From Natural Language to Long-Range Path Plans in Outdoor Environments

Matthew Oliver Berg

Department of Computer Science

Brown University

Advisor

Stefanie Tellex, Ph.D.

Reader

George Konidaris, Ph.D.

In partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science

April 7, 2021

Acknowledgements

My research and thesis advisor, Professor Stefanie Tellex, has provided invaluable guidance since I arrived at Brown. Stefanie, you have shaped my growth as a student, researcher, and roboticist. I am deeply grateful for your support.

My research advisor and thesis reader, Professor George Konidaris, has been generous with his time and wisdom. George, you have pushed my ideas in exciting directions and modeled how to write with elegance and precision.

Stefanie and George, it has been a privilege to be your student.

Part of this thesis is collaborative work with Deniz Bayazit, Rebecca Mathew, Ariel Rotter-Aboyoun, and co-advised by Professor Ellie Pavlick. A project with Peter Huson inspired my work on long-range planning. To all my collaborators and advisors: it has been a joy to work together.

My friends and mentors at Brown and home are always supportive of my aspirations. Thank you for everything you have shared with me.

Last, to my family: Mom, Dad, and Henry, your love and support make this life special. Thank you for being the source of so many wonderful blessings.

Abstract

As robots begin to operate in outdoor, large-scale environments, their control interfaces may need to understand more complex human inputs. This work presents language grounding and long-range planning systems for a simulated aerial robot operating in large outdoor environments. The language system generalizes to new environments without training and the planning system leverages environmental abstractions and filtering to compute plans in tractable time. Instructions such as *"go to Boston and go through the state forest on the way"* are translated into structured logical formulae and resolved to locations in the environment. Long-range goals like *"go to Boston"* are processed alongside finer-grained constraints like *"go through the state forest on the way"* by using a hierarchical representation of the environment and semantic utterances in users' language. Operation at multiple levels of abstraction is central to the planning system's tractable performance in city and state-scale environments. The language system demonstrates functional accuracy in an unseen environment and a user study rates the system as high performance and low workload. The planning system is evaluated on logical formulae specifying distances up to 80 kilometers and demonstrates tractable performance while obeying complex temporal goals and constraints. Both systems are tested on a simulated aerial robot to validate their functionality.

Contents

Readers note: Chapters 1, 2, 4, and 5 were advised by Stefanie Tellex and George Konidaris. Chapters 2 and 4 are currently under peer review and have been updated since submission. Chapter 3 is joint work with Deniz Bayazit, Rebecca Mathew, and Ariel Rotter-Aboyoun, was advised by Stefanie Tellex and Ellie Pavlick, and Sections 3.2-3.6 have appeared in conference proceedings (Berg et al., 2020).

Chapter 1

Introduction

Outdoor environments are an important frontier for robotics. Autonomous robots will drive on roads, deliver packages to doorsteps, survey construction sites, care for farmland, and more. Continuing advancements in autonomous systems are making these robots a reality, for example, farm robots, fixed-wing aerial robots, and quadcopters are currently used in commercial operations (Padwick, 2020, Ackerman and Koziol, 2019, Skydio, 2020).

Robots are developed to operate within a focused set of tasks. In turn, their input interfaces are limited to application-specific functions and levels of control granularity. However, as robots enter more challenging domains and perform increasingly sophisticated tasks, humans may prefer richer input interfaces. Potential solutions that build on existing interfaces, such as adding new functions or developing finer-grained control inputs coupled with training programs, could offer effective albeit shorter-term remedies; over time, the capabilities of some robots' autonomous systems will outpace the expressive power of existing interfaces.

Natural language is a highly expressive interface. From the user's standpoint, language interfaces can unlock new opportunities to collaborate with robots. The challenge for the robot is parsing an unrestricted and ambiguous input into structured signals for its autonomous systems. For example, in the command *"go to Boston and go through the state forest on the way,"* the robot must resolve *"Boston"* and the *"state forest"* to locations in the environment. The robot must distill a temporal ordering—visit a *"state forest"*, then visit *"Boston"*—and compute a plan in tractable time. These processes are technically feasible but computationally expensive. Expanding to outdoor environments further increases computational cost because the robot must understand a growing number of referring expressions and task constraints. As the environment grows large, natural language control interfaces can become computationally intractable.

This work presents language grounding and long-range planning systems for use in large outdoor environments. The language system translates natural language instructions into structured logical formulae and resolves unrestricted location referring expressions to real-world landmarks. This system understands instructions in new environments without further training. The planning system uses structured logical formulae and resolved referring expressions to generate a temporally correct plan. Planning performance is significantly accelerated by leveraging spatial abstractions in mapping data and semantic utterances in users' language. Both systems are tested on a simulated aerial robot to validate their functionality. The language system is additionally tested on a quadcopter operating in a real outdoor environment. Aerial robots are a compelling choice for initial work because they contend with fewer obstacles and (in simulated environments) experience no hard constraints on navigable areas.

The following chapters are organized as follows: Chapter 2 discusses background material and related work, Chapter 3 presents the language system, and Chapter 4 presents the environment representation and the planning system.

Chapter 2

Background and Related Work

This chapter introduces building blocks of the language and planning systems, then discusses related work on route instructions, language grounding, and planning.

2.1 Linear Temporal Logic

Linear Temporal Logical (LTL) encodes goals and constraints in the language command. LTL is a domain-independent formalism that supports temporal conditions with an infinite time horizon. By extension, LTL is capable of capturing non-Markovian goals and constraints. 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 formulae, $\mathcal{F}$ denotes *finally*, $\mathcal{G}$ denotes *globally*, and $\mathcal{U}$ denotes *until*. See Manna and Pnueli (1992) for semantic interpretations of LTL.

2.2 Planning Components

2.2.1 Deterministic Büchi Automaton

Every LTL formula has a corresponding deterministic Büchi automaton (DBA) (Büchi, 1990). DBAs handle infinite sequences of states, such as those encoded by LTL. In effect, DBAs provide a deterministic structure to non-Markovian goals and constraints, allowing the planner to process a stream of state-action pairs and compute whether or not the LTL formula has been satisfied. A DBA 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. In this work, the DBA is treated as a DFA. The planner considers only finite trajectories that satisfy the accepting condition.

2.2.2 Markov Decision Process

A Markov Decision Process (MDP) provides a model for sequential decision making in stochastic environments. An 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^*. The robot iteratively refines its policy to π^*. Convergence is guaranteed because the space of policies is finite and the robot iteratively improves its policy.

2.2.3 Labeled Markov Decision Process

A Labeled Markov Decision Process extends an MDP to consider atomic propositions of an LTL formula. A labeled MDP is defined by the tuple $M = (S, A, T, s_0, AP, L, R)$, where S, A, T, and R are the standard MDP components, s_0 is the initial state, AP is a set of atomic propositions, and $L : S \rightarrow 2^{AP}$ is a labeling function that maps states to the atomic propositions they satisfy.

2.2.4 Abstract Labeled Markov Decision Process

An Abstract Labeled Markov Decision Process (Oh et al., 2019) extends labeled MDPs to abstract state spaces. An AL-MDP is defined by the tuple $M^i = (S^i, A^i, T^i, s_0^i, AP, L^i, R^i)$, where i is the level of abstraction, S^i, A^i, T^i, s_0^i, and R^i are the labeled MDP components in abstraction level i, $L^i : S^i \rightarrow 2^{AP}$ is the labeling function, and AP is the set of atomic propositions. Atomic propositions can be defined at varied levels of abstraction. The state $s^i \in S^i$ is assessed against atomic propositions at level i or higher. An MDP M can be defined as the composition of AL-MDPs $\{M^0, \ldots, M^i, \ldots, M^l\}$, where l denotes the highest level of abstraction. This construction supports fluid planning at different levels of abstraction.

2.3 Related Work

Route directions can be ambiguous and refer to an immense number of paths in the environment. Lovelace et al. (1999) discuss a three-step model of generating route directions: spatial knowledge, choice of route, and translating the route into spoken instructions. The 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 for a robot (Kuipers, 2000) and following instructions in spaces containing a variety of environment features (Matuszek et al., 2010b, Kollar et al., 2010, Tellex et al., 2011a). Matuszek et al. (2010b) 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 that are available to the robot. Kollar et al. (2010) 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, unrestricted natural language commands can present non-sequential and non-Markovian goals and constraints.

The robot's planner requires a grounded representation of the language command. Recent work has grounded natural language to linear temporal logic (LTL) (Gopalan et al., 2018, Oh et al., 2019, Berg et al., 2020, Patel et al., 2020, Wang et al., 2020). LTL is a first-order logic capable of expressing non-Markovian goals and constraints. Importantly, every LTL formula has a corresponding deterministic automaton (Büchi, 1990), which the planner follows to obey the task specification. Gopalan et al. (2018) presents a sequence-to-sequence model (Sutskever et al., 2014) for grounding natural language to a variant of LTL (Littman et al., 2017). While users are restricted to location referring expressions seen during training, this work facilitates the development of planning systems that obey non-Markovian language commands. The language system introduced in Chapter 3 (Berg et al., 2020) handles unbounded location references, allowing the robot to generalize to unseen outdoor environments. However, these environments are not defined at multiple levels of abstraction. Chapter 4 adapts location reference resolution to large outdoor environments defined at multiple levels of abstraction. The environment map is searched for name-based matches (eg, *"Boston"*) at higher levels of abstraction, and name and semantic-based matches (eg, *"state forests"*) at the base level of abstraction.

Markov Decision Processes (MDPs) have been used for planning non-Markovian tasks specified by natural language commands (Gopalan et al., 2018, Oh et al., 2019, Berg et al., 2020, Patel et al., 2020). MDPs are a convenient choice for the planning problem because they are compatible with intermediate representations of the user's language command. However, MDPs struggle to tractably operate in large state spaces. In this vein, Oh et al. (2019) introduces the Abstract-Product Markov Decision Process (AP-MDP). AP-MDPs are a framework for planning non-Markovian tasks at different levels of abstraction. Planning at multiple levels of abstraction offers considerable reductions in computing time and the number of backups (Oh et al., 2019). The planning system (Chapter 4) is inspired by AP-MDPs but its formulation is modified and it uses a different environment representation. In particular, the planning system uses A* (Hart et al., 1968) to induce pruned, subtask-specific state spaces. Long-range planning operates at a higher level of abstraction than the robot's metric map, where retaining the majority of the state space is critical. There is a rich body of work on hybrid representations that combine low-level and high-level maps (Thrun, 1998, Poncela et al., 2002, Kuipers et al., 2004, Kuric et al., 2017). This work focuses on higher-level non-Markovian path planning, and assumes the robot contains lower-level autonomous systems capable of responding to local obstacles and changes in the environment.

Chapter 3

Grounding Language to Landmarks in Arbitrary Outdoor Environments

3.1 Introduction

Natural language can be temporally complex and contain unrestricted references. Consider the command *"go to the red bridge but first stop at CVS."* This command references the *"red bridge"* before *"CVS,"* but specifies that CVS should be visited first. One location is referenced by proper name (*"CVS"*) and the other by semantic descriptors (*"red bridge"*). The robot must translate temporal objectives into a structured form, resolve location referring expressions to real-world landmarks, and integrate this information to support lower-level operations such as planning. There has been work on translating language to structured forms that assumes the robot can be trained on its environment map (Tellex et al., 2011b, Artzi and Zettlemoyer, 2013, Paul et al., 2018, Oh et al., 2019). However, as the robot enters large outdoor environments, it would be computationally infeasible to pre-train on all possible locations and referring expressions. There is a need for a system that combines a neural model supporting the globally consistent task—inferring temporal structure—with a lower-cost, generalizable procedure for the location-dependent task—resolving location references to real-world landmarks. Taken together, the system should allow the robot to understand natural language commands in new outdoor environments without further training.

