Robotic Information Gathering using Semantic Language Instructions

Ian C. Rankin, Seth McCammon, and Geoffrey A. Hollinger

Abstract— **This paper presents a framework that uses language instructions to define the constraints and objectives for robots gathering information about their environment. Designing autonomous robotic sampling missions requires deep knowledge of both autonomy systems and scientific domain expertise. Language commands provide an intuitive interface for operators to give complex instructions to robots. The key insight we leverage is using topological constraints to define routing directions from the language instruction such as *'route to the left of the island.'* This work introduces three main contributions: a framework to map language instructions to constraints and rewards for robot planners, a topology constrained information gathering algorithm, and an automatic semantic feature detection algorithm for upwelling fronts. Our work improves on existing methods by not requiring training data with language instruction to planner constraint pairs, allowing new robotic domains such as marine robotics to use our method. This paper provides results demonstrating our framework producing correct constraints for 84.6% of instructions, from a systematically generated corpus of over 1.1 million instructions We also demonstrate the framework producing robot plans from language instructions for real-world scientific sampling missions with the Slocum underwater glider.**

I. INTRODUCTION

Designing information gathering missions for robots requires either setting reward functions, cost functions, and constraints for the planning algorithm or manually programming routes. These functions are typically challenging to define and require combined knowledge of autonomy systems and the application domain. Additionally, routes hand-designed by humans is time-consuming [1] and often leads to sub-optimal information gathering routes [2]. In this paper, we propose a framework which allows the user to set mission goals, constraints, and reward functions more naturally using language instructions. This framework can be applied to any field robotic domain that semantic features can be extracted from the environment. However, we focus on the marine robotics task of studying coastal upwelling fronts. These fronts are locations where cold, nutrient rich, and deep water is pushed to the surface. The mixing of the warm surface water with the deep ocean water and currents caused by the fronts are of interest to both biological and physical oceanography [3]. Our framework allows complex language instructions for information gathering tasks; an example command and planned path is given in Figure 1.

This research was funded in part by NSF grants IIS-1723924 and IIS-1845227. Authors are with the Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis OR, United States {rankini, mccammos, geoff.hollinger}@oregonstate.edu. Thanks to Pat Welch from the College of Earth, Ocean, and Atmospheric Sciences for providing the ocean glider mission descriptions.

The main challenge in generating robot plans from language instructions is grounding the language to real features and actions in the environment. Language instruction grounding is broadly defined as mapping language instructions to real-world robot plans [4]. This requires identifying salient features in the environment then mapping language instructions to these features. Previous works have performed this grounding, but are designed for indoor environments with well defined features, and require a large corpus of training data. Neither of these exist in field robotic domains.

We separate the language understanding task and the detection of semantic features into separate problems to simplify each solution. We ground language instructions without requiring a corpus of mappings from instructions to robot plan constraints. This mapping is done using the dependency information contained in Universal Dependency (UD) trees, a parse of natural language sentences [5]. The robot plan constraints define the robot action, reward function, information field, and route constraints. We use homology constraints [6] to define the robot path as they lend themselves to language routing instructions such as '*route to the left of feature A'*. These homology constraints prescribe the information gathering paths to trajectories in distinct homology classes.

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 homology-constrained information gathering algorithm which uses Dijkstra's algorithm to precompute an h-augmented graph, a graph augmented with topological information [6], 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). Initial work from this paper was first presented in a related workshop paper by the authors [7]. This work extends the previous work by expanding the results of the language grounding framework.

II. RELATED WORKS

A. Language grounding

One area where there has been a significant amount of research on grounding language instructions to robot plans is embodied AI. This field studies visually-grounded navigation instruction following and question answering [8], [9], and combine the language understanding and feature recognition into a single framework. However, the semantic grounding techniques commonly used in embodied AI systems focus on

end-to-end deep learning methods which require a significant amount of training data. This training data is gathered from photorealistic simulated indoor environments allowing the algorithms to train over thousands to millions of iterations [9]. Unfortunately, it can be difficult to collect the large quantities of data or simulation needed to train these networks from field robotics applications. This makes current embodied AI algorithms infeasible for these environments.

