Optimized Robotic Information Gathering using Semantic Language Instructions

Ian C. Rankin, Seth McCammon, Geoffrey A. Hollinger
Collobrative Robotics and Intelligent Systems (CoRIS)
Oregon State University
Corvallis, Oregon
Email: {rankini, mccammos, geoff.hollinger}@oregonstate.edu

***Abstract*—This work presents a framework that uses language instructions to define the constraints and objectives for robots gathering information about their environment. Commanding and operating robots in the field is a challenging task, often requiring a team of expert operators to run a single robot. Commands require setting non-intuitive constraints and reward functions in order for the robot to achieve sensible actions. This work introduces a framework to convert language instructions to robot plans without a large corpus of domain specific data. Language commands provide an intuitive interface for operators to give complex instructions to robots. This work introduces three main contributions: a framework to map language instructions to constraints and rewards for robot planners; a homotopic constrained information gathering algorithm; and an automatic semantic feature detection algorithm of upwelling fronts.**

I. Introduction

Designing information gathering missions for robots requires setting reward functions, cost functions, and constraints for the planning algorithm. These functions are typically challenging to define, and they cannot be changed without modification of the robot's software. In this paper, we propose a framework which allows the user to set mission goals, constraints, and reward functions more naturally using language instructions. We focus on the marine robot task of studying coastal upwelling fronts. These fronts are locations where cold, nutrient rich, deep water is pushed to the surface. The mixing of the warm surface water and the deep water is of interest scientifically for both biological and physical oceanography [27]. Our framework allows complex language instructions for information gathering tasks; an example command and planned path is given in Figure 1.

The main challenge in generating robot plans from language instructions is grounding the language to real features and actions in the environment [2]. Solving this problem requires a process of identifying salient features in the environment then mapping the language instructions to these features. There are three key components to the information gathering from language instruction framework we present. First we separate the detection of semantic features and the language understanding task into separate problems. Then we directly

This research was funded in part by NSF grants IIS-1723924 and IIS-1845227.

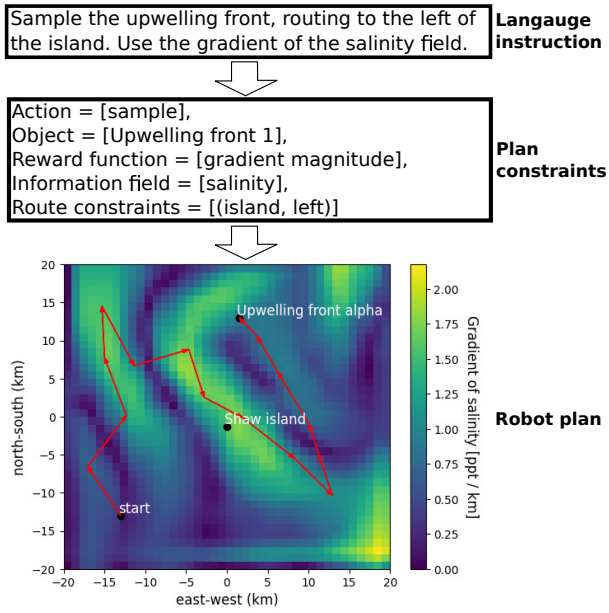


Fig. 1: Example planned path for "Sample the upwelling front, routing to the left of the island. Use the gradient of the salinity field." The language instruction is grounded into a set of plan constraints, and then a constrained information gathering algorithm is ran from the given plan constraints.

ground the language instruction from the dependency information contained in Universal Dependency (UD) trees, without requiring a corpus of language instructions to robot plan constraints. Finally, we use homotopic constraints [3] to constrain the information gathering paths to distinct homotopy classes defined by the given language instruction.

To create an information gathering plan from a given language instruction, we first use the Stanford parser [5, 11, 17] to generate a UD tree. Semantic features are generated from the automatic semantic feature extractor and other named features. The UD tree is then used to directly ground the language instruction to robot action, constraints, reward function, and information field. The constraints are turned into a set of robot plan constraints with a given topological trajectory class, goal sample locations, and a reward function. Finally, an information gathering algorithm constrained to the desired topological class is used to generate the information gathering robot plan.

