2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Expressing Homotopic Requirements for Mobile Robot Navigation through Natural Language Instructions

Daqing Yi[1], Thomas M. Howard[2], Michael A. Goodrich[1] and Kevin D. Seppi[1]

***Abstract*— Allowing a human to express topological requirements to a robot in language enables untrained users to guide robot movement without requiring the human to understand sophisticated robot algorithms. By using a homotopy class or classes to represent one or more topological requirements, we build a framework that helps a robot understand a human's intent. This paper reviews a homotopic decomposition method that is used to convert any path into a string, which allows homotopic path equivalence to be performed by comparing strings. We then integrate the Homotopic Distributed Correspondence Graph (HoDCG) to infer the homotopic constraint in the format of strings from a language instruction. Finally, we use a homotopic path-planning algorithm that finds the optimal paths for a given objective and homotopic constraint. Experiment results show how a language instruction is converted into a path driven by an implicit topological requirement.**

I. Introduction

Language-based interactions between a human and a robot theoretically extend the scope of interaction from only trained users to anyone who can use language. Language-based interactions require that a robot can understand what a human supervisor intends when he or she describes a task. Since humans are ostensibly good at high-level spatial reasoning, telling a robot where to move and where to avoid is a direct and efficient way for a human to express intent in assigning a task. With proper algorithmic support, humans need not transform the human-like high-level information into robot-based quantitative models for robot path-planning. We address this by allowing a human supervisor to specify a path topology and describe it to a robot. For example, a human could say "go to the left of the fountain and on to the hospital" in order to avoid the ambulance in Figure 1a, and "go between the hotel and the shop on the way to the hospital" in order to keep away from the traffic in Figure 1b.

Such constraints can be expressed by, for example, having a human draw a path on a map, but it is desirable to explore other ways of expressing constraints. One advantage of using language to express constraints is that a human could think and express without converting into a graphical or robotic perspective. Allowing the human to use language to express topological constraints requires that a robot is able to understand an abstract topological description and then plan a path that honors the desired topology. In this paper, a topological constraint for robot path-planning is derived from a language instruction. The topological constraint defines a set of eligible paths that have the required topology shape. The path-planning problem is finding the optimal path within this set of paths.

(a) Left of the fountain (b) Between shop and hotel

Fig. 1: Topological requirement for navigation

The mathematical notion of a homotopy is widely used in defining a topology constraint of paths because it defines the similarity between paths within the same path topology. When a path can be deformed into another path without encroaching into any obstacle, the two paths are homotopic [1]. In path-planning, when the start position and the end position are constrained, all the feasible paths can be classified into homotopy classes [1], each of which includes all the paths that are homotopic. In this paper, we explore the spatial relations between paths and objects in a world. By considering objects as obstacles, we can represent a topological requirement by a homotopy class or a set of homotopy classes. We express a topological requirement in a problem of instructing robotic navigation by

- translating a language instruction into a homotopic requirement; and
- planning a path subject to the homotopic requirement.

II. Related Work

There are a few approaches to helping a robot understand the topological information in a language instruction. Understanding the requirement of an instruction for execution planning depends on grounding the spatial information according to the phrases [2]. An execution plan is inferred from the grounded information. A semi-structured grammar, e.g. Tactical Behavior Specification [3], [4], is proposed as a guide for a human to express commands that a robot can understand. The grounded spatial constraint from a command is integrated into path-planning algorithms to get paths. The language understanding is extended to all types of natural language. Language examples are used to train semantic parsers to extract intents [5], [6]. Execution plans

[1]Daqing Yi, Michael A. Goodrich and Kevin D. Seppi are with Computer science Department, Brigham Young University, Provo, UT 84604, USA yi@byu.edu, mike@cs.byu.edu, kseppi@byu.edu

[2]Thomas M. Howard is with Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627, USA thoward@ece.rochester.edu

could be obtained by applying the extracted intents on the world. For better generalization, graphical models are used to model the relationships between phrases and groundings [7]. In [8], DCG (Distributed Correspondence Graph) grounds implicit constraints, which model the correspondences between phrases and groundings. The correspondences between phrases and groundings can be factorized by conditional independence assumption. A path-planning problem can be created to generate a path that satisfies an inferred constraint. It has been successfully applied in spatial-constraint inference from instructions [8].