An alternative way to ground language instructions is to use natural language dependency parsers, such as the Stanford Parser [10], to find the structure of the language instructions before grounding. Dependency parsers extract a UD tree [5], which decomposes a sentence into its component grammatical structures. In a UD tree each word has a tag describing its dependency relationship, such as object or determiner, and the root of the relationship. These parsers use a Probablistic Context-Free Grammar (PCFG) model to predict the dependencies [11]. Recent dependency parsers, such as the Stanford Parser [10], use neural-networks to model the PCFGs, which predict a dependency tag for each input word. In our work we use the Stanza library [12], an implementation of the Stanford Parser, to generate the dependency trees which uses a bi-directional LSTM model proposed by Qi et al. [13].

Previously the shape of the dependency tree has been used to inform the language grounding process [14], [15]. More recent works have focused on grounding language instructions using neural networks for mapping semantic features to language instructions [16], mapping instructions to skills [17], and learning unknown object groundings [18]. These methods all depend on large libraries of training data in order to operate effectively. The closest work to our proposed approach is Howard et al. [19], which uses a probabilistic graphical model with the shape generated from the UD tree to construct robot plan constraints from natural language instructions. One common element across natural language mission descriptions is defining the mission relative to features in the environment using relational terms such as *'around'*, *'via'*, or *'north of'*. By directly encoding the groundings of these terms, we can produce groundings without requiring a training corpus.

B. Semantics and Homology

We use semantic features to provide an interface between the grounding framework and the robotic planner. Semantic maps and features have been used in robotics in indoor environments extensively [20], [21], [22]. 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, in our domain scientifically significant ocean features such as upwelling fronts provide nameable semantic features for planning.

In this paper we use homology constraints to allow the user control over the path the robot takes rather than only the start and goal locations. Two robot paths are defined to be in the same homology class if they share start and end points and they form the boundary of an area that does not contain or intersect with any of the obstacles [6]. Previous works have used homology classes during robot planning [23], [24]. Bhattacharya et al. [6] shows using an h-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 h-augmented graph to the informative path planning problem. Although previous works have used topology to inform robot motion planning [6], [25], and for informative path planning [23], [26], the desired topological classes are unambiguous and known. In this work we relax this assumption, and construct topological route constraints from language instructions.

III. SEMANTIC FEATURE DETECTION

In order to generate features for the planning and grounding framework, semantic features need to be extracted from the environment. Some of these features such as island, cities, or shipping lanes can come from maps; however, others need to be detected from the environment. In our marine robotics domain, we want to extract coastal upwelling fronts from the environment. These salient scientific features can then be used as semantic features within our grounding framework.

Real-world upwelling fronts are subject to large amounts of noise, making detection challenging. We propose a CNN approach to detect the upwelling fronts. Since we have small amounts of labeled data from glider missions performed off the Oregon coast, we selected a small supervised CNN, with three convolutional layers, and two fully connected layers, for our network architecture. We used salinity generated by the Regional Ocean Modeling System (ROMS) for the Oregon Coast for our training data [27]. We tried two different representations of the input. 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. GROUNDING LANGUAGE INSTRUCTIONS

After we have a list of semantic features, we ground the language instruction by finding a mapping from the language instruction to a robot planner constraints. An example of a grounding in the phrase *"Sample the upwelling front, routing to the east 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 we use the Stanford Parser to create a Universal Dependency (UD) tree [28], [5]. Universal dependencies is a language-agnostic framework for annotation of grammar. We use these annotations to determine groundings for actions, physical features, modifiers of features, and path constraints.