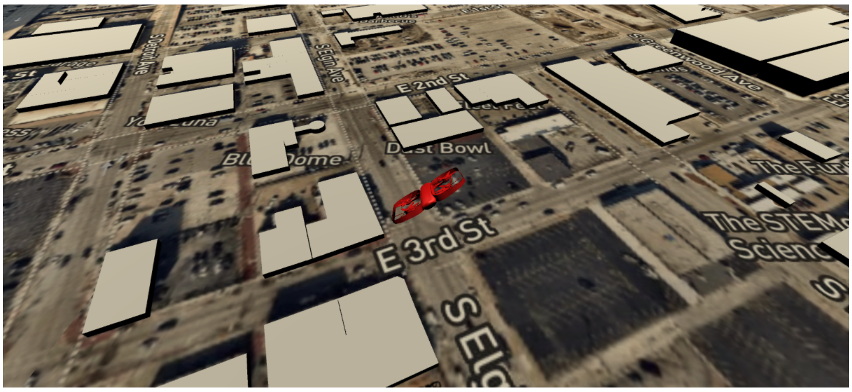

Figure 3.1 Simulated Skydio R1 in Tulsa, Oklahoma. This map was not shown during training and the end-to-end system succeeds at performing 76.19% of the tested natural language commands in this environment.

This chapter presents a model that grounds natural language to Linear Temporal Logic (LTL) and resolves unrestricted referring expressions to real-world landmarks. The model understands temporally complex natural language instructions and generalizes to new environments without further training. By combining the language model and an Abstract Product Markov Decision Process-based planning model (Oh et al., 2019), we establish an end-to-end framework that takes natural language instructions as input and returns motion plans as output. The language model was evaluated on a corpus of 1540 natural language instructions referring to locations in 22 cities. In addition, we used the end-to-end framework coupled with a simulated aerial robot, shown in Figure 3.1, to conduct a user study. Last, we tested the framework on a Skydio R1 quadcopter in a park near Brown's campus. Our results demonstrate functionality, successful generalization to new environments, and a high-performance, low-workload user experience.

3.2 Overview of the Approach

Our system allows a person to command a drone with natural language in a never-before-seen environment. The system can interpret natural language commands, including references to nearby landmarks, with no training data for the environment. A graphical representation of our system is shown in Figure 3.2.

The language model grounds natural language commands to LTL formulae. The LTL structure is created by CopyNet (Gu et al., 2016), a Seq2Seq model capable of copy-

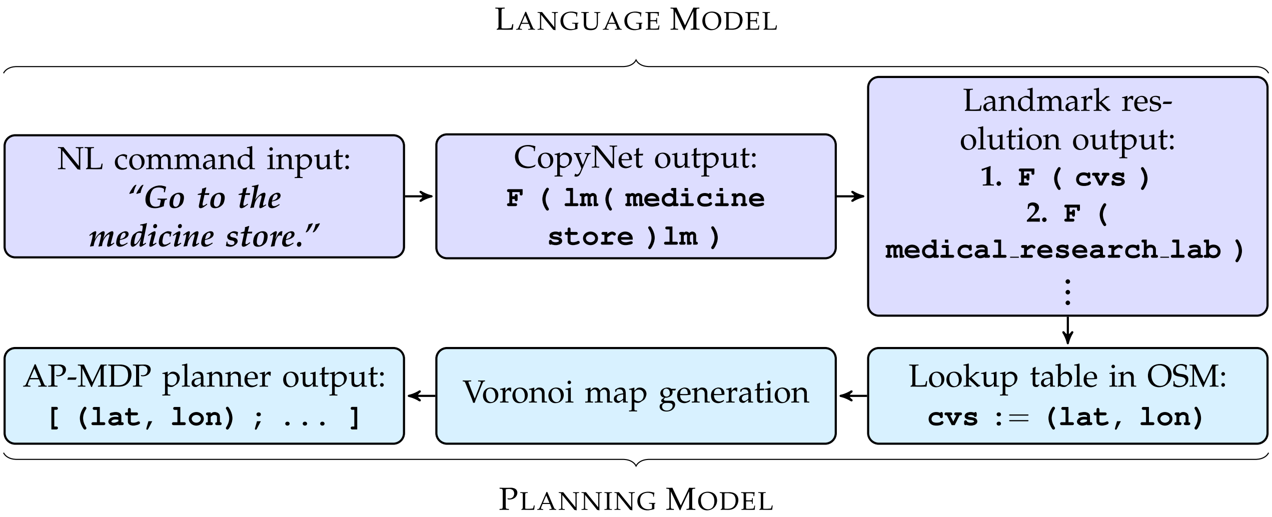

Figure 3.2 End-to-End System Pipeline. Natural language is given to the language model, which returns a grounded LTL formula. The planning model then creates a motion plan which satisfies the LTL formula.

ing out of vocabulary (OOV) words. To ground natural language landmark referring expressions to landmarks in a map unseen to the language model, we use a resolution model that draws semantic information from a mapping database. The final output of the language model is an LTL formula with natural language in the logical form, e.g. $\mathcal{F}(CVS \land \mathcal{F}(red\ bridge))$.

The LTL formula is then passed to the planning model. The planning model uses a map generated from OSM, partitioned into Voronoi cells (Voronoi, 1908). The partitioned map along with the LTL formula are supplied to the AP-MDP planner (Oh et al., 2019). This planner extracts goals and constraints from the LTL formula to create a motion plan as a series of latitude and longitude points.

3.3 CopyNet

To translate natural language commands into logical forms, current approaches use a Seq2Seq model (Oh et al., 2019, Gopalan et al., 2018, Dong and Lapata, 2016). Seq2Seq models learn how to translate input sequences into output sequences. However, existing Seq2Seq models learn a mapping from a fixed input language to a fixed output language, and require all symbols in the output language to have appeared at training time. In contrast, our language model generalizes to any region, and thus needs the ability to understand words and commands the language model has not been trained on. In particular, it is essential that we extract unseen landmark referring expressions from the natural language command. For example, given the command *"go to the medicine store,"* our model needs to correctly identify that *"medicine store"* is the referring expression and the corresponding LTL formula would be $\mathcal{F}(\textit{medicine store})$.

We approach this challenge with CopyNet (Gu et al., 2016), which is developed for cases when the output contains many subsequences from the input. CopyNet introduces a copy-attention mechanism atop the traditional Seq2Seq framework (Bahdanau et al., 2014). This copy mechanism is fundamental to our language model, allowing for a more domain-general model even with a small training set.

When comparing CopyNet to a purely generative recurrent neural network with the LCSTS dataset (Hu et al., 2015), Gu et al. demonstrates that CopyNet improves production of readable output for out-of-vocabulary (OOV) words. We selected CopyNet because it was accessible in multiple open-source implementations. We use Adam Klezcweski's implementation of CopyNet[1] with the addition of pre-trained GloVe embedding vectors (Pennington et al., 2014). We use mjc92's dataset[2] to validate Klezcweski's model.

To train our model, we use a corpus of 668 natural language navigation instructions collected by Oh et al. (2019). Each command has a corresponding LTL formula, making this dataset well-suited for training a Seq2Seq model like CopyNet. We augment the data by replacing goal locations with Brown campus landmark names scraped from OSM. We then divide these landmarks into unique datasets containing landmarks from north campus and south campus. In addition, we wrap references to landmarks with `lm(` and `)lm` as shown in step two of Figure 3.2, simplifying extraction of landmark referring expressions for the landmark resolution model. Finally, we limit the dataset to the following three LTL structures:

$$\mathcal{F}(\phi) \mid \mathcal{F}(\phi \wedge \mathcal{F}(\psi)) \mid \mathcal{F}(\phi \wedge \neg\psi)$$

[1]https://github.com/adamklec/copynet
[2]https://github.com/mjc92/CopyNet

3.4 Landmark Resolution Model

3.4.1 Mapping Database

A key focus of our framework is the language model that grounds language to landmarks, as humans find landmarks important for navigation instructions, particularly for unfamiliar environments (Lovelace et al., 1999). Landmarks are geographic objects important to human spatial cognition (Richter and Winter, 2014). Following previous work (Rousell et al., 2015, Drager and Koller, 2012) we use OSM as our landmark database.

OSM is a global open-source map where any user can add landmarks and information about the landmarks. Critically, this information can be semantic in nature, such as the type of cuisine for a restaurant or the function of a building. We leverage OSM's extensive semantic database as the foundation of our language model, enabling groundings of semantic referring expressions to landmarks.

Two building blocks of the OSM database are nodes and ways. Nodes are points with a latitude, longitude, and unique numerical ID. Nodes commonly represent landmarks such as statues, benches, and trees. Ways are lists of nodes, commonly representing larger landmarks like buildings, roads, and greens. Closed ways have a polygon geometry. Both nodes and ways can be tagged with key-value pairs about their appearances, functions, or other semantic information.

3.4.2 Landmark Resolution Model

Given all the possible landmark candidates in the map, the model needs to resolve the user's referring expression to the correct landmark. The landmark resolution model finds the maximally probable candidate by calculating the similarities between the referring expression and each landmark's semantic information.

The landmark resolution model receives the CopyNet output of an LTL formula with the user's referring expression. While any arbitrary model could resolve this expression given textual descriptions, images, or robot sensor data, we present a model that uses word embeddings to resolve the user's referring expression to the landmark name.

The model uses the database's semantic information about each landmark to find the intended landmark. However, the user's referring expression may not lexically align with the landmark database. For example, we would expect "store" and "shop" to have similar meaning, even if OSM's data model only supports `key:shop`. To resolve these lexical conflicts, we use word embeddings, which represent words or phrases as vectors in a high-dimensional vector space (Pennington et al., 2014, Mikolov et al., 2013, Bojanowski et al., 2016, Joulin et al., 2017, Grave et al., 2018). High-dimensionality allows us to use cosine similarity (the cosine of the angle between vectors) to compare semantic referring expressions.

A referring expression may fall into one or more of three possible categories: name, address, and general description. An example of a command using more than one category would be *"fly to CVS pharmacy,"* which includes name and description.

name: Our model exclusively uses the OSM key `name`.

address: Our model exclusively uses `addr:house number` and `addr:street`.

descriptions: Our model uses keys we observed to be semantically significant in natural language commands, such as `amenity`, `shop`, and `leisure`.

For each category we gather the key values into lists. Then, to handle multiple categories of values, we create all possible combinations of these lists. For each combination, we compute the average of their word vectors. We then calculate the cosine distance between each of these averaged vectors and the phrase vector for the referring expression. Finally, we use the minimum cosine distance to identify the referred landmark. We evaluate this approach against other models in Section 3.6.2.2. The cosine distance between two vectors is defined as the difference between one and their cosine similarity.

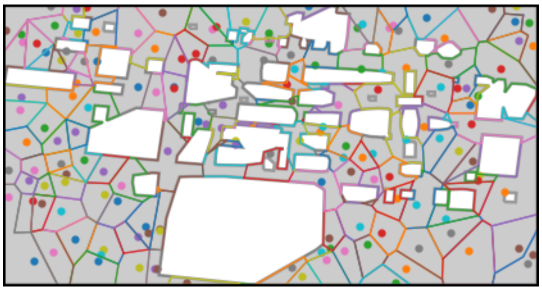

Figure 3.3 Map partitioned into Voronoi cells. White holes represent regions containing landmarks.

3.5 Voronoi Maps and Planning

We use the AP-MDP planner to convert grounded LTL formulae to high-level motion plans, and leave lower-level motion planning to the drone's autonomy system. Oh et al. (2019) partitions a hard-coded map into a grid of flyable zones and target landmarks. However, since other real-world geometries can be large and complex, a more flexible approach to map partitioning is required.