This paper introduces three main contributions. The first

contribution is a grounding framework which directly grounds language instructions from the relationship information in a UD tree to robot plan constraints. The second contribution is a homotopy-constrained information gathering algorithm which uses Dijkstra's algorithm to precompute a homotopy augmented graph to ensure expansion of the information gathering algorithm to locations that satisfy the given constraints. The final contribution is a semantic feature extractor to automatically locate coastal upwelling fronts using a Convolutional Neural Network (CNN).

This paper is structured as follows. In Section II, we discuss related works to this paper. In Section III, we outline our proposed method including grounding language instructions, semantic homotopy graph formulation, constrained information gathering algorithm, and the automatic semantic feature extractor. Then, in Section IV we present the results of the paper including results of generating robot plans from language instructions taken from robotics researchers, and results from the automatic upwelling front detector. Finally, in Section V we discuss the limitations of our approach, future research directions, and the conclusions of the paper.

II. Related Works

A. Language grounding

One way to solve the mapping of language instruction to robot plan is using embodied AI, which is studying visually-grounded navigation instruction following and question answering [1, 25]. These methods combine the language understanding and feature recognition into a single framework. However, embodied AI is focused on deep learning methods which require a significant amount of training data. In current embodied AI research this training is gathered from photo-realistic indoor environments allowing the algorithms to train over thousands to millions of iterations [25]. Unfortunately, it can be difficult to collect the large quantities of data needed to train these networks from field robotics domains. This makes current embodied AI algorithms infeasible for these environments.

An alternative way to solve this problem is to use natural language dependency parsers, such as the Stanford Parser [5], to find the structure of the language instructions before grounding. The Stanford Parser extracts a UD tree [14], which decomposes the sentence into its component grammatical structures. Previous works have used the shape of the dependency tree to inform the language grounding process [9, 23]. More recent works have focused on grounding language instructions using neural networks for mapping semantic features to language instructions and learning unknown object groundings [15, 24]. These methods all depend on greater amount of learning requiring more data in order to operate effectively. The closest work to the approach presented here is Howard et al. [6], which uses a probabilistic graphical model with the shape generated from the UD tree to construct robot plan constraints from natural language instructions. Unlike Howard et al. [6], which requires a corpus of language instructions that map to robot constraints, our method uses explicit groundings to map language instructions to robot plan constraints without the need for a corpus.

B. Semantics and Homotopy

Semantic maps and features have been used in robotics in indoor environments extensively. Semantic maps are represented using graphs $G = (V, E)$ where vertices, $v \in V$, represent rooms, while edges, $e \in E$, represent connections between rooms [4, 10]. Semantic features for these environments are easy to determine using names of rooms like "kitchen". However, in unstructured field robotic domains finding viable semantic features is challenging. Fortunately, scientifically significant ocean features, such as upwelling fronts provide nameable locations to use as semantic features for planning.

In this paper we use homotopic constraints to allow the user control over the path the robot takes rather than only the goal and start locations. Two robot paths are defined to be in the same homotopy class if they share start and end points and there exists a continuous and invertible transformation between them [3]. Previous works have used homotopy classes during robot planning [12, 16]. We use the h-signature, a homotopy invariant, to constrain paths to particular homotopy classes. Bhattacharya et al. [3] shows using an homotopy augmented graph, standard graph search algorithms, such as A*, are still valid for any admissible function valid in the standard graph. We use these results to apply the homotopy augmented graph to the informative path planning problem.

C. Upwelling front detection

Traditionally, human experts are used to create bounding boxes around upwelling fronts from satellite sea-surface temperature images. These images are then used to guide robotic exploration. Once deployed, robots can automatically detect upwelling fronts using stratification of the water temperature [28]. In our work we use the output of the Regional Ocean Modeling System (ROMS) [20] along with a convolutional neural network to automatically detect upwelling fronts before robot deployment. Our labeled data representation and problem domain are similar to Rao et al. [18]. In their paper they use data collected from an underwater unmanned vehicle as input to a learned network classifying benthic habitats from bathymetry patches. In our work we extend the patches of overhead data to coast-aligned slices of salinity data, which was found to outperform patches for detecting upwelling fronts.

Putting each of these three main contributions together we present our combined framework to ground language instructions into a robot plan.