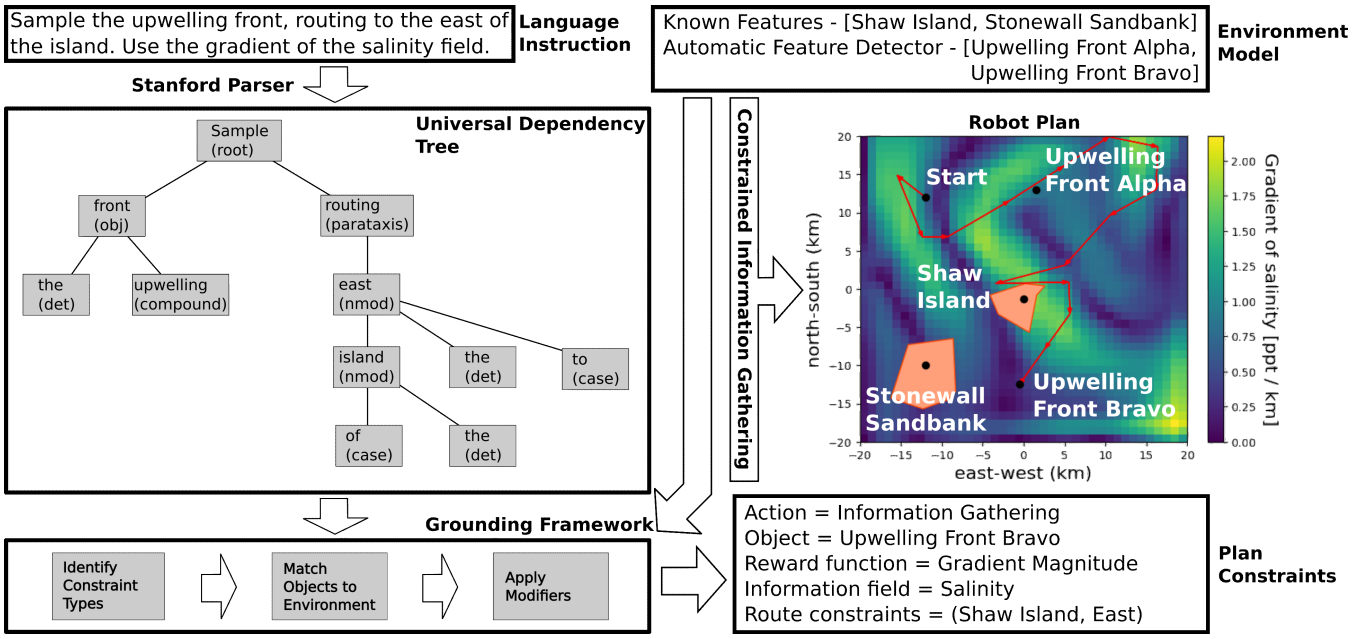


Fig. 1: System diagram for generating a full robot plan from the command *"Sample the upwelling front, routing to the east of the island."* The instruction uses the Stanford parser to generate a UD tree. We use our grounding framework to generate a list constraints from the UD tree and the list of semantic features from either a known list or using the automatic feature detector. The constrained information gathering algorithm takes the planner constraints and semantic features of the environment and generates robot plans.

A. *Identifying Constraint Types*

Each node in the UD tree produced by the parser represents a word in the sentence, with a dependency relationship to the word's parent. The dependency encodes 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.
- **Adjective 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 that require being ground to features in the environment.
- **Compound:** Compounds 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. Specifically, this lookup maps a word to one of multiple types of semantic constructs: object features in the environment, actions (e.g. Move, Sample), modifier (e.g. Left, North), adposition (e.g. to, from), argument (e.g. routing), or objective (e.g. gradient magnitude). In practice, most nodes are attached to their parents modifying the grounding of objects or actions.

The root of the UD tree is the action word whose grounding is the particular action planner that is used to plan the final robot behavior (e.g. route following planner or information gathering planner). The remainder of the groundings provide this planner with its objectives and constraints. For example in the UD tree shown in Figure 1, *'routing to the east of the island'* imposes a constraint on the planner, while *'the upwelling front'* provides the objective.