Our approach uses Voronoi cells (Voronoi, 1908). We query OSM for landmarks in a 300 meter radius square around a center point, creating holes for each way polygon and five meter radius square holes around each node. Then, we randomly generate points inside the solid region, which are used to partition the map into Voronoi cells as shown in Figure 3.3. We have observed the Voronoi cells can enable faster planning over large distances. When comparing our results in the predefined map by Oh et al. (2019), Voronoi-based planning between two landmarks 48.28 meters apart ran in 37.08 $\pm$ 6.43 seconds, whereas the grid-based approach ran in 90.49 $\pm$ 0.27 seconds (over three runs). Further, the AP-MDP planner understands landmarks as a single latitude and longitude coordinate, not a polygon. As such, we represent ways in the planner by choosing one corner node as its representative point.

To align with limitations of both natural language and our framework, we filter certain landmarks. Landmarks need to be named for the purposes of natural language commands, so they must have a `key:name`. We exclude any landmark containing the key `highway`, `railway`, `place`, `boundary`, or `waterway`, because it is difficult to use a singular representative point for very large landmarks.

	Percentage (%)
Speech-to-text errors	4.76
Incorrect grounding (Landmark Resolution)	2.38
Planner errors	4.76
Improper LTL (CopyNet)	11.90
Succeeded	**76.19**

Table 3.1: System performance accuracy for in-person user evaluation

	Raw NASA-TLX (pts)
Performance	14.85 ± 05.38
Mental demand	03.50 ± 02.42
Physical demand	02.83 ± 04.49
Temporal demand	01.50 ± 01.50
Effort	03.40 ± 03.17
Frustration	05.50 ± 05.08

Table 3.2: Raw NASA-TLX scores on a 20 point scale

3.6 Evaluation

We test that our system accurately grounds natural language commands with references to landmarks, without being trained on those landmarks. We conduct an end-to-end user evaluation where participants give natural language commands to the drone and observe the robot's actions in simulation. In addition, we perform a corpus-based evaluation on a diverse set of maps to test the limits of our framework. Finally, we demonstrate the system acting in a real outdoor domain[1].

3.6.1 User Evaluation

To test end-to-end performance on a map unseen to the language model during training, we ran an in-person user evaluation with 14 voluntary student participants. Each student gave three spoken natural language commands to our system and evaluated the resulting behavior of a Skydio R1 drone in a simulated outdoor map of Tulsa, Oklahoma.

The simulator is built in Unity (Unity Technologies), using outdoor environments generated with the Mapbox SDK (Mapbox) (Figure 3.1). Using ROS and ROS# (Quigley et al., 2009, Siemens, 2017), the simulator and planner communicate about the drone's

[1] `https://youtu.be/a-JGems7fzs`

flight status and flight trajectories. The simulator allows the participant to view the trajectory the drone takes given the participant's natural language command.

As shown in Table 3.1, our model accurately grounds natural language commands to LTL and formed correct motion plans for 76.19% of user commands. In this table, we also break down failure cases. We observe challenges with two forms of natural language commands: commands that include spatial language, such as *"go to* l_1 *near* l_2*"*; or commands with verbs or unexpectedly long phrases that CopyNet has not been sufficiently trained on. Spatial language phrases cause CopyNet to not copy enough words, resulting in improper groundings or improper LTL structures. We hypothesize that CopyNet failures are due to the limited use of spatial language in CopyNet's training dataset, and that a more representative training dataset would address these problems. Also, planner errors were due to an indexing bug that we resolved post-evaluation.

After using our system, users answered the NASA-TLX questionnaire to measure workload on a scale of 0 to 20 (least to most) (Hart and Staveland, 1988). On average, users reported high performance and low workload (Table 3.2). Additionally, we use the Systems Usability Scale (SUS) (Brooke, 1996) to understand system ease of use. We report a mean SUS score of 76.25 with standard deviation of 18.39, which is above the average SUS score of 68 (Sauro, 2011).

3.6.2 Component Evaluation

We analyze the performance of individual components of our language pipeline to understand failure modes and potential improvements to our end-to-end system.

3.6.2.1 CopyNet Evaluation

We trained two models to evaluate CopyNet. The first is trained on natural language commands with a single landmark, the second on natural language commands with two landmarks. We trained with a learning rate of 0.001 over 8 epochs for the single landmark model and 15 epochs for the two landmark model. The models were then evaluated against phrases with seen and unseen landmarks as shown in Table 3.3.

Number of Landmarks	Seen (%)	1 Seen, 1 Unseen (%)	Unseen (%)
One	100.00 ± 0.00	N/A	74.50 ± 2.88
Two	99.48 ± 0.20	69.18 ± 2.52	53.49 ± 2.95

Table 3.3: CopyNet accuracy

		Name	Uniform	tf-idf	Our Model
Accuracy (%)	fastText	41.86	43.41	51.94	**58.14**
	Word2Vec	42.64	45.74	54.26	**58.91**
	Glove840B	44.96	48.06	55.81	**68.99**
MRR (%)	fastText	47.72	61.73	49.15	**66.84**
	Word2Vec	51.22	63.51	54.38	**68.97**
	Glove840B	51.39	65.22	54.38	**76.35**

Table 3.4: Landmark grounding accuracy and MRR results for different landmark resolution and word embedding models

For two landmark commands, we observe on average that CopyNet grounds 69.18% of commands containing one unseen landmark and 53.49% of commands containing two unseen landmarks to the correct LTL structure (Table 3.3). CopyNet errors are principally attributed to not copying enough words from input to output.

3.6.2.2 Landmark Resolution Evaluation

We compare our landmark resolution model to other models, as shown in Table 3.4. We create the following baselines to evaluate the effectiveness of our landmark resolution model. The Name model represents a landmark by just its name phrase vector, an average of word embedding vectors for every word in its name. The Uniform model represents a landmark by assigning equal weight to every OSM semantic feature (including the name of the landmark) and averages their phrase vectors. The term frequency-inverse document frequency (tf-idf) (Sammut and Webb, 2010) model weighs each semantic feature's phrase vector with its tf-idf score, a metric to down-weigh frequent or uninformative words by document, where each map is a document. All models use minimum cosine distance to identify the referred landmark.

Landmark names often contain proper nouns, which may be OOV. We evaluate if using morphological information (e.g. prefixes, suffixes, roots, etc.) helps the model process OOV words by comparing fastText (Bojanowski et al., 2016, Grave et al., 2018), which uses such information, to larger word embedding models like Word2Vec and GloVe (Mikolov et al., 2013, Pennington et al., 2014).

We evaluate on 129 references collected from seven researchers in the Brown University Humans to Robots Lab. We showed each person OSM landmark information from a single map and asked for different landmark referring expressions by type(s): name, address, and description.

We define the grounding accuracy to be the percentage of landmarks returned by our language model that matches the intended reference. We calculate grounding accuracy and mean reciprocal rank (MRR) of every landmark resolution model and word embedding combination. MRR is defined as the average of the reciprocal rank scores across multiple queries. The reciprocal rank score of a query (a user's semantic reference) is the multiplicative inverse of the correct landmark's ranking. For example, if the landmark resolution model ranks the true landmark corresponding to a user's semantic reference as third-most likely, the reciprocal rank would be $1/3$ (assuming the list is three landmarks long). Table 3.4 shows that our landmark resolution model performs best with GloVe, which we attribute to its large vocabulary.

3.6.3 Corpus-Based Evaluation

We test our language model's ability to both identify the appropriate LTL structure and properly extract landmarks from unseen commands by collecting a test set of challenging natural language commands from AMT. We collected commands for 22 urban American regions. (Table 3.5).

AMT workers viewed a screenshot of a region in OSM with an overlaid trajectory (Figure 3.4). Trajectories allow us to ask AMT workers for natural language commands without extensive language prompting. At the start of each task, AMT workers saw an example map and related example commands. We further provided a detailed task description to ensure AMT workers responded with high-level commands, not low-level, action-oriented instructions. Every AMT worker was given semantic information about each landmark to allow for flexibility in landmark referring expressions. We provided Google search cards without the landmark's address as to not bias the

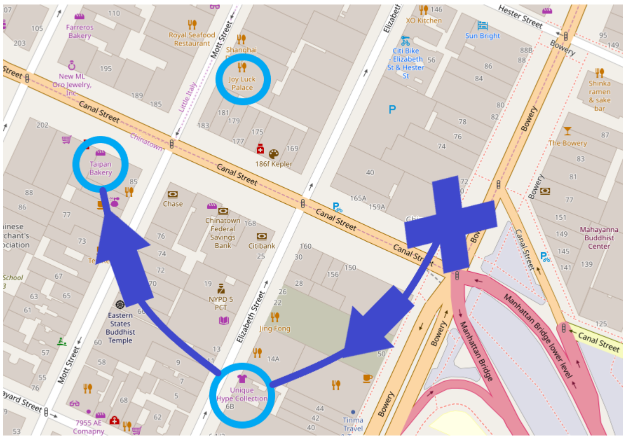

Figure 3.4 AMT trajectory example. An OSM region with trajectory that corresponds to $\mathcal{F}(\text{lm}(l_1)\text{lm} \wedge \mathcal{F}(\text{lm}(l_2)\text{lm}))$

AMT workers with OSM semantic data. We have published 1540 collected commands, each formed by a unique AMT worker. Compensation was $0.50 per task.

City Name	Number of Landmarks	Accuracy (%)
Jacksonville #2	16	17.14
Boston	39	20.00
New York #1	71	30.00
Chicago #2	26	35.71
Charlotte #1	24	35.71
Seattle	119	37.14
Denver #1	27	40.00
Philadelphia #1	21	44.29
Indianapolis	10	45.71
Denver #2	21	45.71
Jacksonville #1	19	47.14
Los Angeles #1	60	48.57
Los Angeles #2	62	52.86
Columbus #2	26	52.86
Chicago #1	22	54.29
Houston	32	54.29
New York #2	73	54.29
Philadelphia #2	90	55.71
San Diego #1	41	55.71
San Diego #2	31	55.71
Charlotte #2	15	57.14
Columbus #1	10	70.00
Average	**38.86**	**45.91 $\pm$ 12.70**

Table 3.5: Corpus-based language pipeline accuracy

We achieve a 45.91% mean accuracy of grounding natural language to correct fully-formed LTL. Some inaccuracies in the corpus-based evaluation may be due to unclear AMT instructions, which would lead to incorrect AMT worker annotations.

Chapter 4

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

4.1 Introduction

Robots are increasingly deployed to outdoor domains for autonomous missions: fixed-wing drones are delivering critical medical goods (Ackerman and Koziol, 2019), quadcopters are surveying infrastructure and land (Skydio, 2020), and autonomous trucks are being tested on public roads (Hirsch, 2020). 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 need an interface that allows a human to give commands to a robot in a natural way, while exploiting all of the knowledge that may be present in such commands.