As we are interested in inferring homotopic constraints, we need a format of representing a homotopy class for identification. The homotopy class identification depends on the recognition of the spatial relation between obstacles and a path. Homology, a different type of similarity, is used as an approximation to homotopy. In [1], a map with obstacles is modeled into a complex plane with undefined points. Then the homology of paths can be recognized by comparing the complex integral values based on Cauchy theorem. Decomposition is the most common approach to identifying the homotopies of paths. Voronoi diagram, the classic decomposition method, is introduced to decompose a map that generates a topology structure of decomposed disjoint regions. Different walks on this structure derive reference paths of different homotopy classes [9]. Paths of diverse homotopy classes can be obtained by deforming the reference paths. Another way is using a set of line segements that decompose a map as reference frames [10]. How a path sequentially crosses the reference frames can be used to represent its topological shape.

In addition to a homotopic constraint, a human also intends a specific objective in a language instruction. Most motion-planning algorithms for homotopic constraints can only find feasible paths [10], [9] or the shortest paths [11], [12]. We also include an objective in planning a path, which leads to an optimal path-planning problem.

In this paper, we propose a framework for language-guided motion-planning algorithm, which translates a language instruction into a path-planning problem and finds the optimal path accordingly. We describe the framework in Section III, and provides experiment results in Section IV to support its performance.

III. A Framework of Language-Instructed Path-Planner

We propose a language-instructed path-planning system that can understand the homotopic requirement of a language instruction and generate the optimal path subject to the homotopic requirement. Figure 2 shows the structure of the system. The Homotopic Distributed Correspondence Graph (HoDCG), discussed in Section III-A, infers spatial relations from a language instruction. A map is decomposed according to the spatial relations in Section III-B, using a homotopic Deterministic Finite Automata (homotopic DFA) [13] to convert paths into string representations that honor topological similarities. The spatial relations are then used to

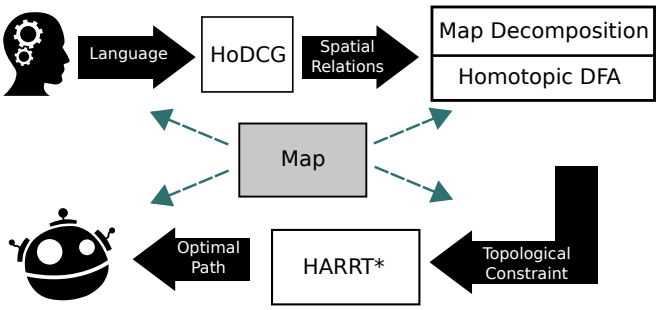


Fig. 2: System Framework

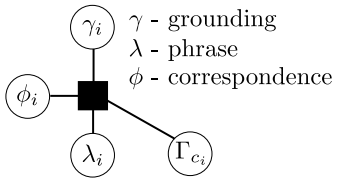


Fig. 3: Factor model

generate rules that define *qualified reference frame sequences* in Section III-C; these sequences implicitly define a grammar of eligible strings. The joint set of this eligible string grammar and the grammar defined by the homotopic DFA is the inferred homotopic constraint. The Homotopy-Aware RRT* (HARRT*) [13], a homotopic path-planner, is then used to find the optimal path in the homotopic constraint in Section III-D.

A. Inferring Spatial Relations from Language

In this section, we adopt a graphical model that grounds spatial relations from a language instruction. The spatial relations are used to derive a homotopic constraint. The homotopic constraint represents a human supervisor's requirement of path topology.

Our graphical model is derived from the Distributed Correspondence Graph (DCG) [8], which is a factor graph that can efficiently ground language sentences. The graph consists of a set of factor models. Each factor model defines a relationship among a correspondence ϕ_i, a phrase λ_i, a grounding γ_i, and a set of child elements Γ_{c_i}. A factor model is visualized in Figure 3.

Essentially, such a factor graph builds a distribution that represents the correspondences between groundings and a sentence. This distribution depends on a phrase structure that is parsed from a sentence by a grammar parser. Figure 4 gives an example of a parse tree from a sentence "walk by the left of the table". The parse tree determines the factor graph. The factor graph (Figure 5) is created by assembling the correspondences between groundings and phrases (Figure 3) according to the parse tree (Figure 4). The inference of the factor graph returns a set of groundings. The details are stated in [8].

