module's input is the example command and its output is an LTL$_f$ formula with labeled atomic propositions. Our approach handles ambiguous utterances like the *"state forest"* by planning paths to all potential goal locations (state forests). Therefore, the number of potential locations must be capped to ensure tractable performance. Results collected in the 80-kilometer environment (Figure 7) demonstrate the relationship between this threshold and planning time: As the threshold increases, the filtered `landmark`-level environment grows and planning time increases. The marginal performance difference between A/S and A/NS suggests `landmark`-level planning to the state forest takes up the bulk of planning time. Balancing the threshold and planning performance is a design decision that could vary across applications and environments, for example, a land surveyor may increase the threshold for nature and water body-related propositions while a delivery operator may set the threshold to 1 and refer to locations with unique names.

D. Simulated Robot Demonstration

We tested our approach with an aerial robot simulator from previous work [7] (Figure 1). The simulator is built with Unity game engine [37], MapBox [38], and ROS and ROS# [39, 40]. The simulated robot communicates with an offboard ROS master that publishes the latitude/longitude path plan. We use an accepting radius of approximately 1 meter. GPS precision could limit the natural language commands a real quadcopter understands, for example, references to nearby `landmarks` may require additional localization modules (e.g. vision). There is also the possibility of operating in GPS-denied environments. The robot could use a transformation between its' local and global frames, but this is susceptible to drift over long distances. Aerial robots capable of following long-range plans are likely subject to airspace rules that the planner would need to support. Simulation allows us to test end-to-end functionality of our approach; real-world deployment is a significant thread of future work.

V. Conclusion

We present a hierarchical map representation and planning approach that enables tractable planning of non-Markovian tasks in large outdoor environments. Our approach leverages abstractions from geographic data and a user's language to compact the state space and accelerate planning time. We test our approach on 7 templates of LTL$_f$ formulas in 30 and 80-kilometer-radius environments containing thousands of locations. The results show our approach plans in tractable time while obeying complex temporal goals and constraints. Further, the paths are similar or shorter-length compared to paths planned with a baseline approach. Future work includes integrating airspace rules in the planner and exploring more advanced environment filtering heuristics.

VI. Acknowledgements

This work was supported by ONR under grant award N00014-17-1-2699 and an award from Echo Labs.

References

[1] E. Ackerman and M. Koziol, "In the air with zipline's medical delivery drones," Available at https://spectrum.ieee.org/robotics/drones/in-the-air-with-ziplines-medical-delivery-drones, 2019.

[2] Skydio, "Company overview," Skydio, Inc., Tech. Rep., 2020. [Online]. Available: https://drive.google.com/file/d/13V4XcHudwhI61zyfz5L7eJ_BNvg0-GOV/view

[3] J. Hirsch, "Waymo tests autonomous trucks in texas," Available at https://www.ttnews.com/articles/waymo-tests-autonomous-trucks-texas, 2020.

[4] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 25–55, 2020.

[5] N. Gopalan, D. Arumugam, L. L. Wong, and S. Tellex, "Sequence-to-sequence language grounding of non-markovian task specifications." in *Robotics: Science and Systems*, 2018.

[6] Y. Oh, R. Patel, T. Nguyen, B. Huang, E. Pavlick, and S. Tellex, "Planning with state abstractions for non-markovian task specifications," in *Robotics: Science and Systems*, 2019.

[7] M. Berg, D. Bayazit, R. Mathew, A. Rotter-Aboyoun, E. Pavlick, and S. Tellex, "Grounding language to landmarks in arbitrary outdoor environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 208–215.

[8] R. Patel, E. Pavlick, and S. Tellex, "Grounding language to non-markovian tasks with no supervision of task specifications," in *Robotics: Science and Systems*, 2020.

[9] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ," https://www.openstreetmap.org, 2017.

[10] K. L. Lovelace, M. Hegarty, and D. R. Montello, "Elements of good route directions in familiar and unfamiliar environments," in *International conference on spatial information theory*. Springer, 1999, pp. 65–82.

[11] B. Kuipers, "The spatial semantic hierarchy," *Artificial Intelligence*, vol. 119, no. 1-2, pp. 191–233, 2000.

[12] C. Matuszek, D. Fox, and K. Koscher, "Following directions using statistical machine translation," in *5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2010, pp. 251–258.