Natural language offers an intuitive and expressive interface for human-robot interaction. There is a substantial body of work on resolving natural language to plans for a robot (Tellex et al., 2020). Motivated by the complex temporal goals and constraints often expressed in natural language, recent work has focused on developing models to handle non-Markovian commands (Gopalan et al., 2018, Oh et al., 2019, Berg

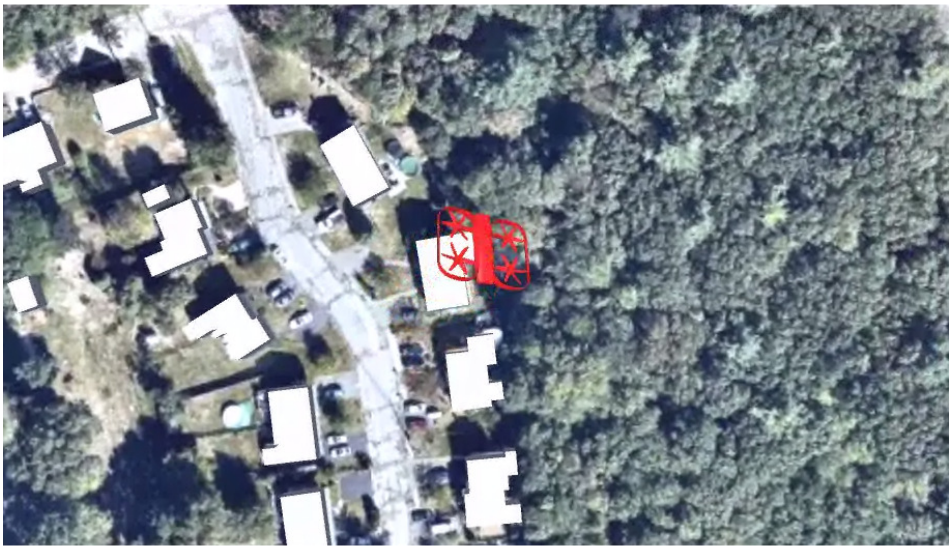

Figure 4.1 Simulated Skydio R1 flying over Massachusetts, USA and following the task *"go to Boston and go through the state forest on the way."*

et al., 2020, Patel et al., 2020). These approaches leverage a sequential decision-making framework that supports non-Markovian objectives, but is computationally expensive due to the necessary processing of extended state histories. For example, to follow the command *"go to Boston and go through the state forest on the way,"* the robot must evaluate its state history to ensure a visit to a state forest followed by a visit to Boston. As the robot's environment grows large, it becomes increasingly difficult to rapidly plan extended state sequences.

This chapter presents a system that combines abstraction and environment filtering to improve planning performance on non-Markovian commands in large outdoor environments. To achieve faster operation, the system dynamically reduces the planner's state space as it progresses through a series of decomposed subtasks. The system is evaluated on LTL formulae spanning seven temporal structures and specified distances up to 80 kilometers. Results show significant performance gains while obeying complex temporal goals and constraints. In addition, the system is tested on a simulated aerial robot, shown in Figure 4.1, to demonstrate its functionality. These results demonstrate more tractable planning of non-Markovian commands in city and state-scale outdoor environments.

4.2 Overview of the Approach

The planner takes a structured representation of the user's language command and outputs a sequence of coordinates in the global frame. First, the system generates a hierarchical map of an aerial robot's outdoor environment from publicly available mapping data. Second, the system initiates planning using an LTL encoding of the language command and an Abstract Process Markov Decision Process-based planner (Oh et al., 2019) with a reward function, abstraction dynamics, and lower-level planning approach adapted to the unique constraints of large outdoor state spaces. Third, as the system's planner transitions through states of the LTL formula, it leverages semantic similarities between the user's language and the environment's map features to induce a filtered representation of the environment containing a constrained set of paths for the planner to consider.

4.3 Modeling an Outdoor Environment

The environment is a multi-level graph with a user-definable radius. We test with a map radius of 80 kilometers. Environment features are downloaded from OpenStreetMap (OpenStreetMap contributors, 2017), a publicly available mapping service. 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 (OpenStreetMap contributors, 2020). Our environment's abstraction hierarchy flows from the geometry and semantic data of these elements. At the base level, we retain `landmarks`, corresponding to named nodes and ways. This level contains features such as buildings, parks, and streets. The intermediate level, `neighborhoods`, contains named nodes and ways that are tagged as neighborhoods. The highest level, `cities`, corresponds to relations at administrative level 8. The aerial robot builds a multi-level environment to accelerate planning performance and represent users' cognition of large spaces. A visualization of the environment is shown in Figure 4.2.

In our problem space, states and actions are defined by geometries and edges in the environment. Consider the example task *"go to Boston and go through the state forest on the way."* The state for Boston is an irregular polygon representing the city

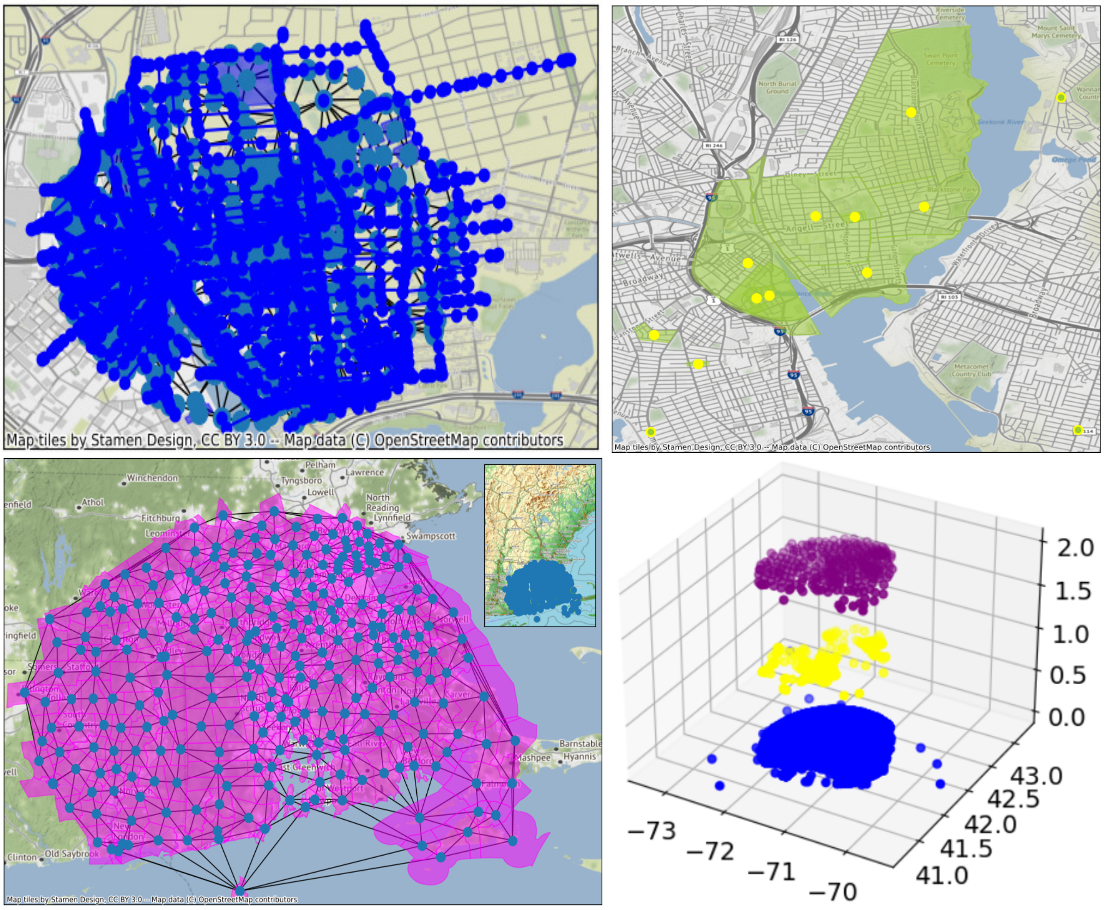

Figure 4.2 Visualization of the environment. The center is Brown University, Providence, RI, USA. The `landmark` (top left, partial view), `neighborhood` (top right, partial view), `city` (bottom left) levels, and hierarchical map (bottom right) are shown.

limits. The aerial robot takes action a to reach Boston, where a identifies the edge connecting the current state and Boston. As the number of states and actions grows large, MDP-based planners can struggle to tractably operate. Performance limitations are especially pronounced at the base `landmarks` level. We use the Abstract Product Markov Decision Process (Oh et al., 2019) to help overcome these limitations. An AP-MDP takes the product of AL-MDPs and the DBA $\mathcal{B}$. This construction correlates transitions in the environment with transitions of an LTL formula's state. Importantly, each transition in $\mathcal{B}$ is solved by an AL-MDP at the minimally satisfiable level of abstraction. For example, going through the state forest is solved by a `landmark`-level AL-MDP, while going to Boston is solved by a `city`-level AL-MDP. Planning over different levels of abstraction leads to considerable performance and efficiency gains.

4.4 Map Generation

In our problem space, the aerial robot receives a non-Markovian natural language command referring to a long-range path. The aerial robot must then resolve the language command to a structured form for the planner, and the planner must resolve the structured form to a path for the aerial robot. Essential to both challenges is constructing a map representation that captures mapping features necessary for grounding the user's language while remaining sufficiently compact for efficient planning.

Our map construction, shown in Figure 4.2, achieves these goals through abstraction. We define the outdoor map as the tuple $MP = (G, H, T, Z)$, where:

1. G is the graph set, centered on a single global coordinate. In the example task, this coordinate is located in Providence, RI, USA. $g^i \in G$ is the graph at abstraction level i. Level 0 corresponds to `landmarks` (Figure 4.2, top left), level 1 to `neighborhoods` (Figure 4.2, top right), and level 2 to `cities` (Figure 4.2, bottom left).

2. H is the hierarchical map composed from all $g^i \in G$ (Figure 4.2, bottom right). This graph merges the `landmark`, `neighborhood`, and `city`-level graphs into a unified environment map.

3. T is the transition map composed from $t^i \in T$ Ball trees (Omohundro, 1989). The aerial robot queries T to transition between levels of abstraction. In the example task, the aerial robot queries t^2 to transition to the `city` level before planning a path to Boston.

4. Z is the filtered graph set (Figure 4.5). Each $\zeta^i_{(q_j, q_{j+1})} \in Z$ is a sub-graph of g^i containing the dynamically filtered environment view for the subtask (q_j, q_{j+1}). In the example task, $\zeta^0_{q_0,q_1}$ contains `landmark`-level paths between Providence and state forests, $\zeta^2_{q_1,q_2}$ contains `city`-level paths between a state forest and Boston.

Mapping features from OpenStreetMap contain rich local geometric and global hierarchical data that we leverage in the construction of *MP*. At the local level, mapping features commonly contain line or polygon geometries that are unnecessary for language resolution. Therefore, we construct each g^i as the Delaunay triangulation (Delaunay, 1934) of the centroids[1] of mapping features, and separately retain geometric data for planning. At the global level, the merged map H is composed from $g^i \in G$ following the `landmark`, `neighborhood`, and `city` abstraction hierarchy. H stores the mappings from child to parent features, as calculated by the containment of and/or intersection between map feature geometries at different levels of abstraction.

As the aerial robot is operating in a large map, it becomes space-inefficient to create edges between child and parent map features. The transition map, T, alleviates space constraints by calculating transitions between sub-graphs at runtime. Each $t^i \in T$ is constructed from the centroids of the corresponding g^i using the Haversine distance metric (Inman, 1835). When the aerial robot transitions between layers of abstraction, it queries T for the nearest centroid at the new level of abstraction. Last, the filtered map set Z leverages semantic data from OpenStreetMap to induce subtask-specific views of the environment. Mapping features can contain a wide variety of textual semantic data, such as building type, waterway type, and land use (OpenStreetMap contributors, 2021). We use this semantic data to prune centroids and edges that are peripheral to the current subtask. In the example task, a `landmark`-level filtered graph contains paths to state forests and a `city`-level filtered graph contains a path to Boston. The construction of these graphs are discussed in Section 4.6.

4.5 Planning

We use a planning system inspired by the AP-MDP framework (Oh et al., 2019) to resolve an LTL formula to a plan for the aerial robot. We summarize an AP-MDP as the tuple $M^J = (S_p^J, A^J, T_p^J, s_{0p}^J, AP, L_p, Q, R_p^J)$, where $j \in J$ is the level of abstraction, $S_p^J = S^J \times Q$ is the set of product states, A^J is the set of actions, $T_p^J : S_p^J \times A^J \times S_p^{J'} \to S_p^J$ is the product transition function, s_{0p}^J is the product start state, AP is the set of atomic propositions in the LTL formula, $L_p : S_p^J \to Q$ is the product labeling function, Q is the set of automaton states, and $R_p^J : S_p^J \times A^J \times S_p^{J'} \to \mathbb{R}$ is the product reward function. An AP-MDP decomposes the deterministic Büchi automaton $\mathcal{B}$ into n_p problems,

[1]At the `landmark` and `neighborhood` level, we use the geometric centroid. At the `city` level, we use the administrative center (e.g., a town hall) when available in the mapping data; otherwise, we use the geometric centroid.

