To appear in IEEE Intl. Conf. on Robotics and Automation (ICRA), May 2019.

Lazy Evaluation of Goal Specifications Guided by Motion Planning

Juan David Hernández, Mark Moll, and Lydia E. Kavraki

***Abstract*— Nowadays robotic systems are expected to share workspaces and collaborate with humans. In such collaborative environments, an important challenge is to ground or establish the correct semantic interpretation of a human request. Once such an interpretation is available, the request must be translated into robot motion commands in order to complete the desired task. It is not unusual that a human request cannot be grounded to a unique interpretation, thus leading to an ambiguous request. A simple example is to ask a robot to "put a cup on the table," when there are multiple cups available. In order to deal with this kind of ambiguous request, we propose a delayed or lazy variable grounding. The focus of this paper is a motion planning algorithm that, given goal regions that represent different valid groundings, lazily finds a feasible path to any one valid grounding. This algorithm includes a reward-penalty strategy, which attempts to prioritize those goal regions that seem more promising to provide a solution. We validate our approach by solving requests with multiple valid alternatives in both simulation and real-world experiments.**

I. Introduction

While industrial robots typically operate in controlled and structured environments, modern robotic systems are expected to conduct autonomous or semi-autonomous tasks in complex and cluttered environments. In these latter challenging scenarios, robotic systems not only use their perception capabilities to understand and safely operate through complex surroundings, but also need to take requests from and collaborate with humans.

In making this collaboration effective, modern robotic systems need to be endowed with different capabilities. In human-robot interaction (HRI), one of the main challenges is to establish the correct semantic interpretation of a human request. This problem is commonly referred to as symbol or variable grounding [1]. Once such an interpretation is available, the human request can be attended by the robotic agent, which converts it into motion commands to complete the desired task. However, there are cases in which it is not possible to establish a unique grounding, thus leading to multi-interpretation requests.

As a first example of such multi-interpretation requests we can consider an automated valet parking (AVP) system. In general, an AVP system consist of a parking lot that has a centralized control infrastructure. This infrastructure is equipped with perception sensors (*e.g.,* optical cameras), and a wireless network to communicate and coordinate with automated (robotic) cars. The infrastructure can use the perception data to create a semantic map, which contains information related to the state (available, occupied, or reserved) of the parking spots [2].

Fig. 1: The manipulator arm of the Fetch Robot is instructed to "put a cup on the table," when multiple cups are available in the scene.

In an AVP infrastructure, a human user can request an automated car to "park." A common approach in AVP systems is to use the semantic map to determine and provide the car the location of one specific parking spot, and then the car will approach and attempt to park in the designated spot [3]. Different heuristics could be employed to determine the specific spot, such as the distance to the car. Nonetheless, the infrastructure cannot determine the difficulty or even the feasibility of parking a specific car in the designated spot. AVP systems could provide multiple available parking spots. In this case, the request "park" cannot be grounded to one specific spot.

Another example of multi-interpretation requests includes manipulation problems in collaborative environments. Let us consider a scenario where a robotic teammate can be asked to fetch an object (see Fig. 1). In these scenarios, robots can use perception information to create semantic maps, which segment, classify and report available objects [4]. Furthermore, perception data can also be utilized to determine the different valid ways to grasp the available objects [5].

In these collaborative scenarios, a human request such as "take that bottle" or "give me that book," contains semantic ambiguities, which do not allow translating the request into specific robot motion commands. In order to clarify such ambiguous requests, a multimodal interface can be used to fuse speech and gestures, thus allowing to ground the specific object [6]. There are more complex cases in which gestures are not enough to clearly ground the implied semantics, *e.g.,* pointing to an object that is nearby other semantically equivalent objects. In such cases, establishing a dialogue

This work has been supported in part by NSF 1317849, NSF 1830549 and Rice University Funds.