III. Methodology

In order to transform a language instruction into a robot plan, we ground the instruction into constraints and an objective function which are then used by a planning algorithm. By then utilizing a semantic homotopy augmented graph that considers both points of interest and hazard nodes, a robot plan

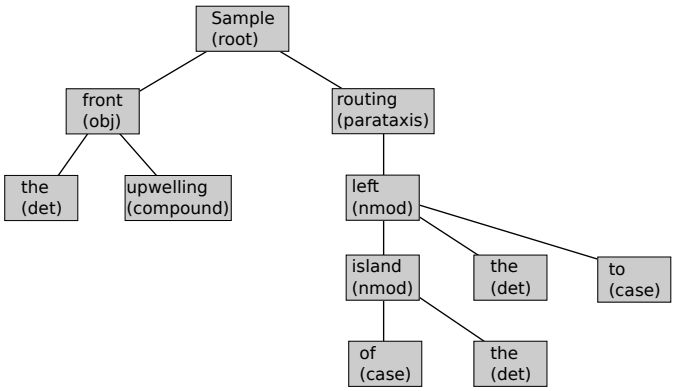


Fig. 2: Universal Dependency Tree for the command "Sample the upwelling front, routing to the left of the island." The two main branches of the tree correspond to the topological feature of interest and its corresponding routing constraint.

can be generated with topological and navigational constraints relating to these features. Two different types of planners are implemented to show the generalizability of the language instruction parsing to both route-following commands and information gathering commands.

A. Grounding language instructions

Language instruction grounding is broadly defined as mapping language instructions to real-world robot plans [2]. An example of a grounding in the phrase "Sample the upwelling front, routing to the left of the island" is the word 'routing' which maps to a constraint on the way that the robot travels through the world. To build these groundings from a language instruction to a set of robot constraints, we use the Stanford Parser to create a Universal Dependency (UD) tree [11, 14]. Universal dependencies is a framework for annotation of grammar across languages. These annotations are used to parse the different words of the instruction into groundings for actions, physical features, modifiers of features, and constraints.

Once complete, this process results in a tree structure with the head of the sentence as the root of the tree. Each node in the tree represents a syntactic word in the sentence, with a dependency relationship to the node's parent. The dependency relationship indicates the structure of the sentence, which is used to determine the type of groundings for each word. Several key dependency relationships in robot instructions and how each is handled are given below.

- **Root Node (root):** Match the main verb to a planner.
- **Adjectival and Adverb modifiers (amod, advmod):** Attach the word as a modifier to its parent node to be used to differentiate between possible groundings.
- **Object, Oblique nominal, nominal modifier, conjunct, parataxis (obj, obl, nmod, conj, parataxis):** Object nodes require grounding via the GroundNoun function to features in the environment.
- **Compound:** Compound words are merged with their parent.

Once the UD tree is generated, we use a breadth-first search to iterate through each node in the tree. At each node, depending on its dependency relationship, we ground the word using a lookup table of possible meanings using the $GroundFromList$ function described below. This function maps a word to a particular semantic feature in the environment or a semantic action (e.g. Move, Sample), modifier (e.g. Left, North), adposition (e.g. to, from), argument (e.g. routing), or function (e.g. gradient magnitude). During this process, most nodes are attached to their parents as arguments modifying the grounding.

The root of the UD tree is the action word whose grounding is the particular action planner (e.g. route following or information gathering) that is used to plan the final robot behavior. The remainder of the groundings provide this planner with its objectives and constraints. For example in the UD tree shown in Figure 2, "routing to the left of the island" imposes a constraint on the planner, while "the upwelling front" provides the objective.

While iterating through each node of the tree a mapping is required between each **obj / obl / nmod / conj / parataxis** node in the tree and the specific physical feature or features in environment it refers to. This mapping if performed by the function $GroundFromList(UDNode, list)$. This function is given a list of possible groundings and a particular word to ground. Each grounding in the list has a unique name and a set of keywords which describe the grounding (e.g. "Shaw Island" is the unique name and keywords would be "island", "shaw"). Each keyword is a member of a Wordnet synset, which means it has a semantic meaning behind that keyword [13]. During a search for a given word (e.g. "isle"), the algorithm first searches through the list to find any exact matches of the name. Failing to do so, the algorithm then looks for exact keyword matches, (e.g. if instead looking for the word island, would find all groundings with the keyword "island"). If this search also fails, then the algorithm uses the Resnik [19] revised Wu and Palmer [26] method of measuring semantic relatedness. If this distance is greater than the user-defined parameter then that feature matches with the tested keyword and is added to the possible grounding list. In practice we found that setting this threshold to 0.875 produced good mappings from words to groundings.