B. *Match objects to environment*

To fully ground a language instruction, a mapping is required between each **obj / obl / nmod / conj / parataxis** node in the tree and the specific physical feature or set of features in the environment it refers to. This mapping is performed by iterating through the tree using a breadth first search and matching each word to a list of possible groundings. 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 synonym set, which has a semantic meaning behind each keyword [29]. Wordnet is a lexical database of English words, where each word is grouped into sets of synonyms. During a search for a given word (e.g. *'isle'*), the algorithm first searches through the list to find exact matches of the name. Failing to do so, the algorithm then looks for exact keyword matches, (e.g. looking for the word *'island'*, would find all groundings with the keyword *'island'*). If this search also fails, the algorithm uses the Resnik [30] revised Wu and Palmer [31] method of measuring semantic relatedness on the synonym set. If this similarity is greater than a 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 without using the test set of instructions.

C. *Apply Modifiers*

After mapping words groundings in the environment, 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 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, such as our example instruction, the system enters a clarification state which asks the user to choose which of the list of possible groundings is correct. This failure mode can occur either if the framework misunderstands something or if the user does not fully specify the grounding. The result of applying groundings to the parsed instruction is a set of relational constraints on actions (e.g. *'to the east of the island'*).

D. Preprocessor

The ability of our grounding framework to correctly ground a sentence is reliant on the ability of the parser to construct the correct UD tree from the language instruction. To improve the parser's performance, we perform a set of systematic find-and-replace preprocessing steps using regex. The first of these is (*'routing'* $\rightarrow$ *'routing to the'*). The second is replacing robot names, which often contain numerals which confuse the parser, with named tokens which the parser recognizes as named entities. Because we use topological constraints, which do not restrict distance, all references to metric distances are removed from the statements.

V. HOMOLOGY CONSTRAINED PLANNING

A. Semantic Topology Augmented Graph

An intuitive way to apply the constraints produced by the grounding framework to paths within the planning domain is to understand the constraints as identifying a particular homology class of trajectories between the robot's current position and its goal. To build a plan for the robot using these constraints, we turn to homology augmented graphs [6]. The homology augmented graph expands a roadmap, such as a Probabilistic Roadmap (PRM) [32], in $\mathbb{R}^2$ with an additional dimension that contains information about the homology class of trajectory required to arrive at a particular vertex. The homology information is encoded in a h-signature, a variable which uniquely describes the topological class of a given trajectory belongs to. The h-signature used in this work is a discrete version of the continuous winding number used in Bhattacharya et al. [25]. Our homology signature uses the same parallel non-intersecting rays as Kim et al. [33] for the discrete winding number, which means each obstacle either has a $\{+\}$, $\{0\}$, or $\{-\}$ signature.

In our planning problem, topological features can fall in one of two classes: points of interest and hazards. Depending on the mission specification, a geographic or oceanographic feature can fall in either category (though never both at the same time). Hazards are features that should be avoided, and that act as the obstacles in the environment which partition it into different topological trajectory classes. Points of interests are named points of our mission, and must be included as vertices of the h-augmented graph. Features are identified as points of interest or hazards by the type of planner constraint that references them: hazards for routing constraints, points of interest for destinations. While generating the h-augmented graph we only use features listed in the the instruction, so we can guarantee the results of the mapping to be topologically unique classes.

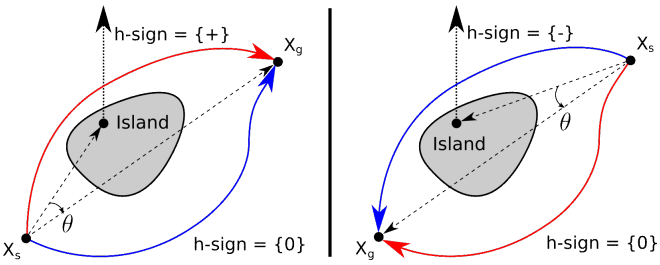


Fig. 2: H-signatures for different start, X_s, and goal, X_g, locations and left or right mappings. Red paths are *'right'* ($\theta \leq 0$), blue paths are *'left'* ($\theta > 0$).