We propose HoDCG (Homotopic Distributed Correspondence Graph) algorithm that extends DCG for supporting homotopic requirements. HoDCG includes new types of

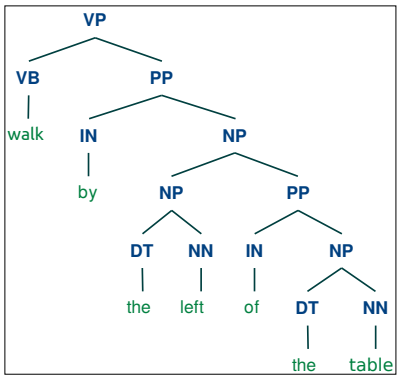


Fig. 4: "walk by the left of the table"

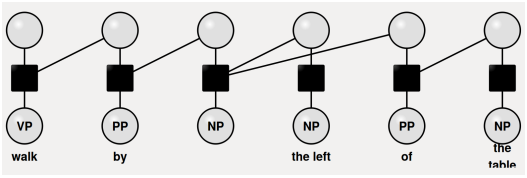


Fig. 5: HoDCG structure of "walk by the left of the table".

groundings to represent homotopies. Inferred groundings are organized to construct a homotopic constraint by phrase structure information.

HoDCG includes a graph model that supports spatial relations as new groundings and a mechanism that derives homotopic constraints from associated groundings. Figure 5 illustrates a structure of HoDCG, which extends the phrase structure in Figure 4. Given the phrases as observed nodes, the inference process is a bottom-up maximum likelihood search, as formalized in Equation (1).

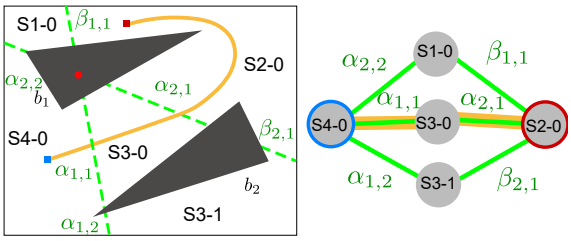
$$\Phi^* = \arg\max_{\phi_{ij}\in\Phi} \prod p(\phi_{ij} \mid \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon). \quad (1)$$

$p(\phi_{ij} \mid \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)$ corresponds to a factor model, as shown in Figure 3, Υ is the world, and the output is a set correspondence values. As illustrated in Figure 3, when a correspondence variable is true the grounding associated with the correspondence variable is said to correspond with a given phrase.

In implementations, each factor model $p(\phi_{ij} \mid \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)$ is approximated by a log-linear model with binary features, which is written as in Equation (2). Tuning the binary feature weight μ changes the approximation of the probability p. The Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [14] algorithm is used in the optimization in the training process.

$$\Phi^* = \arg\max_{\phi_{ij}\in\Phi} \prod \Psi(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon),$$
$$\Psi(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon) = \frac{e^{\sum_l \mu_l f_l(\phi_{ij}, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)}}{\sum_q e^{\sum_l \mu_l f_l(\phi_q, \gamma_{ij}, \lambda_i, \Gamma_{c_{ij}}, \Upsilon)}}. \quad (2)$$

HoDCG grounds spatial relations from a language instruction. We use the spatial relations to obtain a homotopic constraint.



(a) Ref. frames (b) State trans. (ST)

Fig. 6: Map with obstacles.

B. Encoding Path Homotopy

We firstly need a format that encodes the shape of a path, which represents a homotopy class. By the definition of homotopy, we know that paths in the same homotopy class share the same topology that is defined by spatial relations with obstacles. This implies a connection between a homotopy class and a given spatial relation.

We use a decomposition method [13] that divides a map into subregions, which is derived from the Jenkins method as in [10]. The decomposition supports the extraction of topological information, which is obtained by how the path sequentially visits the subregions. Based on the proposed decomposition method, a homotopic DFA (Deterministic Finite Automata) $\boldsymbol{M}^h$ [13] can be constructed that converts a path σ to a string representation v, which is written as $v = \boldsymbol{M}^h(\sigma)$. The string representation is used to identify the homotopic equivalence, which distinguishes topological difference of paths.

In this paper, we present a novel decomposition method, which is modified from the one in [13]. The change we made is moving the center point c into an obstacle so that more generated reference frames are associated with the obstacle. Specifically in our case, the center point locates in one of the obstacles that is associated with inferred spatial relations. This makes the decomposition centered at the obstacle.