After mapping words using the $GroundFromList$ function, a word may ground to multiple features if it does not uniquely map to a single feature. For example, the phrase "upwelling front" may result in a list of possible groundings if there is more than one upwelling front in the environment. The grounding modifiers derived from an object node's children can be used to help to identify a mapping to a single feature. The parse function applies modifier functions on lists of groundings (e.g. the "leftmost" or "southern"), and resolves groundings to strict constraints on the action's planner. This function also checks for the validity of groundings (e.g. a reward function passed to a Move action is marked as invalid). Finally, the number of groundings for each node is checked. If the word is plural multiple groundings are allowed, otherwise if multiple groundings remain, a graceful failure is enacted which asks the user to clarify which of the groundings is correct. This failure mode can occur either if the framework

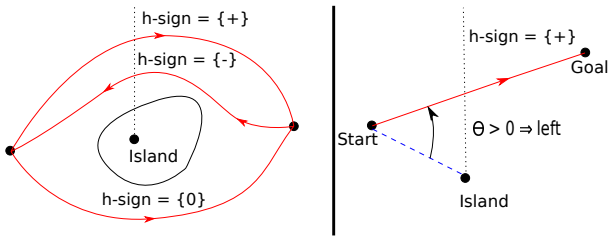


Fig. 3: Left: H-signature signs for a single obstacle. Right: Selecting the h-signature $\{0\}$ for the routing constraint (island, right) using the straight line h-signature between start to goal to determine the opposite h-signature.

misunderstands something or if the user does not fully specify the grounding.

B. Semantic homotopy augmented graph

The result of applying groundings to the parsed mission command statement is a set of relational constraints on actions (e.g. "to the left of the island"). An intuitive way to apply these constraints to paths within the planning domain is to understand the constraints as identifying a particular homotopy class of trajectory between the robot's current position and its goal. To build a plan for the robot incorporating these constraints, we turn to homotopy augmented graphs [3]. The homotopy augmented graph expands a roadmap, such as a Probabilistic Roadmap (PRM) [7], in $\mathbb{R}^2$ with an additional dimension that contains information about the homotopy class of trajectory required to arrive at a particular vertex on the graph. The homotopy information is encoded in a h-signature, a variable which uniquely describes the topological class of a given trajectory belongs to.

In our planning problem, topological features can fall in one of two classes, and depending on the mission specification, a given geographic or oceanographic feature can fall in either category (though never both at the same time). The first of these feature classes is hazards: features which should be avoided, and which act as the obstacles in the environment which partition it into different topological trajectory classes. The second type of features are points of interest. These features are named points of interest of our mission, and must be included as vertices of the homotopy augmented graph.

When planning with a topological constraint, the first step is to construct our homotopy augmented graph. We begin this process by adding all points of interest as vertices, and then transitioning to randomly sampling the environment to construct a PRM. Then, we construct our homotopy augmented graph using the method described in [3], using the locations of the hazards as the representative points of obstacles. Once the homotopy augmented graph is constructed, the function $translate(\text{routing constraints})$ is used to translate the constraints from the mission description into a desired h-signature. It is guaranteed that route instructions to the left will have either the h-signature $\{+\}$ or $\{0\}$ for a single feature, and right instructions will have either the h-signature $\{-\}$ or $\{0\}$. To translate the route constraint each plan needs to select whether left or right has the h-signature $\{0\}$. This is done by checking the h-signature and the direction of the straight line path from the start to the given goal. If the direction matches the given mission description, then that h-signature is used otherwise the opposite signature is used (see Figure 3). Finally, we use A* to plan the shortest path along the homotopy augmented graph to the next point of interest in the desired homotopy class.