When planning with a topological constraint, the first step is to construct our h-augmented graph. We begin this process by adding all points of interest as vertices, and then transition to randomly sampling the environment to construct a PRM. Then, we construct our h-augmented graph using the method described in [6], using the hazards as the representative points of obstacles. Once the h-augmented graph is constructed, we translate the constraints from the mission description into a desired h-signature. We first map absolute direction (e.g. north or south), to the relative directions left and right, based on the robot's direction of travel. Then, we need to determine what the appropriate h-signature is for the relative directions. As shown in Figure 2, the h-signature is guaranteed to be $\{+\}$ for left instructions and $\{-\}$ for right instructions, since they will always cross the reference ray left-to-right and right-to-left, respectively. However, the $\{0\}$ homology signature is ambiguous and could map to either direction. We determine which mapping to use by checking the h-signature of the straight line path between the start and goal and determining if that path is to the *'left'* ($\theta > 0$) or *'right'* ($\theta \leq 0$) of the obstacle. If the straight line path has the h-signature $\{0\}$, then the mapping is clear. Otherwise, if it is to the left with signature $\{+\}$, then $\{0\}$ must be to the right, and vice-versa if it is to the right with signature $\{-\}$.

B. Constrained information gathering algorithm

In this paper we use a modified formulation of the Informative Path Planning problem [34]. 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.

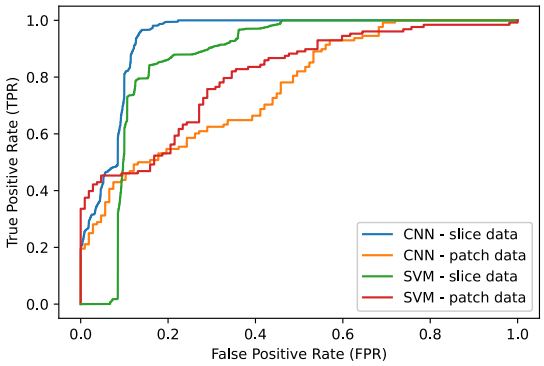


Fig. 3: ROC curves for SVM and CNN upwelling front detectors using the patch and slice data representations.

Planning for the constrained information gathering problem is performed on the semantic h-augmented graph. This graph allows easy checking of the homology constraint during expansion of the search tree. As the informative path planning problem is an NP-hard problem [35], we use a Monte-Carlo Tree Search (MCTS) to refine the search of the tree to areas of higher reward [36]. We use MCTS, since its expansion and rollout function can be constrained to always produce paths that meet the desired routing constraints.

To ensure the homology constraint is met, a precomputed list of the lowest cost path in each homology class is used. We perform this precomputation with Dijkstra's algorithm rooted at the goal node. During the MCTS tree expansion, we only expand to nodes where the cost of the path plus the shortest precomputed path of the given homology class is less than the budget. In the rollout stage of the MCTS algorithm, random actions are selected until $B < C(\mathcal{P}) + C(\mathcal{P}_{pre})$, where $\mathcal{P}_{pre}$ is the shortest path that satisfies all constraints from the precomputed list. This reduces the search space to areas of only viable expansion. Additionally, the approximate rewards from each rollout stage is a better approximation of rewards on that sub-tree due to the precomputed path forcing reasonable paths from the rollout function.

VI. RESULTS

The results are broken into three sections: results for the automatic upwelling front detector, results for the grounding instructions, and end-to-end results for the grounding framework and planner. We use a test dataset of upwelling fronts labeled by experts to verify the detection algorithm. To validate the grounding framework we tested it on a library of 1.1 million systematically generated language instructions. Finally, end-to-end results of instructions for real-world marine robotic sampling missions performed at Oregon State University (OSU) are grounded to a full robot plan. Comparison methods are not used in the language grounding task because to the best of the authors knowledge there are no existing methods which can ground language instructions to field robotic sampling tasks.