The map decomposition method starts with a random sampling process. First, a representative point b_k is randomly sampled from each obstacle B_k. This point may not lie on a line that connects any other two representative points. A center point c is sampled inside one of the obstacles, which must also not lie on a line that connects any two representative points. Restricting the center point in an obstacle reduces ambiguity in homotopy identification. A radial structure is created from the center point c toward all other representative points b_k. Obstacles in the map cut the radial structure into line segments. The end of each line segment terminates within either an obstacle or a map boundary. We use the line segments to identify the homotopy of paths. The line segments are defined as *reference frames* $\boldsymbol{R}$, which separate the map into subregions $\boldsymbol{S}$. Figure 6a gives an example of the map decomposition.

A sequence of reference frames that a path visits reveals the topological information. We assign an *ID character* to each reference frame. The sequence of reference frames

can be represented by a *string* of ID characters. In a path-planning problem, both a start position and a goal position are given, which imply a start subregion and a goal subregion, respectively. In Figure 6a, the blue point is the start position and the maroon point is the goal position. Let each subregion be a state of the DFA, and each reference be an edge of the DFA. We can have a DFA [13] that represents how subregions in Figure 6a are connected in Figure 6b. The DFA in Figure 6b is then modified for homotopic equivalence, which derives a homotopic DFA [13] in Figure 6b.

The homotopic DFA abstracts any path into a string representation. The REPTRIM() algorithm in [13] translates a string for any path into the smallest possible string for any path within that homotopy class. Thus, the homotopy of any two paths can be identified by comparing the strings generated by the homotopic DFA after processed by the REPTRIM() algorithm. Notice that we use a map decomposition method that is slightly different from our prior work in [13]. Because the center point is moved into an obstacle, there is no reference frame that is connected to the center point. It removes the ambiguity brought by reference frames that are connected to the center point, but preserves the validity of other properties. We restate the key property from [13] because it will be important to the results of this paper:

Theorem 1: $\text{REPTrim}(\boldsymbol{M}^h(\sigma_i)) = \text{REPTrim}(\boldsymbol{M}^h(\sigma_j))$ This theorem means that $\text{REPTrim}(\boldsymbol{M}^h())$ encodes a path into a string, and we can use the strings of paths to identify homotopies.

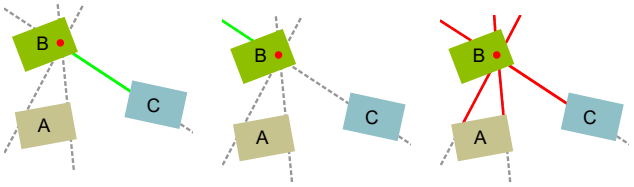
C. Rules from Spatial Relation Functions

As discussed above, we suppose that humans can reason and express intent using spatial language. Thus, natural language does not necessarily describe a path topology. Rather, a human's instruction implies a spatial relation that constrains the allowed path topologies. In Section III-B, a homotopic DFA is used to represent abstract topologies in a string format when a map, a start position and a goal position are known. In this section, we describe how to determine which of the strings generated by a homotopic DFA satisfy a spatial relation constraint specified by a human.

Suppose that a spatial relation constraint is always associated with objects in a map. We define a *spatial relation function* over objects O and returns a corresponding rule ℓ.

A path will never cross multiple reference frames at the same time. Given this condition, we propose a set of operators that assemble ID characters of reference frames into strings, and describe how these character operators correspond to rules on spatial relations.

- **CONCATENATION** $\cap$ is used to concatenate two ID characters. It corresponds to a sequential order of visiting corresponding reference frames.
- **UNION** $\cup$ is used to union two ID characters. It indicates that a path may visiting either of the regions that correspond to the two ID characters of the reference frames.



(a) INBETWEEN(B, C) (b) LEFTOF(B) (c) AVOID(B)

Fig. 7: Spatial Relation Functions.

- **NEGATE** $\neg$ is used to negate one ID character. It indicates that the region associated with an ID character should be avoided.