C. Constrained information gathering algorithm

In this paper a modified formulation of the Informative Path Planning problem is used [21]. The standard formulation is interested in finding an optimal path for a robot which maximizes the information gathering reward function, subject to a cost budget. We reformulate it to look for an optimal path subject to both cost, goal, and homotopic constraints given by the language instruction. The formal optimization problem is given below:

$$\mathcal{P}^* = \underset{\mathcal{P}\in S}{\operatorname{argmax}}\ I(\mathcal{P}) \text{ s.t. } C(\mathcal{P}) \leq B \text{ and } \mathcal{P}_{\text{H-sign}} = \mathcal{H} \quad (1)$$

where $\mathcal{P}$ is a path, S is the set of all obstacle free paths, $I(\cdot)$ is the information reward function, $C(\cdot)$ is the cost function for the budget B, and $\mathcal{H}$ is the desired H-signature of the optimization. For this work the path $\mathcal{P}$ is defined as a set of waypoints $(x_0, x_1, ..., x_n), x_i \in \mathbb{R}^2$. The cost function $C(\cdot)$ can have a wide range of possible functions such as energy of path, time, or distance. Additionally the information reward function $I(\cdot)$ can be different functions, such as summing directly over temperature or the gradient magnitude of the salinity field. The cost function and information field is selected from the language groundings.

Planning for the constrained information gathering problem is performed on the semantic homotopy augmented graph. Using this graph allows easy checking of the homotopy constraint added to the planning problem. As the informative path planning problem is an NP-hard problem [22], a Monte-Carlo Tree Search (MCTS) is used, to refine the search in the tree to areas of higher reward [8]. The MCTS is selected as expansion and rollout function can be forced to always provide paths that meet the desired constraints

In general each node of the search is defined as $n = (\text{location}, \text{h-signature})$, where the successors are every edge connected to the node. For the constrained search, expansion is only desired in directions which can still meet the homotopy and goal constraints. To check this constraint, a precomputed list of the lowest cost path which meets the constraints for every node, n is performed. This precomputation can be performed using Dijkstra's algorithm rooted at the goal node in $O(N \log N)$ time, where N is the number of nodes in the semantic homotopy augmented graph which grows with both the number of sampled points and the number of hazards in the environment.

During expansion of the MCTS algorithm each node can be checked to ensure that the cost of the path plus the shortest pre-computed path is less than the total budget. During the rollout stage of the MCTS algorithm, random actions are selected until $B < C(\mathcal{P}) + C(\mathcal{P}_{pre})$ and the tree from the last viable node plus precomputed path is returned as the full

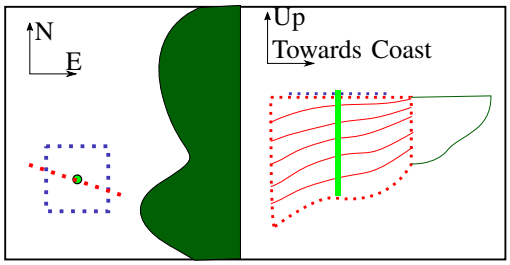


Fig. 4: Patch versus slice data representation. Blue dashed line is a patch centered on query point along surface of ocean. Red dashed line is a slice of the ocean with stretched terrain following lines.

sequence. This reduces the search space to areas of only viable expansion. Additionally, the approximate rewards from each rollout stage is a closer approximation of good rewards on that sub-tree due to the precomputed path forcing reasonable paths from the rollout function.

D. Automatic semantic feature detection

In order to have features for the planning and grounding framework, semantic features need to be detected from the environment. The key idea behind the automatic semantic feature detection is to extract salient scientific features, in our case upwelling front positions, from the environment. These scientific features are then used within the language grounding framework. Real-world upwelling fronts are subject to large amounts of noise making detection challenging. To perform the detection we compare the performance of a CNN and Support Vector Machine (SVM).

The network architecture we selected to perform the upwelling front detection is a small supervised CNN with three convolutional layers, and two fully connected layers. Each convolutional filter is 3x3 with no pooling between layers. The data for the network was salinity generated by the Regional Ocean Modeling System (ROMS) for the Oregon Coast [20]. We tried two different representations of the input, see Figure 4. The first representation is an overhead patch representation extracted from a grid of surface salinity. The second representation is coast oriented depth slices. These slices find the point nearest to the coast to the query point and orient the slice in that direction. This representation has the benefit of directly encoding the direction of the coast into the data making it both robust to orientation change and adding depth information. Both representations have a single label output indicating if the center of the slice/path is an upwelling front.

IV. Results

The results are broken into two sections: results for the grounding framework and results for the automatic upwelling front detector. To demonstrate the grounding framework, two results are shown. The first shows the number of valid robot plans generated from a list of language instructions. Second, example paths are generated by the grounding framework and gives examples of how paths with different constraints are generated. To demonstrate the upwelling front detection, a test dataset labeled by experts is classified using our approach.