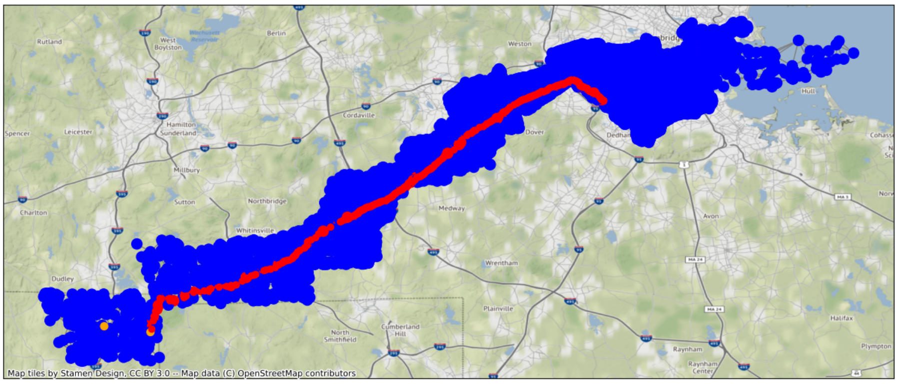

Figure 4.3 A* planning within a `landmark`-level projection of subtask (q_1, q_2). Blue points denote `landmark`-level map feature centroids within the projection, red points denote the `landmark`-level path, and orange points denote where a `landmark-city` transition is made during planning. The path starts at Quaddick State Forest, Thompson, Connecticut and completes within Millennium Park, Suffolk County, Boston, Massachusetts.

where n_p is the number of paths from the start state to the accepting state. We show the example task's DBA in Figure 4.4. Each subproblem is further decomposed into n_i subproblems, which we refer to as subtasks and denote as (q_j, q_{j+1}). A subtask represents a transition between states in $\mathcal{B}$. Each subtask is solved using an AL-MDP M_j, where j is the lowest level of abstraction across the stay condition (logical constraint on $q_j \rightarrow q_j$) and goal condition (logical constraint on $q_j \rightarrow q_{j+1}$) of the subtask. The action-minimizing path along $\mathcal{B}$ satisfying F is accepted as the solution. We refer the reader to Oh et al. (2019) for complete details.

We modify the AP-MDP framework in three ways. First, our reward function considers the sequence of states between the current and next state that are crossed but not visited. Second, abstract transitions are handled through the transition map T. Third, the AL-MDPs are configured to exclusively plan at the highest-possible level of abstraction, forgoing the base-level environment MDP in favor of A* path planning, when possible. The first change enables compatibility with our environment construction, the second and third changes reduce computational overhead.

The environment can contain discontiguous connections. For example, the `city`-level map in Figure 4.2 contains an edge between Pawtucket, RI and Seekonk, MA that crosses the tip of East Providence, RI. We adapt the reward function to handle these state crossings. At a high level, we ensure the current subtask's stay condition is satisfied along all states preceding the next state $s_p^{i'}$. Assuming the stay condition holds, the aerial robot receives its reward based on $s_p^{i'}$. Let a^i be the current action, e be the edge that is traversed to complete action a^i, and V be the set of states mapping to atomic propositions in the stay and goal constraints that intersect with e. The reward r is assigned as:

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

Each subtask can start at a different level of abstraction than the previous subtask. For example, the subtask from the aerial robot's start location to the state forest is planned at the `landmark` level, while the subtask from the state forest to Boston is planned at the `city` level. Transitioning the abstraction hierarchy becomes expensive as the environment grows large, due to the space-inefficiency of storing edges between child and parent map features. The aerial robot uses the transition map T to transition from the `landmark` to `city` level of the hierarchical map H. First, the aerial robot performs a k-nearest neighbors query (we use $k = 5$) on the Ball tree t^2 for `cities` near its global coordinate. Second, the neighbor list is refined to parent `city`(ies) of the state forest. Third, the aerial robot selects the neighbor closest to its current location, records the transition point and edge in t^2, and moves into the `city` level of H. The transition process is the same for moving down the abstraction hierarchy, except the neighbor list is refined by child map features of the aerial robot's current location.

The planner produces a sequence of states and actions (s_{seq}, a_{seq}) corresponding to vertices and edges in H. In the example task, the state sequence contains two subsequences: a series of `landmarks` satisfying subtask (q_0, q_1) and a series of `cities` satisfying subtask (q_1, q_2). The state sequence satisfies the LTL formula, but it does not specify a path at the base level of the environment. To calculate a base-level plan, the

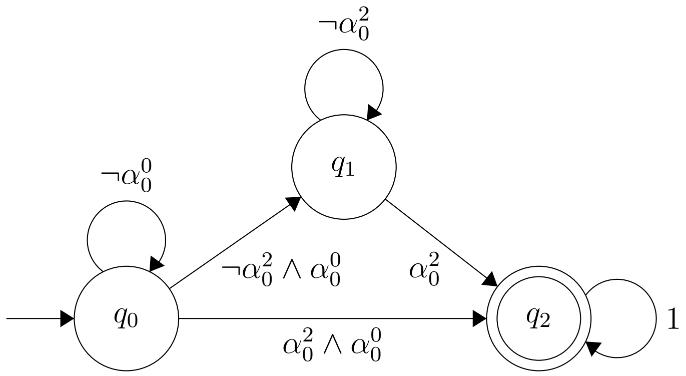

Figure 4.4 DBA for the command *"go to Boston and go through the state forest on the way."* The LTL formula is $\mathcal{F}(\alpha_0^0 \wedge \mathcal{F}(\alpha_0^2))$, where α_0^0, α_0^2 are atomic propositions resolved to map features at the `landmark` level (*"state forest"*) and the `city` level (*"Boston"*). The transition from q_0 to q_2 is considered spatially infeasible and removed before planning.

aerial robot uses A* within a lower-level projection of transited higher-level mapping features, as shown in Figure 4.3. The aerial robot constructs this projection using two R-Trees (Guttman, 1984, Howard Butler, Brent Pedersen, Sean Gilles, and others)—one storing edges, the other storing map feature geometries—to identify `landmark`-level edges within visited and crossed `neighborhood` and/or `city` states. Additionally, the R-Trees identify `landmark`-level edges bridging a single discontinuous region between higher-level states, for example, a river separating two `cities`. The path completes at a `landmark` near the boundary of the higher-level goal state. The process of calculating a `landmark`-level projection and planning an A* path is repeated for all sub-sequences computed at higher `neighborhood` and `city` levels of abstraction.

4.6 Semantic Filtering

Goals and constraints can often be specified at the base level of abstraction. For example, the *"state forest"* refers to a map feature at the `landmark` level of abstraction. The aerial robot must use a `landmark`-level MDP to ensure the first subtask—go through the state forest—is satisfied. However, as the environment grows large, it becomes computationally infeasible to plan at the base level of abstraction with an MDP.

We approach this problem by filtering the environment according to the stay and goal conditions of the current subtask. Notice these conditions are composed from atomic propositions of the LTL formula. For example, in Figure 4.4, the goal condition of (q_0, q_1) is $\neg\alpha_0^2 \wedge \alpha_0^0$, which translates to $\neg$Boston $\wedge$ state forest. All $\alpha \in AP$ are resolved

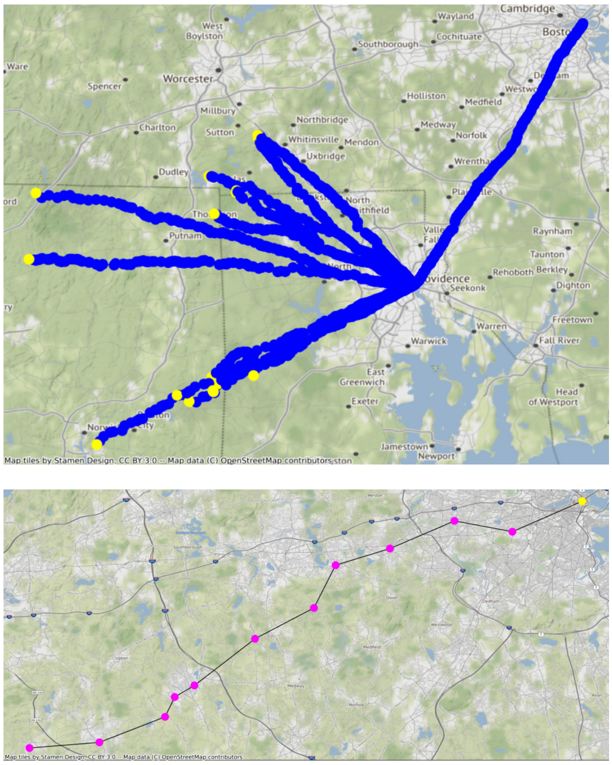

Figure 4.5 Filtered graphs for tasks (q_0, q_1) (top) and (q_1, q_2) (bottom). The blue points denote `landmark`-level paths and the magenta points denote a `city`-level path. The yellow points represent the goal state(s) for the subtask.

to salient mapping features for the user's language command (see Section 4.7); by extension, all stay and goal conditions are resolved to salient mapping features for their corresponding subtask. The aerial robot uses these features to calculate a filtered view of the environment graph.

First, the aerial robot generates a set of paths Ω that include and exclude salient mapping features Υ of the current subtask. These paths are generated at the minimally-satisfiable level of abstraction using A*. In the example task, $\Omega_{q_0 . q_1}$ is initialized with `landmark`-level paths to state forests and Boston, including paths to state forests that avoid Boston, and a path to Boston that avoids state forests. Since Boston is a `city`-level feature, A* plans to a `landmark`-level feature near Boston's centroid, identified by retrieving the centroid from H and querying the Ball tree $t^0 \in T$. Second, all vertices and edges in Ω_{q_0,q_1} are used to create the filtered graph $\zeta^0_{q_0,q_1}$, shown in Figure 4.5. Last, ζ_{q_0,q_1} is provided to the AL-MDP of subtask (q_0, q_1) and planning commences.

The semantic filtering procedure is repeated at the start of each subtask. In effect, the semantic filter induces a compact, subtask-specific view of the environment before using an AL-MDP to solve the subtask. 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
$AP_{q_j,q_{j+1}}$, atomic propositions in the goal and stay conditions of (q_j, q_{j+1})
Output: $\zeta^i_{q_j,q_{j+1}}$, the filtered graph for (q_j, q_{j+1})

Initialize $\Omega_{q_j,q_{j+1}}$
$\Upsilon \leftarrow$ retrieve_map_features($AP_{q_j,q_{j+1}}$)
for α *in* $AP_{q_j,q_{j+1}}$ **do**
    for υ *in* Υ_α **do**
        $\omega_{to} \leftarrow$ astar(v_0, υ, g^i, H, T)
        add($\Omega_{q_j,q_{j+1}}$, ω_{to})
        if $|\Upsilon \setminus \Upsilon_\alpha| > 0$ **then**
            avoids $\leftarrow \Upsilon \setminus \Upsilon_\alpha$
            $\omega_{avoid} \leftarrow$ astar(v_0, υ, g^i, H, T, avoids)
            add($\Omega_{q_j,q_{j+1}}$, ω_{avoid})
        end
    end
end
$\zeta^i_{q_j,q_{j+1}} \leftarrow$ subgraph(g^i, $\Omega_{q_j,q_{j+1}}$)
return $\zeta^i_{q_j,q_{j+1}}$

4.7 Language Resolution

Atomic propositions of the LTL formula must be resolved to features in the environment. This challenge has been addressed in Chapter 3, however, we briefly discuss a modification to the landmark resolution procedure that enables support of hierarchical environments. Our approach follows the intuition that spatially large regions are referred to by name, while smaller local regions are referred to by name and semantic descriptors. We use cosine similarity to assess name and semantic similarity between atomic propositions and mapping features, as in Chapter 3. For each mapping feature, we pre-compute 300-dimensional GloVe embeddings (Pennington et al., 2014) via the spaCy library (Honnibal et al., 2020) of the feature's name and semantic properties. At plan time, we search for similar map features based on name similarity at the `city` and `neighborhood` levels, and both name and semantic similarity at the `landmark` level. We search the environment from the highest to lowest level of abstraction and stop the search once a match has been identified. A match is defined as the cosine similarity exceeding λ_n or λ_s, thresholds for name and semantic similarity respectively. At the `city` and `neighborhood` level we select the highest name-based match. At the `landmark` level, we select the top k number of semantic and name-based matches. We use $\lambda_n = 0.95$, $\lambda_s = 0.7$, and $k = 20$.