The following examples of spatial relation functions illustrate how these functions are translated into operators on the IDs:

- $(r_1 \cup \cdots \cup r_n) = \text{INBETWEEN}(o_1, o_2)$,
- $(r_1 \cup \cdots \cup r_n) = \text{LEFTOF}(o) \mid \text{RIGHTOF}(o) \mid \text{TOPOF}(o) \mid \text{BOTTOMOF}(o)$,
- $\neg(r_1 \cup \cdots \cup r_n) = \text{AVOID}(\text{INBETWEEN}(o_1, o_2))$,
- $\neg(r_1 \cup \cdots \cup r_n) = \text{AVOID}(\text{LEFTOF}(o))$,
- $\neg(r_1 \cup \cdots \cup r_n) = \text{AVOID}(o) = \text{AVOID}(\text{LEFTOF}(o) \cup \text{RIGHTOF}(o) \cup \text{TOPOF}(o) \cup \text{BOTTOMOF}(o))$.

A *rule* ℓ describes a logical proposition of reference frames in the disjunctive normal form. For example, a sentence "go between B and C " for the world in Figure 7a is translated to the rule INBETWEEN(B, C), which in turn returns the ID character from visiting the reference frame that connects object B to object C. A sentence "go by the left of B" is associated with LEFTOF(B) returns ID characters that locate at the left of object B, which is shown in Figure 7b. Also, a sentence "avoid B" is associated with AVOID(B), which equals to avoiding four direction near object B. Also by the logic, we can have $\neg(r_1 \cup \cdots \cup r_n) = \neg r_1 \cap \cdots \neg \cap r_n$. It indicates that all the reference frames associated with object C should be avoided.

We can see that a spatial relation function can return multiple associated reference frames. Assembling rules by operators into a new rule supports complicated semantic in a language expression.

- **Sequence** There are often many orders implied in forcing a sequence of ID characters of reference frames. For example, "go to left of A before going to bottom of B". Let ℓ^A be a rule for spatial relation with object A and ℓ^B be a rule for spatial relation with object B. We can have the rule for the command as $\ell^A \cap * \cap \ell^B$.
- **Reverse** It is used to negate a given rule. For example, "never go to left of A". It means all the reference frames that are on the left of object "A" should be avoided, which is written as $\neg\ell^A$.

If we do an exhaustive search on the topology of homotopic DFA, we can have all the possible strings from a start state to a goal state. Each string represents a sequence of reference frames. Rules derived from a spatial relation function can be used to filter out ineligible strings. The set of eligible strings defines a homotopic constraint. Only paths

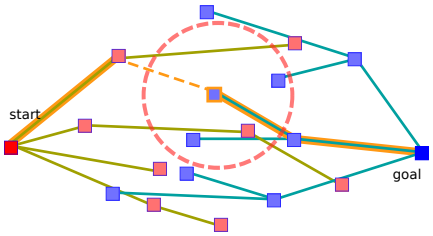
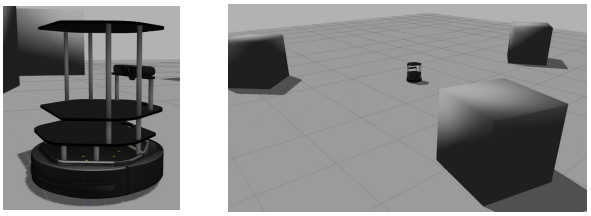
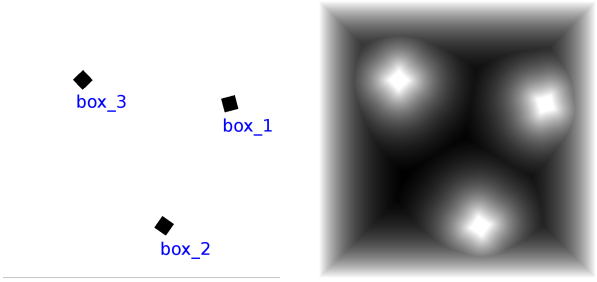


Fig. 8: HARRT*



(a) Turtlebot (b) Gazebo

Fig. 9: Simulation environment.



(a) Map with three boxes (b) Collision avoidance

Fig. 10: Map and costmap of an environment with three boxes

that the homotopic DFA generates strings belonging to the set are considered in planning.

D. Homotopic Path-Planning

We use a homotopic path-planner to find a path by a homotopic constraint. Because of the lack of precision in an instruction, one or more homotopy classes may be included in defining a homotopic constraint. In above sections, we can derive a homotopic constraint from a language expression.