A. Upwelling front detector

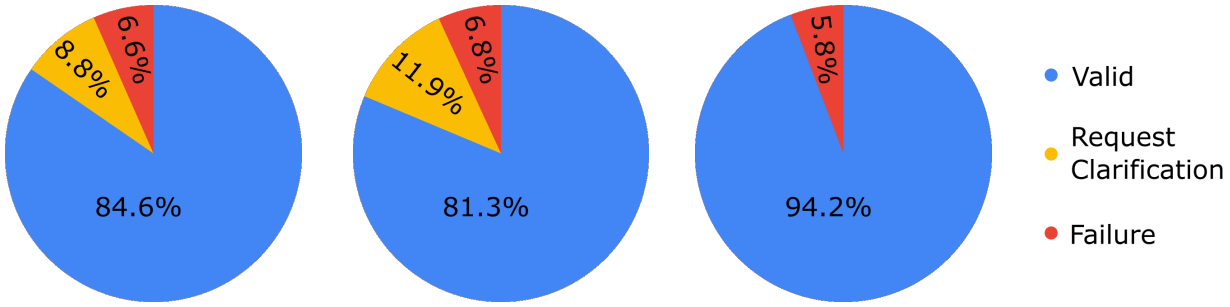
Expert labeled data is used to train and test the upwelling front detector. The labeled data used to train the upwelling front detector came from 7 Slocum glider deployments done by OSU between 2011 to 2013. The data collected was hand-labeled by an expert researcher in OSU's College of Earth Ocean, and Atmospheric Sciences (CEOAS) with upwelling front positions. We then extracted surface patches and depth slices from the ROMS model output at each label. Since upwelling fronts occur relatively rarely in the ocean, there is an 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 the entire dataset. We compare the performance of our proposed CNN and Support Vector Machine (SVM) as baseline method using both the patch and slice representations. Results of the upwelling front detector are shown as Receiver Operator Characteristic (ROC) curves for the test data in Figure 3. 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. This can be attributed to the slice data better representing the data by providing the classifier rotation invariance to the coast line, as well as the additional information added through the depth data.

B. Grounding language instructions

We test the language grounding framework by finding the planner constraints of 1.1 million automatically generated language instructions from phrasal templates and checking to the true planner constraints. An example phrasal template is *"{move_action} to {feat_1} by going to the {dir} of {feat_2}"* where each variable in brackets is replaced from a list of possibilities. During the procedural generation of the language commands, we also record the corresponding ground truth plan constraints. We use 10 phrasal templates that are representative of common language instructions in our domain of scientific sampling missions to generate a testing library of 1.1 million language instructions and their corresponding constraints. This process involves iterating through all possible combinations of features, directions, action words, reward functions, and information objective types for the map shown in Figure 1.

We compare the ground truth constraints with those produced by our grounding framework. The results from this test are shown in Figure 4. These results show a low failure rate of 6.51%. In cases where the grounding fails to produce correct results, the most common cause of failure was an incorrect UD tree produced by the parser. An example failed instruction is *"Move to upwelling front alpha, routing to the left of upwelling front bravo and routing to the right of Shaw island."* The grounding framework failed to produce the correct constraints for this instruction because the parser incorrectly assigns *'to the right of Shaw Island'* as a modification redundantly clarifying the position of upwelling front



(a) All instructions; 1 149 576 total instructions.

(b) Route following instructions; 855 000 total instructions.

(c) Information gathering instructions; 295 576 total instructions.

Fig. 4: Accuracy of groundings from phrasal templates. Figure (a) shows all instructions from Figures (b) and (c). Request Clarification uses the method outlined in Section IV-C

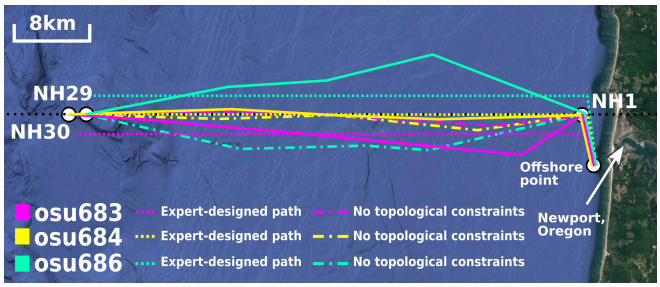


Fig. 5: Full system results compared with expert designed routes and paths without topological constraints. The solid lines using topological constraints are closer to the expert-designed paths. Instructions for these paths are given in Section VI-C.

bravo rather than assigning it as a new routing instruction. Future work could potentially handle parser errors using a machine learning approach trained on synthetic data to generate constraints from the parse tree.