TABLE I: Human-Generated Path Results

Type	Valid	Required Clarification	UD tree failures	Other Failures
Route Following	9 (69.2%)	0 (0%)	0 (0%)	4 (30.8%)
Information Gathering	3 (33.3%)	1 (11.1%)	2 (22.2%)	3 (33.3%)
Overall	12 (54.5%)	1 (4.5%)	2 (9.1%)	7 (31.8%)

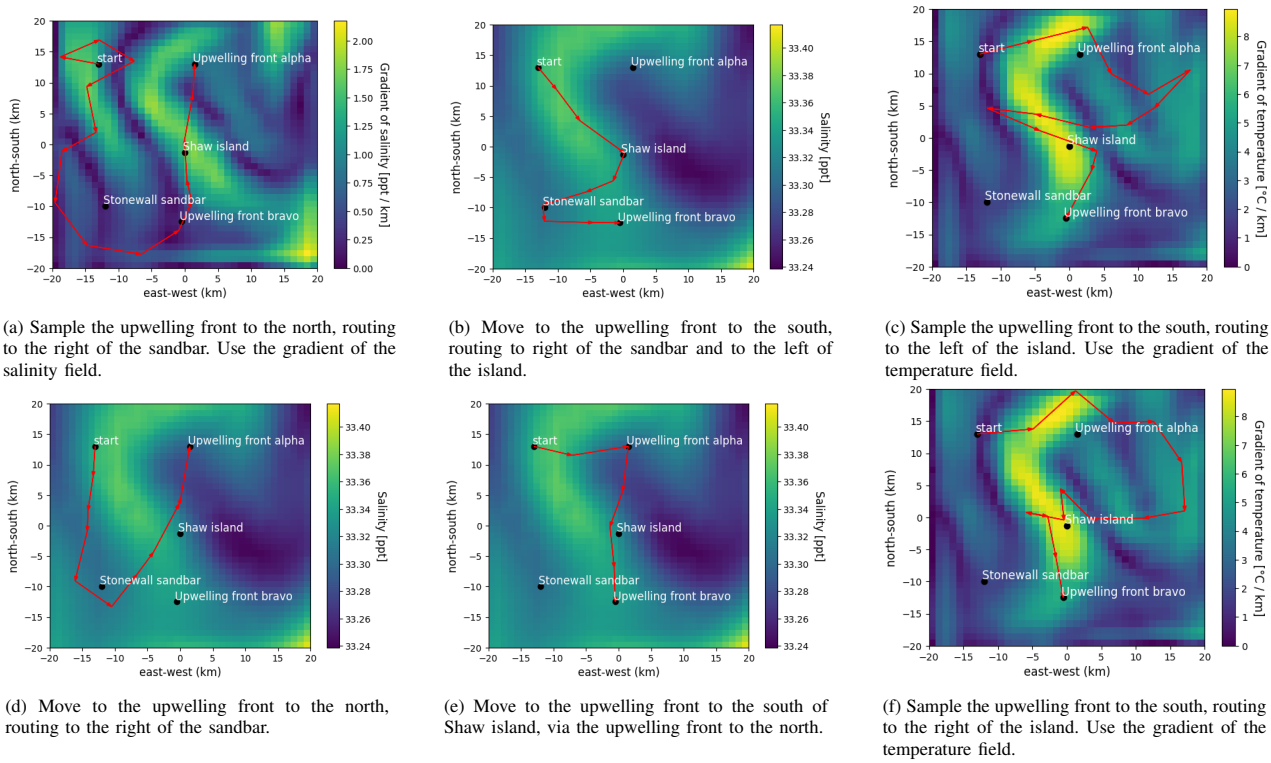
These results are compared with a SVM for both proposed data representations.

A. Grounding language instructions

To show the validity of the grounding framework twenty-two language instructions were collected from a mailing list of robot researchers at Oregon State University. The request asked for example mission descriptions from a given semantic map. Of the twenty-two total instructions, thirteen were route-following instructions, and the remaining nine were information-gathering instructions. Before providing the language instructions, each researcher was given simple written directions on creating valid language instructions using the environment shown in Fig. 5. Once collected, each instruction was evaluated to determine if the resultant path matched the constraints in the instruction. As can be seen in the results in Table I, overall we were able to successfully and accurately interpret over half of the language instructions.