J.D. Hernández, M. Moll and L.E. Kavraki are with the Department of Computer Science at Rice University, Houston, TX, USA. {juandhv, mmoll, kavraki}@rice.edu

between the human and the robot can clarify the desired action [7], [8], [9]. In this work, we are interested in solving ambiguous requests, such as "put a cup on the table" when multiple cups are available, without asking a human partner further information.

In both examples, *i.e.,* when asking an automated car to "park," or when requesting a robotic teammate to "put a cup on the table," ambiguity comes from requests that contain multiple interpretations. Although such interpretations can be semantically valid, not all of them might be feasible for the robotic agent. One option to deal with these scenarios could be to exhaustively ground and test each alternative interpretation, until finding a feasible one. Another option could be to ground and test, independently and simultaneously, all the available interpretations, until solving any of them. While the first option would make the human-robot collaboration inefficient, the second alternative would be computationally expensive for a robotic agent, which also has to deal with other concurrent computational processes (*e.g.,* perception, navigation, etc.).

Aiming to cope with this type of multi-interpretation requests, in this paper we introduce a delayed or lazy variable grounding aided by motion planning. In this paper, we assume that interpretations can be mapped to goal regions in configuration space. While this is a hard problem in general, for the specific applications considered in this paper such mapping can be defined. Section II then describes how a multi-interpretation request can then be treated as a start-to-goal-region motion planning problem. The goal regions are implicitly defined in the configuration space (C-SPACE) and are approximated by random samples. Furthermore, in order to cover a wide range of applications, which can include high-dimensional C-SPACES, we solve this start-to-goal-region problem with a new sampling-based motion planner, which is presented in Section III. This new planner attempts to find a solution path from a given start configuration to any of the goal regions; the novelty of this planner is the use of a reward-penalty strategy, which prioritizes those goal regions that are more promising to provide a solution. This strategy also permits determining when more goal samples are required to improve the goal regions' approximation. Finally, in order to evaluate our proposed approach, Section IV presents simulation and real-world results in different scenarios, which range from navigation to manipulation problems.

A specific instance of the problem of multi-interpretation requests also comes up in manipulation in the form of end effector constraints for possible grasps: the goal is not a specific goal configuration, but rather any configuration that satisfies some workspace constraints on the end effector pose. Different planning methodologies have been adapted to deal with this problem [10], [11], [12]. Our work is closest in spirit to [10]. Unlike that work, we restrict the number of goal states considered (or the number of groundings in the context of this paper) as long as the planner can make progress in expanding towards them. This can be seen as putting more emphasis on exploitation at the price of less emphasis on exploration. Also, unlike [10] our algorithm never performs gradient descent over end-effector distance, in part because we do not assume goal regions are restricted to end effector constraints, but also for computational efficiency reasons.

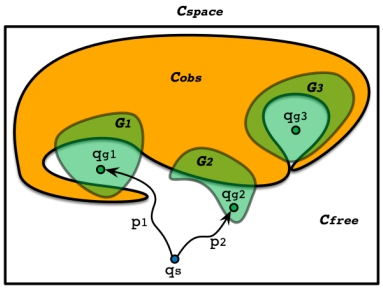


Fig. 2: Start-to-goal-region motion planning problem, which consists in finding a continuous path from a given start configuration q_s, to a goal configuration q_{g_j} that must be contained in any of the provided goal regions G_{1-3}. The goal configuration q_{g_j} must be not only valid (collision-free, as occurs with q_{g_1}, q_{g_2}, and q_{g_3}), but also reachable (which does not occur with q_{g_3}). Possible solution paths to this problem are p_1 and p_2.

II. Problem Setup

A. Definitions & Assumptions

Definition 1: The robotic agent's motion capabilities are described through the set of configurations q, C-SPACE [13]. The C-SPACE, $\mathcal{C}$, is divided into *free space* ($\mathcal{C}_{free}$) and the *obstacle region* ($\mathcal{C}_{obs}$), *i.e.,* $\mathcal{C} = \mathcal{C}_{free} \cup \mathcal{C}_{obs}$. The dimensionality of the C-SPACE is given by the robot's n degrees of freedom (DOF).