C. Full system results

We validate the full system using several mission description of real ocean glider missions undertaken at OSU provided by scientists in the OSU CEOAS. Three instructions for a robotic sampling mission were given for different ocean gliders to be performed at the same time. These three instruction are shown verbatim below, see Figure 5.

- *Fly osu683 2km south of the NH line starting at NH1 and going to NH30, with the initial waypoint being the offshore point.*
- *Fly osu684 on the NH line starting at NH1 and going to NH29.5, with the initial waypoint being the offshore point.*
- *Fly osu686 2km north of the NH line starting at NH1 and going to NH29, with the initial waypoint being the offshore point.*

The NH line is a line of constant latitude at $N44°39.2'$, NH1 is on the NH line 1 nautical mile from the Oregon Coast line, and NH30 is on the NH line 30 nautical miles from the coast line. An issue with these language instructions being converted directly to our planning framework is the distance above or below the NH line desired. As mentioned previously in Section IV-D, our routing framework produces topological routing constraints, which do not encode distance.

The resultant paths from the first three language instructions are shown in Figure 5. These paths do not exactly follow the parallel plans described by instructions, but they generate a similar path that fits in the context of the topological constraints used by our framework. We believe the paths using topological constraints meet the spirit of the provided instructions with metric constraints. These results show that our method works end-to-end, generating constraints from a set of language instruction and outputting a robot executable plan for a real-world robotic sampling mission.

While our framework is able to generate paths for the above set of instructions. We also received an additional set of three instructions. One path is shown verbatim below.

- *Fly osu683 roughly parallel to the 400m isobath for a line 4km long centered on the NH line.*

Our system fails to ground these instructions to a path since it has no notion of isobathymetry, and does not understand placing a route centering an isobathymetry line at a line of constant latitude. We leave this as an avenue for future work.

VII. 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 h-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 the system correctly grounding over 970 000 unique language instructions, as well as grounding instructions for a real robotic sampling missions performed by OSU into executable robot paths. In future work we would like to investigate ways of improving the language understanding of novel instructions without relying solely on hand-designed rules on the UD tree. These could help alleviate the errors in grounding caused by the Stanford Parser. We would also like to investigate an automatic method for understanding novel words without a predefined lookup of possible groundings. Finally, we would like to investigate ways to handle temporal constraints beyond the current spatial constraints.

REFERENCES

[1] T. Somers, N. R. J. Lawrance, and G. A. Hollinger, "Efficient learning of trajectory preferences using combined ratings and rankings," in *Proc. Robotics: Science and Systems Conference Workshop on Mathematical Models, Algorithms, and Human-Robot Interaction (RSS), Boston, MA*, 2017.

[2] T. Somers and G. A. Hollinger, "Human–robot planning and learning for marine data collection," *Autonomous Robots*, vol. 40, no. 7, pp. 1123–1137, 2016.

[3] Y. Zhang, M. A. Godin, J. G. Bellingham, and J. P. Ryan, "Using an autonomous underwater vehicle to track a coastal upwelling front," *IEEE Journal of Oceanic Engineering*, vol. 37, no. 3, pp. 338–347, 2012.

[4] D. Arumugam, S. Karamcheti, N. Gopalan, E. C. Williams, M. Rhee, L. L. Wong, and S. Tellex, "Grounding natural language instructions to semantic goal representations for abstraction and generalization," *Autonomous Robots*, vol. 43, no. 2, pp. 449–468, 2019.

[5] J. Nivre, M.-C. De Marneffe, F. Ginter, Y. Goldberg, J. Hajic, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira *et al.*, "Universal dependencies v1: A multilingual treebank collection," in *Proc. International Conference on Language Resources and Evaluation (LREC)*, 2016, pp. 1659–1666.

[6] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, 2012.