We found that the framework works best on the relatively simpler route-following instructions. All of the failures our framework experienced on route-following instructions occurred when users tried to specify paths using words that our system lacked grounding rules for, such as “in between” and “at the midpoint of”. The information-gathering instructions were more complex, requiring the additional specification of an information objective function. This resulted in a decrease in the overall number of successful paths, with three valid paths generated, and a fourth generated following that prompted the user for additional clarification of an ambiguous grounding. The ambiguous grounding was automatically discovered by having multiple candidate groundings for a non-plural location, and requested the user to directly select a location. Of the failure cases, three were the result of missing grounding rules similar to the route-following failures, while one instruction included the phrase “that upwelling front” which could not be grounded. The final three failures were due to the Stanford parser outputting invalid UD trees, with two of the mistakes being misinterpreting compound words such as “current field” and “magnitude reward”.

Exemplar paths of mapping language instructions to robot plans are shown in Fig. 5. These paths demonstrate the same set of constraints for both information gathering tasks and move to tasks Fig. 5a and 5d. In Fig. 5b and 5e different move to commands are demonstrated with both via and homotopy constraints. Finally, two information gathering tasks, shown in Fig. 5c and 5f, were demonstrated with opposite homotopy constraints to show how the constraints impact the output



(a) Sample the upwelling front to the north, routing to the right of the sandbar. Use the gradient of the salinity field.

(b) Move to the upwelling front to the south, routing to right of the sandbar and to the left of the island.

(c) Sample the upwelling front to the south, routing to the left of the island. Use the gradient of the temperature field.

(d) Move to the upwelling front to the north, routing to the right of the sandbar.

(e) Move to the upwelling front to the south of Shaw island, via the upwelling front to the north.

(f) Sample the upwelling front to the south, routing to the right of the island. Use the gradient of the temperature field.

Fig. 5: Example paths generated from languages instructions. Figures (a) and (d) show the differences between the sample command and move command with the same constraints. Figures (b) and (e) show move to commands with different constraints. Figures (c) and (f) show the same sample command with only a different h-signature constraint.

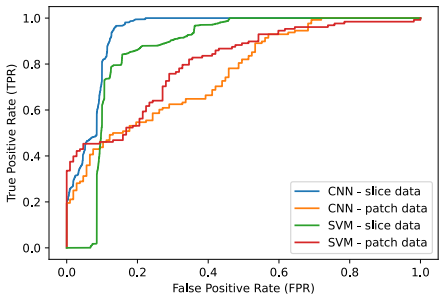


Fig. 6: ROC curves for SVM and CNN upwelling front detectors using the patch and slice data representations. Shows the CNN classifier on the slice data representation is the best.

information gathering plans.

B. *Upwelling front detector*

Expert labeled data is used to train and test the upwelling front detector The labeled data used to train the upwelling front detector came from several Slocum glider deployments done by Oregon State University between 2011 to 2013. The data collected was hand-labeled by expert oceanographers with upwelling front positions. We then extracted patches and slices from ROMS model output at each label. Since upwelling fronts occur relatively rarely in the ocean, there is a significant imbalance in the data. To prevent bias during training the minority upwelling front data was oversampled.

Our data was split into training, validation, and test sets. The test and validation sets were each a complete deployment chosen at random from all of the deployments in the dataset. Results of the upwelling front detector for both data representation are shown as Receiver Operator Characteristic (ROC) curves for the test data in Figure 6. These results show that the classifier performance relies on the data representation more than the particular classifier since representing the data as a slice data outperformed the patch representation for both classifiers.

V. Conclusion

In this paper, we have presented a framework for generating robot plans from language instructions. This framework is well suited for field robotic applications where acquiring a large corpus of language instructions to robot plan constraints is infeasible. By using a semantic homotopy augmented graph, we are able to generate robot plans following topological constraints derived from natural language instructions. Finally, we presented an automatic upwelling front detector which can detect the locations of upwelling fronts from ROMS model data. Combined with the semantic language groundings, this detector forms a complete system for information gathering plans to be generated from language instructions. We demonstrated this system with instructions collected from a set of novice users. Future directions for this work include improving the ability of the system to fail gracefully in the case of either over or under-specified language instructions. We would like to investigate ways of improving the language understanding of novel instructions without relying solely on hand-designed rules for applying groundings to the UD tree. Finally, we would like to investigate ways to handle temporal constraints beyond the current spatial constraints.