We also want to support adverb in a language instruction [15]. For example, "carefully" and "quickly" indicate different criteria in measuring the performance. This implies a specific objective to consider in planning a path. The objective differs with different verbs and adverbs in language. Thus, we define our problem as a *homotopy-based optimal path-planning problem* [13]. The objective and homotopic constraint are both inferred from a language expression.

We need an algorithm that could explore several homotopy classes in parallel and return the optimal path of each class, because a homotopic constraint could consist of several homotopy classes. HARRT* (Homotopy-Aware RRT*) proposed in [13] is a homotopy-based optimal path-planner, which utilizes the RRT* structure to explore a map according to homotopic information. A homotopic DFA is used to identify the homotopic information of each branch of an RRT* structure. The homotopic information is used to constrain the exploration only in necessary regions. There are two tree that are extended from the start position and the goal position bidirectionally. By the asymptotic optimality of RRT*, both trees converge to optimal structures. To any position, one provides an optimal-to-come subpath, while the other provides an optimal-to-go subpath. Concatenating two subpaths gets a path that is optimal from the start to the goal subject to a constraint of visiting the concatenating position. An example is shown in Figure 8. A node highlighted in orange color in the goal tree. A red dash-line circle indicates the neighboring nodes for adding and rewire process in tree extension. A path in orange color is obtained from this node. At each iteration, one path is found in the start tree, and one path is found in the goal tree. If any new path is better than the current best found path in the hotomopy class it belongs to, the optimal found path will be updated. The optimal paths of multiple homotopy classes can be obtained, because of the variation in selecting concatenating positions. We can thus have the optimal paths of homotopy classes after iterations.

IV. Experiment

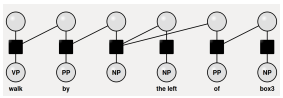
In this section, we use simulations to validate the language-instructed path-planner. In [8], how the goal is inferred from a language instruction is demonstrated. In this paper, we focus on only how a homotopic constraint can be inferred and how it is used for instructing the navigation of a robot. The experiments were running in Gazebo simulator, in which we instructed a Turtlebot moving in among numbered boxes.

Currently the training of HoDCG has not been applied to a big dataset. But it has only been verified with a small dataset with five examples that enumerates all the possible spatial relation functions. The inferred result is consistent with the training dataset. Some examples are shown in Table I. Because HoDCG uses the same training and inference algorithm with DCG and only extends the search space with new types of grounding, we observe that HoDCG has a consistent performance with DCG [8], [16].

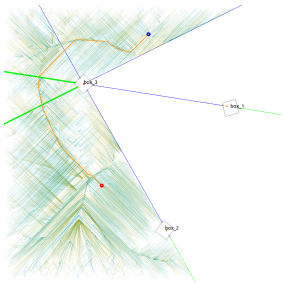
TABLE I: Examples for training

LeftOf(o)	"Swing by the left of ..."
InBetween(o_1, o_2)	"Go between ... and ..."
Avoid(InBetween(o_1, o_2))	"Avoid going between ... and ..."
Avoid(LeftOf(o))	"Stay away from the left of ..."

We obtained a map of the environment with three boxes from the Gazebo, which is shown in Figure 10a. Each box is labeled for reference. We choose maximizing the distance to any obstacle or environment boundary as the objective for planning, and have the costmap of the environment shown in Figure 10b. The darker a position is, the lower cost there is.

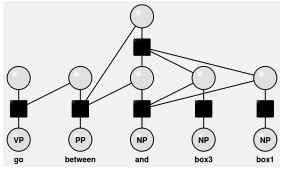


(a) HoDCG

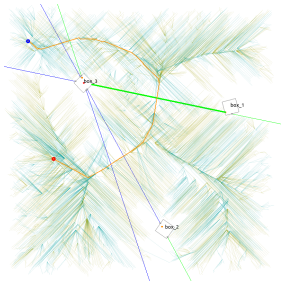


(b) HARRT* and path

Fig. 11: “Walk by the left of box 3”.



(a) HoDCG



(b) HARRT* and path

Fig. 12: “Go between box 3 and box 1”.