4.8 Evaluation

In this section, we test our system's ability to plan for natural language instructions in large outdoor environments. First, we evaluate performance and efficiency on 7 classes of LTL formulae. Second, we test performance on the example task with different multi-feature thresholds. Third, we test natural language commands referring to features in our environment. Last, we demonstrate functionality with a simulated aerial robot. Our system demonstrates tractable planning performance on non-Markovian instructions in large outdoor environments.

4.8.1 Environment

We conduct our tests in the environment centered on Brown University, Providence, RI, USA (Figure 4.2). The environment was generated by querying for `landmarks`, `neighborhoods`, and `cities` within an 80-kilometer radius of the center point. The `landmark` and `city`-level queries returned map features with centroids more than 80 kilometers from the environment center point. We decided to keep these features in the map. In total, the environment contains 251,184 states and 752,866 action edges. There are 250,502 landmarks, 408 neighborhoods, and 274 cities. Up to 23 actions are available for `landmark`-level states ($\mu = 5.99$, $\sigma = 1.47$), up to 12 actions are available for `neighborhood`-level states ($\mu = 5.912$, $\sigma = 1.45$), and up 10 actions are available for `city`-level states ($\mu = 5.85$, $\sigma = 0.99$).

4.8.2 Long-Range Planning

We test our system on 112 LTL formulae encoding 7 temporal structures and spanning specified distances up to 80 kilometers. Each formula contains up to 4 atomic propositions specified at the `city` level of the abstraction hierarchy. Each proposition contains location references that are known to resolve to one mapping feature. An example input formula is $\{\neg b \; \mathcal{U} \; a$, b: *cumberland*, a: *wrentham*$\}$, where *cumberland* and *wrentham* resolve to the centroids of Cumberland, Rhode Island, USA and Wrentham, Massachusetts, USA. The a proposition represents the goal state and is varied for each formula. The b, c, and d propositions represent constraints and are constant across all commands. In this experiment, we consider two measures of distance: specified and planned. Specified distance is the straight-line geodesic distance between the start coordinate and the goal state's centroid. Planned distance is the total length of the path.

Results for this experiment are shown in Figures 4.6, 4.7, and 4.8. We test three planning configurations: no abstraction and semantic filtering (NA/S), abstraction and semantic filtering (A/S), and abstraction without semantic filtering (A/NS). For each planning configuration, we evaluate planning performance (planning time), planning efficiency (number of backups), and spatial efficiency (difference in planned distance) as a function of the LTL formula's specified distance. Planning time includes location reference resolution, AP-MDP-based planning, semantic filtering (for NA/S and A/S), and `landmark`-level path resolution (for A/S and A/NS). The horizontal error bars in the performance graphs (Figure 4.6) represent the difference between specified distance and planned distance. Planned distance can be less than specified distance because SD is calculated between state center points, but the planners can satisfy higher-level goals and constraints at the state boundaries. Conversely, planned distance can be greater than specified distance when temporal constraints require substantial deviations along the path to the goal state.

The abstract planners exhibit considerably higher average performance ($\mu_{\text{A/S}} = 20.53$s, $\mu_{\text{A/NS}} = 27.05$s) than the no-abstraction planner ($\mu_{\text{NA/S}} = 206.13$s) and sustain high performance as temporal complexity and specified distance increases (Figure 4.6). We observe a marginal difference in average performance and planning efficiency between A/S and A/NS, suggesting that abstraction offers comparatively greater performance and efficiency gains (Figure 4.7). Importantly, the benefits of abstraction are not consistently available to the aerial robot. When goals and constraints are specified at the `landmark` level, the aerial robot must plan in a significantly larger state space without abstraction. The NA/S planner simulates these lower-level planning conditions and demonstrates tractable performance and efficiency. In the simplest LTL formula ($\mathcal{F}(a)$), the NA/S planner exhibits slightly better performance than A/NS over shorter distances (specified distance = 4.38km, 10.82km, 16km, 20.42km). Abstract planning carries additional overhead (`landmark`-level path resolution) that can lower relative performance efficiency when the state space is sufficiently small. However, as temporal complexity and size of the state space increases, abstract planning offers considerable performance gains. We attribute variance in NA/S planner efficiency to the non-uniform density of the `landmark`-level state space. Some paths go through urban, higher-density regions, while others go through rural, lower-density regions. As a concrete example, the NA/S planner with 70-kilometer $\mathcal{F}(a) \wedge \mathcal{G}(\neg b)$ plans in a state space containing 761 landmarks. The NA/S planner with 80-kilometer $\mathcal{F}(a) \wedge \mathcal{G}(\neg b)$ plans in a state space containing 311 landmarks. The corresponding difference in planning time suggests state space compaction is especially important at the `landmark` level.

We compare planned distances between the A/S, A/NS, and NA/S configurations to evaluate spatial efficiency (Figure 4.8). The average spread in planned distance is relatively small over all configurations ($\mu = 3.23$km, $\sigma = 3.02$km) and even smaller between semantic filtering configurations ($\mu = 2.07$km, $\sigma = 2.39$km). We attribute cases of longer NA/S paths to the reward function, which is designed to minimize the number of actions taken by the aerial robot. In large outdoor environments composed of irregular geometries, NA/S action minimization at the `landmark`-level can be less spatially efficient than A/S and A/NS `city`-level action minimization coupled with A* `landmark`-level path resolution.

Last, as a performance baseline for NA/S we planned one LTL formula ($\mathcal{F}(a)$, specified distance = 4.38 kilometers) with no abstraction and no semantic filtering; the planning time was 11.57 hours. While this comparison is not direct—semantic filtering places hard constraints on the depth of the state space—our system's ability to tractably operate at the `landmark` level is an encouraging result.

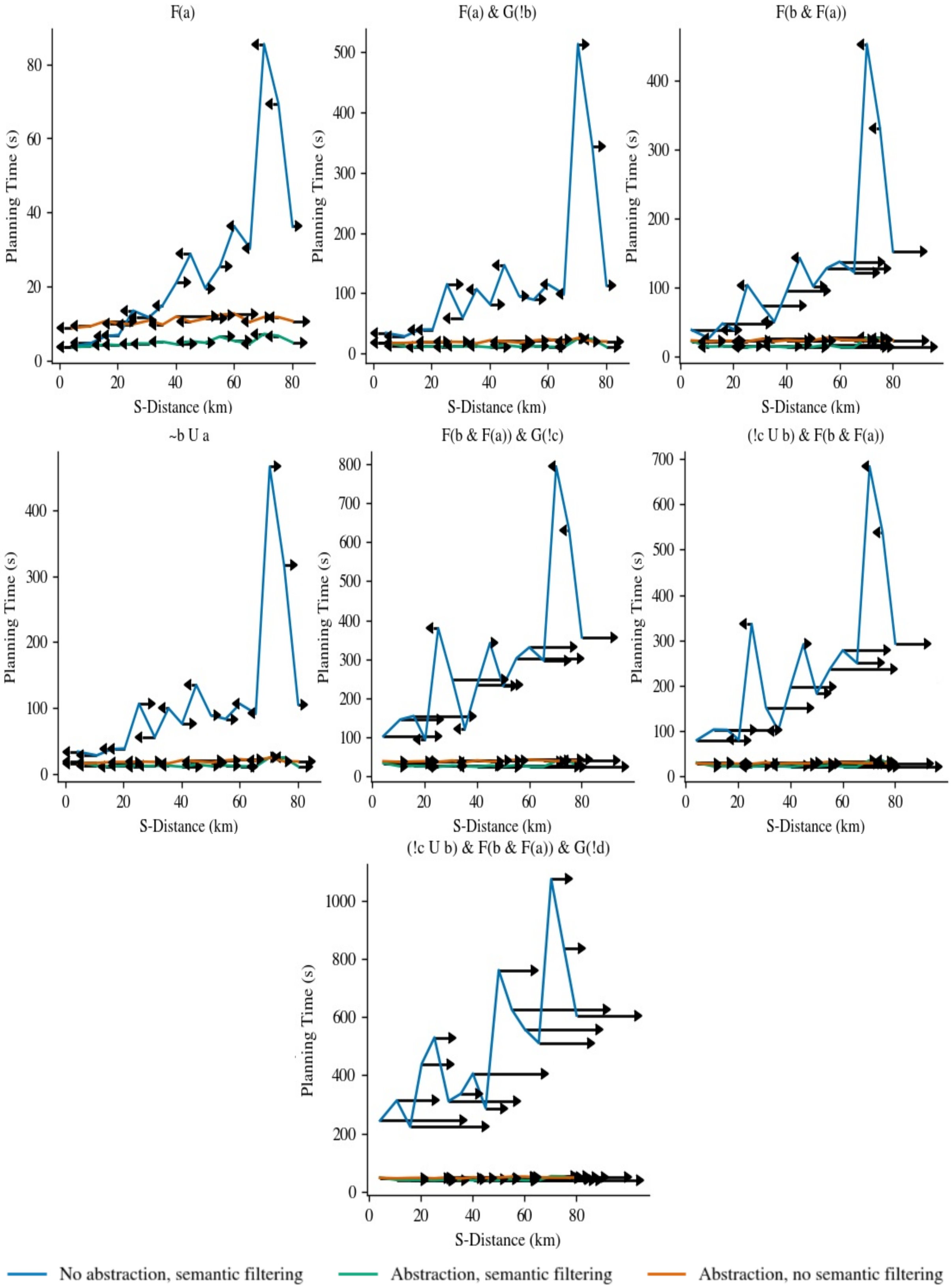

Figure 4.6 Long-range planning performance. Planning time (seconds, lower is better) is shown on the vertical axes. Specified distance (kilometers) is shown on the horizontal axes. The difference between specified and planned distance is shown by the horizontal arrows on the performance curves.