References

[1] Peter Anderson, Ayush Shrivastava, Devi Parikh, Dhruv Batra, and Stefan Lee. Chasing ghosts: Instruction following as Bayesian state tracking. In *Proc. Advances in Neural Information Processing Systems*, pages 369–379, 2019.

[2] Dilip Arumugam, Siddharth Karamcheti, Nakul Gopalan, Edward C Williams, Mina Rhee, Lawson LS Wong, and Stefanie Tellex. Grounding natural language instructions to semantic goal representations for abstraction and generalization. *Autonomous Robots*, 43(2):449–468, 2019.

[3] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3):273–290, 2012.

[4] Adam Borkowski, Barbara Siemiatkowska, and Jacek Szklarski. Towards semantic navigation in mobile robotics. In *Graph Transformations and Model-Driven Engineering*, pages 719–748. Springer, 2010.

[5] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, 2014.

[6] Thomas M Howard, Stefanie Tellex, and Nicholas Roy. A natural language planner interface for mobile manipulators. In *Proc. 2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6652–6659. IEEE, 2014.

[7] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Proc. IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[8] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proc. European Conference on Machine Learning*, pages 282–293. Springer, 2006.

[9] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *Proc. 2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 259–266. IEEE, 2010.

[10] Matteo Luperto, Alberto Quattrini Li, and Francesco Amigoni. A system for building semantic maps of indoor environments exploiting the concept of building typology. In *Robot Soccer World Cup*, pages 504–515. Springer, 2013.

[11] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. URL http://www.aclweb.org/anthology/P/P14/P14-5010.

[12] Seth McCammon and Geoffrey A Hollinger. Planning and executing optimal non-entangling paths for tethered underwater vehicles. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3040–3046, 2017.

[13] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[14] Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal dependencies v1: A multilingual treebank collection. In *Proc. Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, 2016.

[15] Daniel Nyga, Subhro Roy, Rohan Paul, Daehyung Park, Mihai Pomarlan, Michael Beetz, and Nicholas Roy. Grounding robot plans from natural language instructions with incomplete world knowledge. In *Proc. Conference on Robot Learning*, pages 714–723, 2018.

[16] Florian T Pokorny, Majd Hawasly, and Subramanian Ramamoorthy. Topological trajectory classification with filtrations of simplicial complexes and persistent homology. *The International Journal of Robotics Research*, 35 (1-3):204–223, 2016.

[17] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A python natural language processing toolkit for many human languages, 2020.

[18] Dushyant Rao, Mark De Deuge, Navid Nourani-Vatani, Stefan B Williams, and Oscar Pizarro. Multimodal learning and inference from visual and remotely sensed data. *The International Journal of Robotics Research*, 36 (1):24–43, 2017.

[19] Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[20] Alexander F Shchepetkin and James C McWilliams. The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Modelling*, 9(4):347–404, 2005.

[21] Amarjeet Singh, Andreas Krause, Carlos Guestrin, William J Kaiser, and Maxim A Batalin. Efficient planning of informative paths for multiple robots. In *Proc. International Joint Conference on Artificial Intelligence*, volume 7, pages 2204–2211, 2007.

[22] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34:707–755, 2009.

[23] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. Twenty-fifth AAAI Conference on Artificial Intelligence*, 2011.

[24] Mycal Tucker, Derya Aksaray, Rohan Paul, Gregory J Stein, and Nicholas Roy. Learning unknown groundings for natural language interaction with mobile robots. In *Robotics Research*, pages 317–333. Springer, 2020.

[25] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 6659–6668, 2019.

[26] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proc. 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[27] Yanwu Zhang, Michael A Godin, James G Bellingham, and John P Ryan. Using an autonomous underwater vehicle to track a coastal upwelling front. *IEEE Journal of Oceanic Engineering*, 37(3):338–347, 2012.

[28] Yanwu Zhang, Carlos Rueda, Brian Kieft, John P Ryan, Christopher Wahl, Thomas C O'Reilly, Thom Maughan, and Francisco P Chavez. Autonomous tracking of an oceanic thermal front by a wave glider. *Journal of Field Robotics*, 36(5):940–954, 2019.