Figure 11 gives an example of an instruction “walk by the left of box 3”. Figure 11a illustrates a HoDCG that is created from the instruction. From the inferred spatial relation, the path is required to two eligible reference frames, which are green bold lines in Figure 11b. The red point is the start position, and the blue point is the goal position. From the costmap in Figure 10b, the optimal path is obtained and is shown in orange color.

Similarly, we can have a HoDCG structure generated for “go between box 3 and box 1”, which is shown in Figure 12a. As a result, a reference frame that connects box 3 and box 1 is inferred to be required to visit, which leads to a planned path accordingly in Figure 12b.

We also tested the negation of a spatial relation. Figure 13 shows an example for “Avoid going in between box 3 and box 2”. A negation of in-between spatial relation is inferred from a corresponding HoDCG. A bold red line is shown in each subfigure of Figure 13, which indicates a reference frame that is not allowed to cross. Subfigures show three different solutions in different homotopy classes all satisfy this homotopic constraint. It reveals that there is lack of precision in the instruction. We can either take the optimal of the three as the solution for robot navigation or introduce an interactive process for human to select.

We also tested in a more complicated environment, which is shown in Figure 14a. There are nice boxes in the environment, which means a higher variety in spatial relations and a bigger number of homotopy classes. The corresponding costmap for collision avoidance is given in Figure 14b.

With more possible spatial relations and homotopy classes, the potential imprecision of a language instruction is increased. Figure 15 shows four paths of different homotopy classes all satisfy the inferred in-between spatial relation, which is inferred from an instruction “go between box 5 and box 8”.

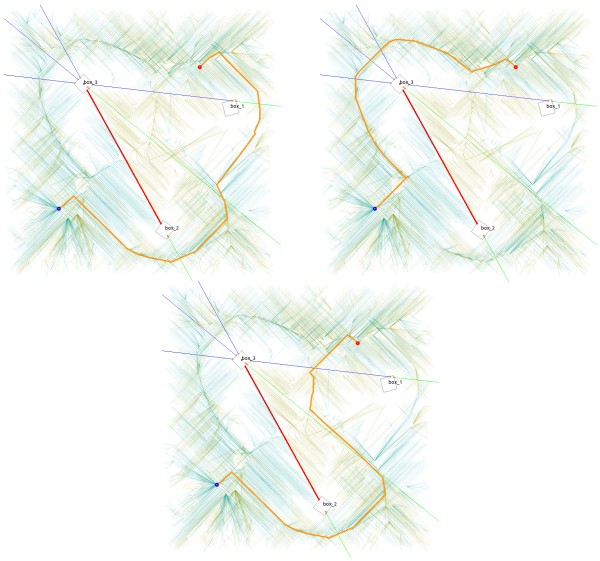
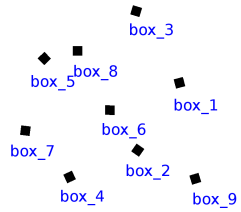


Fig. 13: “Avoid going in between box 3 and box 2”.



(a) Map with nine boxes

(b) Collision avoidance

Fig. 14: Map and costmap of an environment with nine boxes

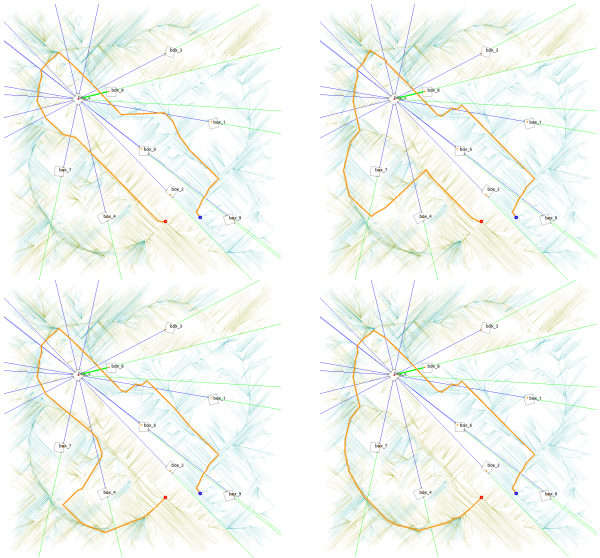


Fig. 15: “Go between box 5 and box 8”.