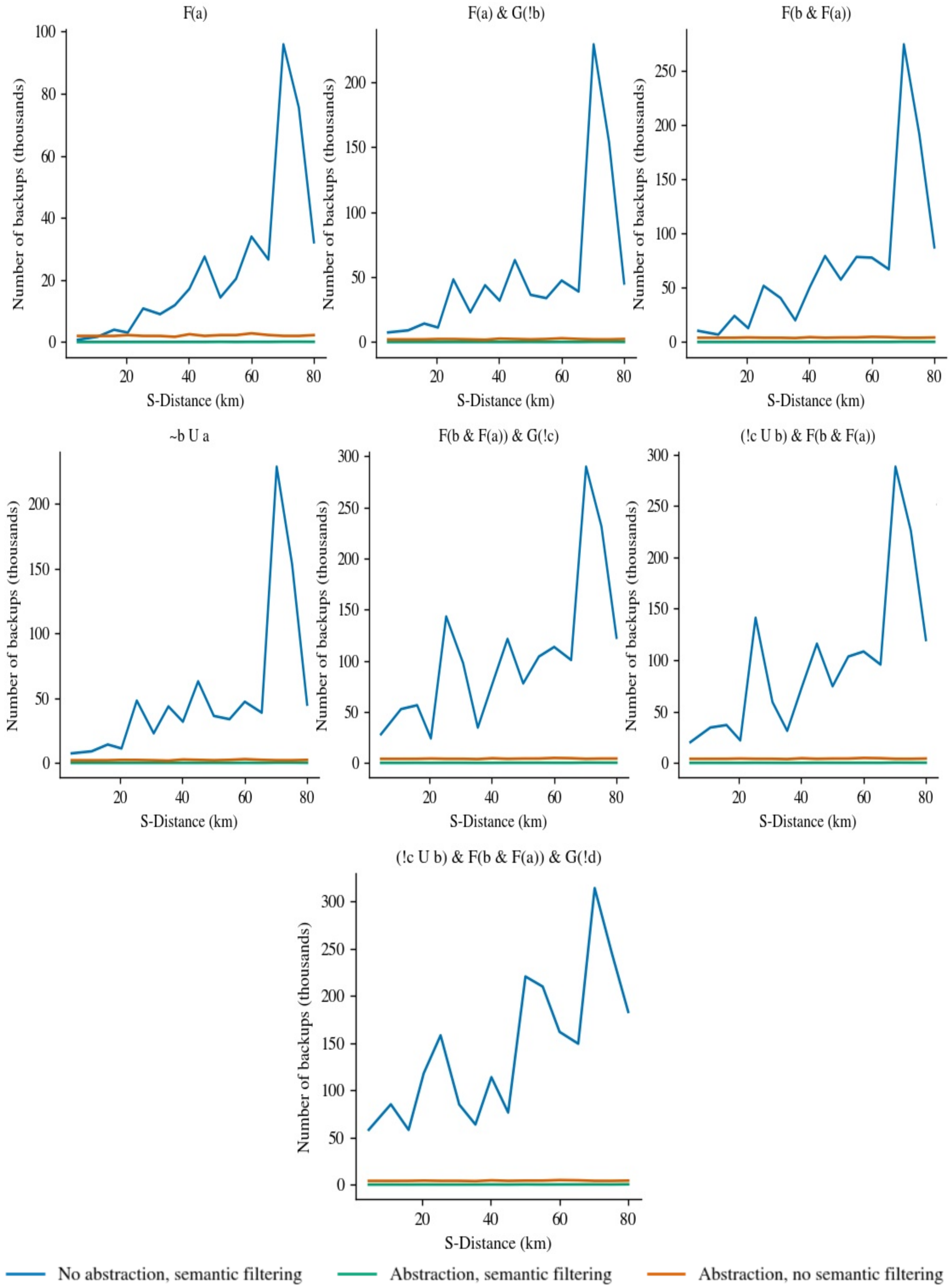

Figure 4.7 Long-range planning efficiency. Bellman backups (lower is better) is shown on the vertical axes. Specified distance (kilometers) is shown on the horizontal axes.

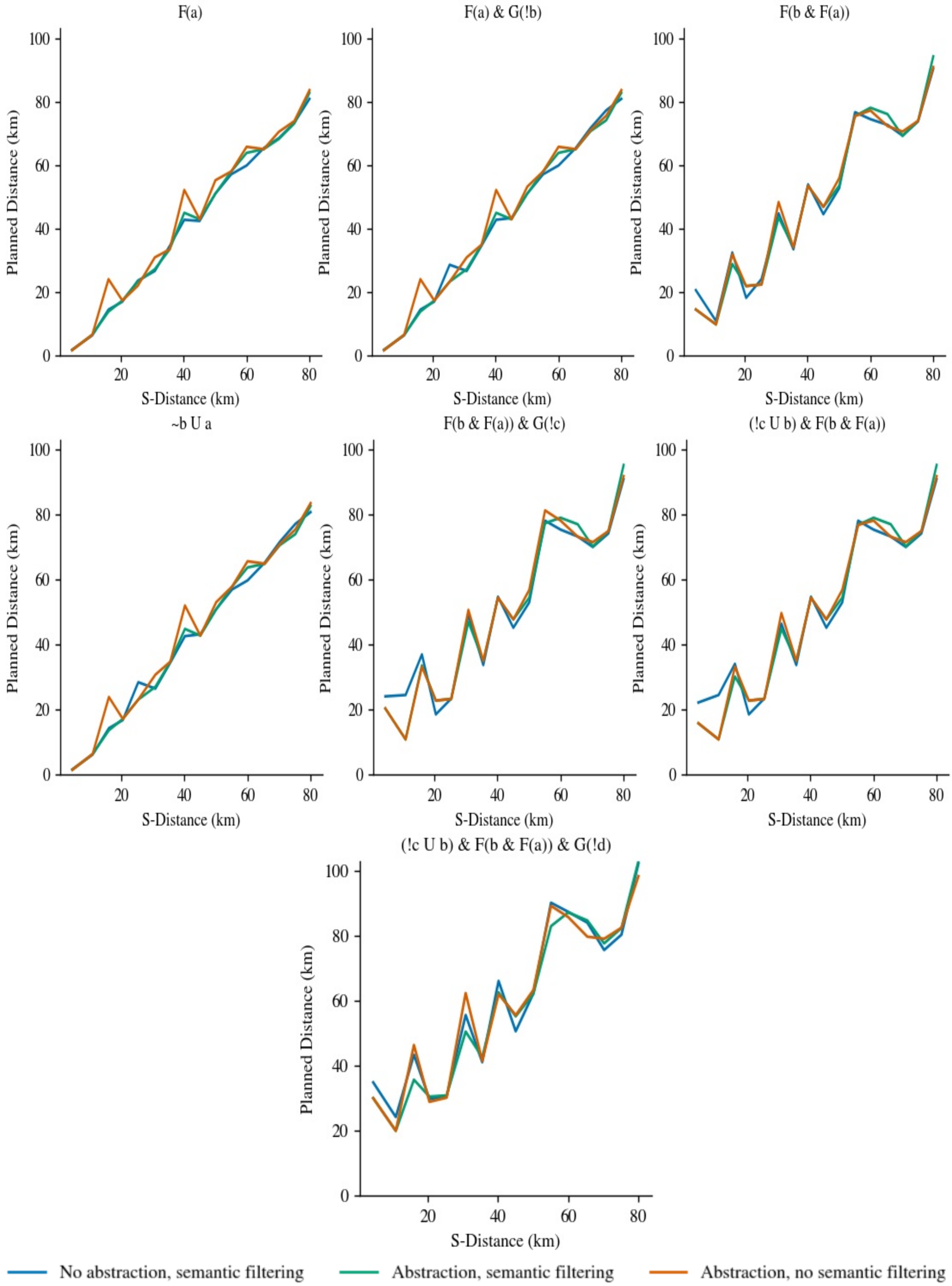

Figure 4.8 Long-range planning distances. Total planned distance (kilometers) is shown on the vertical axes. Specified distance (kilometers) is shown on the horizontal axes.

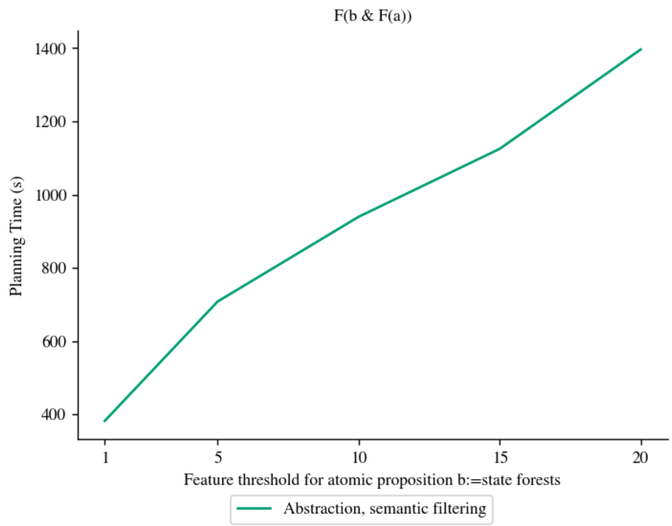

Figure 4.9 Planning results for the example task *"go to Boston and go through the state forest on the way."* The LTL formula is $\mathcal{F}(b \wedge \mathcal{F}(a))$, where atomic proposition a corresponds to *"Boston"* and proposition b corresponds to the *"state forest."* The horizontal axis is the feature threshold for atomic proposition b, the vertical axis is planning time.

4.8.3 Planning with multiple semantic matches

In the example task *"go to Boston and go through the state forest on the way,"* the phrase *"state forest"* could refer to multiple unique features in the environment. The planner supports multi-feature atomic propositions by selecting the feature along the highest-reward path. The number of potential features must be capped to sustain tractable operation. We test the example task with a systematically increasing multi-feature threshold. Our results, shown in Figure 4.9, demonstrate the relationship between planning time and environment size. Planning time for the example task's LTL formula is 382.15 seconds with a feature threshold of one; planning time is 1396.44 seconds with a feature threshold of 20. The environment size scales accordingly: the one-feature threshold induces an environment view with 565 states (552 landmarks, 13 cities); the 20-feature threshold induces a view with 1615 states (1603 landmarks, 12 cities). Larger multi-feature thresholds require more path generations during semantic filtering and Bellman backups during planning. Balancing the multi-feature threshold and planning time 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 one for all propositions and refer to environment features with unique names.

Figure 4.10 Simulated Skydio R1 flying over Quaddick State Forest, Connecticut (left) and reaching Millennium Park, Boston (right).

4.8.4 Language Grounding

We test the example task *"go to Boston and go through the state forest on the way"* and 44 randomly sampled commands from the Language to Landmarks dataset (Berg et al., 2020) on the sequence-to-sequence model presented in Oh et al. (2019). We used this model to test an alternate grounding procedure that leverages a publicly available pre-trained model (Shi and Lin, 2019, Gardner et al., 2018) to pre-process location reference extractions. The dataset contains language-LTL pairings referring to trajectories in real OpenStreetMap environments. These commands are challenging to parse because they exhibit unrestricted vocabulary and grammar. In each command, we replace location references grounding to LTL atomic propositions with randomly selected `landmark`, `neighborhood`, and `city` location names from the testing environment. We use Semantic Role Labeling (Shi and Lin, 2019, Gardner et al., 2018) to extract location references and map to in-vocabulary phrases before grounding. Similar to Oh et al. (2019), we observe lower grounding accuracy on unseen commands. Challenges include difficulty of the testing commands, faulty SRL extractions, and limited generalizability of the model. However, we note the example task successfully grounded to its corresponding LTL formula.

4.8.5 Aerial robot in simulation

We tested the example task on a simulated Skydio R1 quadcopter, shown in Figure 4.1 and Figure 4.10. The simulator is built with Unity (Unity Technologies) and is detailed in Section 3.6.1. The quadcopter successfully followed the planner's output.

Chapter 5

Conclusions

This work presents language grounding and long-range planning systems for an aerial robot operating in large outdoor environments. These systems understand temporally complex instructions and unrestricted references to map features without advance training on the environment. Corpus and user evaluations establish accuracy and usability of the language system, while benchmarks on multiple classes of LTL formulae demonstrate tractable planning performance while obeying complex goals and constraints. Testing on a simulated aerial robot demonstrates functionality of both systems.