Definition 2: A goal region is a subset of the C-SPACE, $GR_j \subset \mathcal{C}$. One or multiple goal regions GR_j can correspond to one specific semantic interpretation I_i. Furthermore, since the goal regions are contained in the C-SPACE, a configuration $q_{g_j} \in GR_j$ can be located either in the *free space* or the *obstacle region*.

Assumptions: For the applications presented in this paper, it is assumed that a semantic entity provides a set of valid interpretations $I = \{I_1, I_2, \ldots, I_k\}$ of a given request. Furthermore, the same semantic entity is also assumed to provide the set of m goal regions $GR = \{GR_1, \ldots, GR_m\}$, which represent the different semantic interpretations. It is important to notice that each semantic interpretation I_i can generate more than one goal region GR_j. Such goal regions are implicitly defined, but, from them, we can sample goal configurations q_g, which have non-zero probability of being sampled.

In this paper, we make these assumptions since we want to focus on the problem that will be formulated in the next section. Furthermore, such semantic entities are generally available. For example, semantic maps are used in AVP systems [2], which provide information about the available parking spots. Another example includes the semantic maps that segment, classify and report available objects in manipulation scenarios [4], together with grasping pose detectors, which can determine the different valid ways to manipulate the available objects [5].

B. The Start-to-goal-region Motion Planning Problem

A basic start-to-goal motion planning problem requires connecting a start configuration q_s to a unique goal configuration q_g. The solution to this problem is a continuous path $p : [0,1] \rightarrow \mathcal{C}_{free}$, such that $p(0) = q_s$ and $p(1) = q_g$. Our objective in this paper is to extend this problem to connect the same start configuration q_s, but now to a goal configuration that is contained in any of the provided goal regions, $q_g = q_{g_j} \in GR_j$ (see Fig. 2). There are different challenges associated with this extended problem.

Although we can assume that a semantic layer will provide the goal regions description, we still need to find the exact goal configuration q_g that has to be both valid and reachable. In this context, we can consider a configuration q as valid if it is collision-free. However, computing the configuration's reachability is as hard as solving the motion planning problem, which is known to be PSPACE-complete [14], [15]. Therefore, verifying both the validity and the reachability of q_g is not a trivial problem, especially when dealing with a complex and high-dimensional C-SPACE. The problem becomes even harder if we have to consider multiple goal regions. Next sections explain how the main concepts of motion planners are used to cope with this extended problem.

C. Tree-based Motion Planner

Given that we want to cover a wide range of applications, which can include high-dimensional C-SPACES, we propose to solve this start-to-goal-region problem with a sampling-based motion planner. Furthermore, since the main objective is to find a solution path to any of the goal regions, and not to all of them, we specifically use a tree-based motion planner. Such planners are rooted at a given start configuration, while the branches can be used to attempt connecting to the different goal regions.

III. SOLVING MULTI-INTERPRETATION REQUESTS

The previous section introduced an extended start-to-goal-region problem. Below we will describe a sampling-based approach to solving this problem in a way that biases the planning towards easy-to-reach goals while still preserving completeness.

A. Initial Goal Region Sampling

We initially approximate the k goal regions by generating a set G of n valid goal samples, so that $G = \{q_{g_i} \in GR \wedge q_{g_i} \in \mathcal{C}_{free}\}$, with $i = \{1,2,\ldots n\}$. This process of goal samples generation can be observed in Fig. 3.

B. Reward-penalty Strategy for Tree-based Planners

In the previous section, we established that there are k goal regions GR_j, which are approximated by a set G of n valid goal samples (q_{g_i}). Since the planning problem consists in finding a feasible path to any goal region, one possible approach to do so is to try a different sample q_{g_i} every time we attempt to connect to a valid goal. Two simple strategies to choose between the goal samples q_{g_i} are: 1) to follow a consecutive order, or 2) to randomly pick q_{g_i}. However, with either approach, we could be trying to connect to difficult or unreachable goal samples.