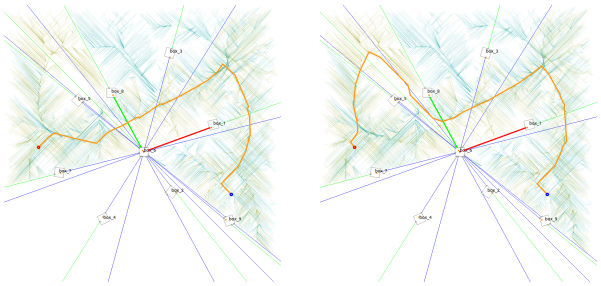


Fig. 16: “Go between box 8 and box 6 and avoid the left of box 1”.

If fewer number of homotopy classes are to be considered, more constraints should be stated in an instruction to reduce ambiguity. Figure 16 shows an example of an instruction “Go between box 8 and box 6 and avoid the left of box 1”. Less paths are obtained because less homotopy classes are eligible for the inferred homotopic constraint.

V. Future Work and Conclusion

The sequence of spatial relations, which supports a sentence like “go ..., then ...” , is not included in the experiment yet. It depends on that the inference of HoDCG could provide sequential information of a set of groundings, which will be added in future work. Enabling the sequence of spatial relation would be another efficient way of reducing ambiguity in a language instruction, which constrains a problem to less number of eligible homotopy classes in a complex environment. We are also going to integrate the framework with other planning information for expanding the features of language instructions, which will be a hierarchical structure [16] that include goal constraints and different objectives. Soft constraints will then be introduced to better support human requirements. For example, in many scenarios, “avoid” only indicates a soft constraint instead of a hard one.

We propose a framework of language-instructed path-planning that integrates a language inference model (HoDCG) and a homotopic path-planning algorithm (HARRT*) so that a robot could read a homotopic requirement in a human’s instruction and planning a required path. The translation from an instruction to a path consists of a mapping from paths to strings, a graph model that interpret an instruction into homotopy classes and a path-planning algorithm that finds optimal paths of the homotopy classes.

References

[1] S. Bhattacharya, “Search-based path planning with homotopy class constraints,” 2010.

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

[3] A. Boularias, F. Duvallet, J. Oh, and A. Stentz, “Grounding spatial relations for outdoor robot navigation,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 1976–1982.

[4] D. J. Barber, T. M. Howard, and M. R. Walter, “A multimodal interface for real-time soldier-robot teaming,” pp. 98 370M–98 370M–12, 2016. [Online]. Available: http://dx.doi.org/10.1117/12.2224401

[5] D. L. Chen and R. J. Mooney, “Learning to interpret natural language navigation instructions from observations,” pp. 859–865, August 2011.

[6] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, “Learning to parse natural language commands to a robot control system,” in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics, J. P. Desai, G. Dudek, O. Khatib, and V. Kumar, Eds. Springer International Publishing, 2013, vol. 88, pp. 403–415.

[7] 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,” 2011.

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

[9] B. Banerjee and B. Chandrasekaran, “A framework of voronoi diagram for planning multiple paths in free space,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 25, no. 4, pp. 457–475, 2013.

[10] E. Hernandez, M. Carreras, and P. Ridao, “A comparison of homotopic path planning algorithms for robotic applications,” *Robotics and Autonomous Systems*, vol. 64, no. 0, pp. 44 – 58, 2015.

[11] J. Hershberger and J. Snoeyink, “Computing minimum length paths of a given homotopy class,” *Computational Geometry*, vol. 4, no. 2, pp. 63 – 97, 1994.

[12] D. Grigoriev and A. Slissenko, “Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane,” in *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, ser. ISSAC ’98. New York, NY, USA: ACM, 1998, pp. 17–24.

[13] D. Yi, M. A. Goodrich, and K. D. Seppi, “Homotopy-aware RRT* : Toward human-robot topological path-planning,” in *Proceedings of the Eleventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI ’16, 2016.

[14] D. C. Liu and J. Nocedal, “On the limited memory bfgs method for large scale optimization,” *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, 1989. [Online]. Available: http://dx.doi.org/10.1007/BF01589116

[15] D. Yi and M. A. Goodrich, “Supporting task-oriented collaboration in human-robot teams using semantic-based path planning,” in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 9084, 2014.

[16] I. Chung, O. Propp, M. R. Walter, and T. M. Howard, “On the performance of hierarchical distributed correspondence graphs for efficient symbol grounding of robot instructions,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 5247–5252.