[13] T. Kollar, S. Tellex, D. Roy, and N. Roy, "Toward understanding natural language directions," in *5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2010, pp. 259–266.

[14] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, 2011.

[15] A. Wandzel, Y. Oh, M. Fishman, N. Kumar, L. L. Wong, and S. Tellex, "Multi-object search using object-oriented pomdps," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 7194–7200.

[16] M. Ahmadi, R. Sharan, and J. W. Burdick, "Stochastic finite state control of pomdps with LTL specifications," *arXiv preprint arXiv:2001.07679*, 2020.

[17] K. Zheng, D. Bayazit, R. Mathew, E. Pavlick, and S. Tellex, "Spatial language understanding for object search in partially observed city-scale environments," in *30th IEEE International Conference on Robot Human Interactive Communication (RO-MAN)*, 2021, pp. 315–322.

[18] C. Bradley, A. Pacheck, G. J. Stein, S. Castro, H. Kress-Gazit, and N. Roy, "Learning and planning for temporally extended tasks in unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4830–4836.

[19] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5113–5120.

[20] R. Paul, J. Arkin, N. Roy, and T. M Howard, "Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators," in *Robotics: Science and Systems*, 2016.

[21] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, "Learning to parse natural language commands to a robot control system," in *Experimental robotics*. Springer, 2013, pp. 403–415.

[22] A. Boteanu, J. Arkin, S. Patki, T. Howard, and H. Kress-Gazit, "Robot-initiated specification repair through grounded language interaction," in *AAAI Fall Symposium on Natural Communication for Human-Robot Collaboration*, 2017.

[23] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Translating structured english to robot controllers," *Advanced Robotics*, vol. 22, no. 12, pp. 1343–1359, 2008.

[24] C. Wang, C. Ross, Y.-L. Kuo, B. Katz, and A. Barbu, "Learning a natural-language to LTL executable semantic parser for grounded robotics," in *4th Conference on Robot Learning (CoRL)*, 2020.

[25] M. Y. Vardi and P. Wolper, "Reasoning about infinite computations," *Information and Computation*, vol. 115, no. 1, pp. 1–37, 1994.

[26] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 2014, p. 3104–3112.

[27] M. L. Littman, U. Topcu, J. Fu, C. L. Isbell, M. Wen, and J. MacGlashan, "Environment-independent task specifications via GLTL," *ArXiv*, vol. abs/1704.04341,

2017.

[28] G. De Giacomo and M. Y. Vardi, “Linear temporal logic and linear dynamic logic on finite traces,” in *IJCAI’13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. Association for Computing Machinery, 2013, pp. 854–860.

[29] OpenStreetMap contributors, “Elements,” https://wiki.openstreetmap.org/wiki/Elements, November 2020.

[30] S. M. Omohundro, *Five Balltree Construction Algorithms*. International Computer Science Institute Berkeley, 1989.

[31] B. N. Delaunay, “Sur la sphère vide,” *Bull. Acad. Sci. URSS*, vol. 1934, no. 6, pp. 793–800, 1934.

[32] J. Inman, *Navigation and Nautical Astronomy for the Use of British Seamen*. C. and J.Rivington, 1835.

[33] OpenStreetMap contributors, “Map features,” https://wiki.openstreetmap.org/wiki/Map_features, February 2021.

[34] R. Bellman, *Dynamic Programming*, 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[35] A. Guttman, “R-trees: A dynamic index structure for spatial searching,” in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 1984, p. 47–57.

[36] Howard Butler, Brent Pedersen, Sean Gilles, and others, “Rtree: Spatial indexing for python.” [Online]. Available: https://toblerity.org/rtree/

[37] Unity Technologies, “Unity.” [Online]. Available: https://unity.com/

[38] Mapbox, “Mapbox unity SDK.” [Online]. Available: https://github.com/mapbox/mapbox-unity-sdk

[39] M. Quigley, J. Faust, T. Foote, and J. Leibs, “ROS: An open-source robot operating system,” in *IEEE International Conference on Robotics and Automation Workshop on Open Source Software*, 2009.

[40] Siemens, “*ROS#*,” 2017, https://github.com/siemens/ros-sharp, [Accessed: 2018].