[7] I. C. Rankin, S. McCammon, and G. A. Hollinger, "Optimized robotic information gathering using semantic language instructions," in *Proc. Robotics: Science and Systems Conference Workshop on Robots in the Wild: Challenges in Deploying Robust Autonomy for Robotic Exploration (RSS), Virtual*, 2020.

[8] P. Anderson, A. Shrivastava, D. Parikh, D. Batra, and S. Lee, "Chasing ghosts: Instruction following as Bayesian state tracking," in *Proc. Advances in Neural Information Processing Systems*, 2019, pp. 369–379.

[9] E. Wijmans, S. Datta, O. Maksymets, A. Das, G. Gkioxari, S. Lee, I. Essa, D. Parikh, and D. Batra, "Embodied question answering in photorealistic environments with point cloud perception," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6659–6668.

[10] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 740–750.

[11] S. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Prentice Hall, 2002.

[12] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A Python natural language processing toolkit for many human languages," *arXiv preprint arXiv:2003.07082*, 2020.

[13] P. Qi, T. Dozat, Y. Zhang, and C. D. Manning, "Universal dependency parsing from scratch," in *Proc. Conference on Computational Natural Language Learning (CoNLL)*, 2019.

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

[15] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *Proc. AAAI Conference on Artificial Intelligence*, 2011, pp. 1488–1493.

[16] D. Nyga, S. Roy, R. Paul, D. Park, M. Pomarlan, M. Beetz, and N. Roy, "Grounding robot plans from natural language instructions with incomplete world knowledge," in *Proc. Conference on Robot Learning*, 2018, pp. 714–723.

[17] N. Gopalan, E. Rosen, G. Konidaris, and S. Tellex, "Simultaneously learning transferable symbols and language groundings from perceptual data for instruction following," in *Proc. Robotics: Science and Systems (RSS), Virtual*, 2020.

[18] M. Tucker, D. Aksaray, R. Paul, G. J. Stein, and N. Roy, "Learning unknown groundings for natural language interaction with mobile robots," in *Robotics Research*. Springer, 2020, pp. 317–333.

[19] T. M. Howard, S. Tellex, and N. Roy, "A natural language planner interface for mobile manipulators," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6652–6659.

[20] A. Borkowski, B. Siemiatkowska, and J. Szklarski, "Towards semantic navigation in mobile robotics," in *Graph Transformations and Model-Driven Engineering*. Springer, 2010, pp. 719–748.

[21] M. Luperto, A. Q. Li, and F. Amigoni, "A system for building semantic maps of indoor environments exploiting the concept of building typology," in *Robot Soccer World Cup*. Springer, 2013, pp. 504–515.

[22] J. Fasola and M. J. Mataric, "Using semantic fields to model dynamic spatial relations in a robot architecture for natural language instruction of service robots," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 143–150.

[23] S. McCammon and G. A. Hollinger, "Planning and executing optimal non-entangling paths for tethered underwater vehicles," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3040–3046.

[24] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Topological trajectory classification with filtrations of simplicial complexes and persistent homology," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 204–223, 2016.

[25] S. Bhattacharya, R. Ghrist, and V. Kumar, "Persistent homology for path planning in uncertain environments," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 578–590, 2015.

[26] S. McCammon and G. A. Hollinger, "Topological hotspot identification for informative path planning with a marine robot," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4865–4872.

[27] A. F. Shchepetkin and J. C. McWilliams, "The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model," *Ocean Modelling*, vol. 9, no. 4, pp. 347–404, 2005.

[28] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

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

[30] P. 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*, vol. 11, pp. 95–130, 1999.

[31] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proc. Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1994, pp. 133–138.

[32] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[33] S. Kim, S. Bhattacharya, and V. Kumar, "Path planning for a tethered mobile robot," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 1132–1139.

[34] A. Singh, A. Krause, C. Guestrin, W. J. Kaiser, and M. A. Batalin, "Efficient planning of informative paths for multiple robots," in *Proc. International Joint Conference on Artificial Intelligence*, vol. 7, 2007, pp. 2204–2211.

[35] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[36] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Proc. European Conference on Machine Learning*. Springer, 2006, pp. 282–293.