Future work can build on the landmark resolution procedure, environment representation, and planning system. Landmark resolution accuracy could benefit from more streams of semantic data, such as visual inputs and user gestures. In addition, a user feedback system would help disambiguate multi-feature atomic propositions. Natural language instructions can specify goals and constraints below the `landmark` level, for example, action-based goals like *"turn right then go to the state forest"* and local feature-based constraints like *"avoid large trees on the way."* Adjustable resolution at salient map regions would support increased planning precision while sustaining a globally compact representation of the environment. For real-world deployment, the environment representation and planner need to integrate airspace rules. Last, users may communicate semantic objectives such as *"take the most scenic route."* The planner could learn intuitive path heuristics and formal representations of non-temporal objectives to support these instructions.

References

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.

J. Andreas and D. Klein. Alignment-based compositional semantics for instruction following. *CoRR*, abs/1508.06491, 2015. URL http://arxiv.org/abs/1508.06491.

Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013. doi: 10.1162/tacl_a_00209. URL https://www.aclweb.org/anthology/Q13-1005.

Y. Artzi, D. Das, and S. Petrov. Learning compact lexicons for ccg semantic parsing. In *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, October 2014.

D. Arumugam, S. Karamcheti, N. Gopalan, L. L. Wong, and S. Tellex. Accurately and efficiently interpreting human-robot instructions of varying granularities. *arXiv preprint arXiv:1704.06616*, 2017.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

M. Berg, D. Bayazit, R. Mathew, A. Rotter-Aboyoun, E. Pavlick, and S. Tellex. Grounding language to landmarks in arbitrary outdoor environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 208–215, 2020. doi: 10.1109/ICRA40945.2020.9197068.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

J. Brooke. SUS-a quick and dirty usability scale. *Usability Evaluation in Industry*, 189 (194):4–7, 1996.

J. R. Büchi. On a decision method in restricted second order arithmetic. In *The collected works of J. Richard Büchi*, pages 425–435. Springer, 1990.

D. L. Chen. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL-2012)*, pages 430–439, July 2012. URL `http://www.cs.utexas.edu/users/ai-lab/?chen:acl2012`.

J. Cheng, S. Reddy, V. Saraswat, and M. Lapata. Learning structured natural language representations for semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44–55, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1005. URL `https://www.aclweb.org/anthology/P17-1005`.

daltonfury42. Truecase. URL `https://github.com/daltonfury42/truecase`.

M. Damonte, R. Goel, and T. Chung. Practical semantic parsing for spoken language understanding. *CoRR*, abs/1903.04521, 2019. URL `http://arxiv.org/abs/1903.04521`.

B. N. Delaunay. Sur la sphère vide. *Bull. Acad. Sci. URSS*, 1934(6):793–800, 1934.

L. Dong and M. Lapata. Language to logical form with neural attention. *CoRR*, abs/1601.01280, 2016. URL `http://arxiv.org/abs/1601.01280`.

L. Dong and M. Lapata. Coarse-to-fine decoding for neural semantic parsing. In *ACL*, 2018.

M. Drager and A. Koller. Generation of landmark-based navigation instructions from open-source data. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012.

J. Dzifcak, M. J. Scheutz, C. Baral, and P. W. Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. *2009 IEEE International Conference on Robotics and Automation*, pages 4163–4168, 2009.

S. Edunov, M. Ott, M. Auli, and D. Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500. Association for Computational Linguistics, 2018.

M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. Liu, M. Peters, M. Schmitz, and L. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2018.

S. Gehrmann, Y. Deng, and A. M. Rush. Bottom-up abstractive summarization. *ArXiv*, abs/1808.10792, 2018.

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.

E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

J. Gu, Z. Lu, H. Li, and V. O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. *ArXiv*, abs/1603.06393, 2016.

Ç. Gülçehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio. Pointing the unknown words. *CoRR*, abs/1603.08148, 2016. URL http://arxiv.org/abs/1603.08148.

A. Guttman. R-trees: A dynamic index structure for spatial searching. *SIGMOD Rec.*, 14(2):47–57, June 1984. ISSN 0163-5808. doi: 10.1145/971697.602266. URL https://doi.org/10.1145/971697.602266.

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

S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In P. A. Hancock and N. Meshkati, editors, *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139 – 183. North-Holland, 1988. doi: https://doi.org/10.1016/S0166-4115(08)62386-9. URL http://www.sciencedirect.com/science/article/pii/S0166411508623869.

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

M. Honnibal and M. Johnson. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL https://aclweb.org/anthology/D/D15/D15-1162.

M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303.

Howard Butler, Brent Pedersen, Sean Gilles, and others. Rtree: Spatial indexing for python. URL https://toblerity.org/rtree/.

B. Hu, Q. Chen, and F. Zhu. Lcsts: A large scale chinese short text summarization dataset. In *EMNLP*, 2015.

A. S. Huang, S. Tellex, A. Bachrach, T. Kollar, D. Roy, and N. Roy. Natural language command of an autonomous micro-air vehicle. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2663–2669. IEEE, 2010.

J. Inman. *Navigation and Nautical Astronomy for the Use of British Seamen*. C. and J.Rivington, 1835. URL https://books.google.com/books?id=-fUOnQEACAAJ.

V. Joshi, M. E. Peters, and M. Hopkins. Extending a parser to distant domains using a few dozen partially annotated examples. In *ACL*, 2018.

A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics, April 2017.

R. Knowles and P. Koehn. Context and copying in neural machine translation. In *EMNLP*, 2018.

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

T. Kollar, S. Tellex, D. Roy, and N. Roy. Grounding verbs of motion in natural language commands to robots. *Experimental Robotics Springer Tracts in Advanced Robotics*, page 31–47, 2014. doi: 10.1007/978-3-642-28572-1_3.

K. Konolige, E. Marder-Eppstein, and B. Marthi. Navigation in hybrid metric-topological maps. In *2011 IEEE International Conference on Robotics and Automation*, pages 3041–3047. IEEE, 2011.

H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Translating structured english to robot controllers. *Advanced Robotics*, 22(12):1343–1359, 2008.

B. Kuipers. The spatial semantic hierarchy. *Artificial intelligence*, 119(1-2):191–233, 2000.

B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, pages 4845–4851 Vol.5, 2004.

I. Kuric, V. Bulej, M. Saga, and P. Pokorny. Development of simulation software for mobile robot path planning within multilayer map system based on metric and topological maps. *International Journal of Advanced Robotic Systems*, 14(6): 1729881417743029, 2017. doi: 10.1177/1729881417743029. URL `https://doi.org/10.1177/1729881417743029`.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition. *ArXiv*, abs/1603.01360, 2016.

L. V. Lita, A. Ittycheriah, S. Roukos, and N. Kambhatla. Truecasing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 152–159, 2003.

M. L. Littman, U. Topcu, J. Fu, C. L. I. Jr., M. Wen, and J. MacGlashan. Environment-independent task specifications via GLTL. *CoRR*, abs/1704.04341, 2017. URL `http://arxiv.org/abs/1704.04341`.

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*, pages 65–82. Springer, 1999.

J. MacGlashan, M. Babes-Vroman, M. desJardins, M. L. Littman, S. Muresan, S. Squire, S. Tellex, D. Arumugam, and L. Yang. Grounding english commands to reward functions. In *Robotics: Science and Systems*, 2015.

M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. *The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference*, 2(6):4, 2006.

Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, Berlin, Heidelberg, 1992. ISBN 0-387-97664-7.

Mapbox. Mapbox unity sdk. URL `https://github.com/mapbox/mapbox-unity-sdk`.

C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*, HRI '10, pages 251–258, Piscataway, NJ, USA, 2010a. IEEE Press. ISBN 978-1-4244-4893-7. URL `http://dl.acm.org/citation.cfm?id=1734454.1734552`.

C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 251–258. IEEE, 2010b.

C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In *Experimental robotics*, pages 403–415. Springer, 2013.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999792.2999959`.

D. K. Misra, K. Tao, P. Liang, and A. Saxena. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 992–1002, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1096. URL `https://www.aclweb.org/anthology/P15-1096`.

D. K. Misra, J. Langford, and Y. Artzi. Mapping instructions and visual observations to actions with reinforcement learning. *CoRR*, abs/1704.08795, 2017. URL `http://arxiv.org/abs/1704.08795`.

Y. Oh, R. Patel, T. Nguyen, B. Huang, E. Pavlick, and S. Tellex. Planning with state abstractions for non-markovian task specifications. *arXiv preprint arXiv:1905.12096*, 2019.

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

OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2017.

OpenStreetMap contributors. Elements. `https://wiki.openstreetmap.org/wiki/Elements`, November 2020.

OpenStreetMap contributors. Map features. https://wiki.openstreetmap.org/wiki/Map_features, February 2021.

C. Padwick. Ai for agriculture: How pytorch enables blue river's robots. Available at https://www.therobotreport.com/ai-for-agriculture-how-pytorch-enables-blue-rivers-robots/, 2020.

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. doi: 10.15607/RSS.2020.XVI.016.

R. Paul, A. Barbu, S. Felshin, B. Katz, and N. Roy. Temporal grounding graphs for language understanding with accrued visual-language context. *ArXiv*, abs/1811.06966, 2018.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

A. Poncela, E. Perez, A. Bandera, C. Urdiales, and F. Sandoval. Efficient integration of metric and topological maps for directed exploration of unknown environments. *Robotics and Autonomous Systems*, 41(1):21 – 39, 2002. ISSN 0921-8890. doi: https://doi.org/10.1016/S0921-8890(02)00272-5. URL http://www.sciencedirect.com/science/article/pii/S0921889002002725.

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.

K.-F. Richter and S. Winter. *Introduction: What Landmarks Are, and Why They Are Im-*

portant, pages 1–25. Springer International Publishing, April 2014. ISBN 978-3-319-05731-6. doi: 10.1007/978-3-319-05732-3_1.

A. Rousell, S. Hahmann, M. Bakillah, and A. Mobasheri. Extraction of landmarks from openstreetmap for use in navigational instructions. In *Association of Geographic Information Laboratories in Europe*, 2015.

C. Sammut and G. I. Webb, editors. *TF–IDF*, pages 986–987. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_832. URL `https://doi.org/10.1007/978-0-387-30164-8_832`.

J. Sauro. Sustisfied? little-known system usability scale facts user experience magazine, 2011. URL `https://uxpamagazine.org/sustified/`.

P. Shi and J. Lin. Simple bert models for relation extraction and semantic role labeling. *ArXiv*, abs/1904.05255, 2019.

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

Skydio. Company overview. Technical report, Skydio, Inc., 2020. URL `https://drive.google.com/file/d/13V4XcHudwhI61zyfz5L7eJ_BNvg0-GOV/view`.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL `http://arxiv.org/abs/1409.3215`.

H. Tan, L. Yu, and M. Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In *Proceedings of NAACL-HLT 2019*. Association for Computational Linguistics, 2019.

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*, volume 25, 2011a.

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy. Approaching the symbol grounding problem with probabilistic graphical models. *AI Magazine*, 32(4):64, 2011b. doi: 10.1609/aimag.v32i4.2384.

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

S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21 – 71, 1998. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(97)00078-7. URL http://www.sciencedirect.com/science/article/pii/S0004370297000787.

Unity Technologies. Unity. URL https://unity.com/.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

G. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs. *Journal für die reine und angewandte Mathematik*, 134:198–287, 1908.

C. Wang, C. Ross, Y.-L. Kuo, B. Katz, and A. Barbu. Learning a natural-language to ltl executable semantic parser for grounded robotics, 2020.

Yelp. Yelp open dataset, 2019. URL https://www.yelp.com/dataset.

P. Yin, C. Zhou, J. He, and G. Neubig. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. *CoRR*, abs/1806.07832, 2018. URL http://arxiv.org/abs/1806.07832.

Q. Zhou, N. Yang, F. Wei, and M. Zhou. Sequential copying networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. URL https://ojs.aaai.org/index.php/AAAI/article/view/11915.