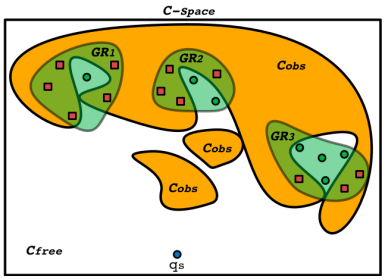


Fig. 3: Goal samples generation from goal regions GR_{1-3}. The valid samples are shown as green dots, while invalid ones (in collision) are represented by red squares.

In Fig. 3, for instance, it can be observed how some goal regions such as GR_2 can be difficult to reach. There are other goal regions like GR_3 that cannot be reached from q_s, although the region contains valid samples. There is another type of goal region that could be easier to access, such as GR_1. At this level of abstraction, the motion planner is the only entity capable of assessing or estimating these intrinsic goal regions' characteristics. It is important to keep in mind that the goal region description does not include information about the goal samples' reachability. Therefore, in order to prioritize those goal regions that seem to be more promising to provide a solution path, we propose to use a reward-penalty strategy.

This reward-penalty strategy consists of identifying whether the attempts to expand towards a goal sample have been successful or not. This information determines if the corresponding goal sample must be rewarded or penalized. In doing so, we can favor and reuse the least penalized goal sample. In this paper, we validate this reward-penalty strategy by using a tree-based planner, which is based on the rapidly-exploring random tree (RRT) algorithm [16]. However, the reward-penalty strategy can be extended to other sampling-based methods such as expansive-spaces tree (EST) [17] and probabilistic roadmap (PRM) [18], or any of their variants (including lazy and asymptotically optimal variants).

In our tree-based planner, we have as inputs the start configuration q_s and the set of goal regions GR (see Algorithm 1). Let us define G_{heap} as a max heap, where all the goal samples q_{g_i} from GR are stored with an initial maximum weight of 1.0. As it occurs with other tree-based planners, our method also biases some of the tree expansions towards the goal (line 5), which, in our case, corresponds to the goal sample with the highest weight in the max heap G_{heap} (line 6). In every tree expansion, *i.e.,* towards a random configuration or a goal sample, the planner expands from the nearest configuration q_{near} in the generated tree (line 11), for a maximum distance ε, thus generating a new configuration q_{new} (line 12).

In our approach, however, we also keep track of whether an expansion towards a goal sample succeeds or fails. In the former case, the goal sample is rewarded by updating its

Algorithm 1: Tree-based planner

Input:
q_s : Start configuration.
GR : set of goal regions.
Output:
$T = (V,E)$: Tree of valid and feasible configurations.
$p : [0,1]$: Solution Path, $p(0) = q_s$ and $p(1) = q_{g_i}$.

```
1  begin
2      V = {q_start}, E = { }
3      G_heap = goalRegionsSampler(GR)
4      while not stopCondition() do
5          if biasToGoal() then
6              q_towards = G_heap.top()
7              goal_biased = True
8          else
9              q_towards = C.genRandomConf()
10             goal_biased = False
11         q_near ← T.findNearNeighbor(q_towards)
12         q_new, success ← calcNewState(q_near, ε)
13         if success then
14             V.addNewNode(q_new)
15             E.addNewEdge(q_near, q_new)
16             if goal_biased then
17                 G_heap.rewardGoalSample(q_towards)
18         else if not success & goal_biased then
19             G_heap.penalizeGoalSample(q_towards)
```

weight w as (line 17):

$$w(q_{g_i}) = w(q_{g_i})/(1.0 - w(q_{g_i})), \text{ if } w(q_{g_i}) < 1.0,$$

while in the latter case, the goal sample's weight is penalized as (line 17):

$$w(q_{g_i}) = w(q_{g_i})/(w(q_{g_i}) + 1.0).$$

Some examples of both situations can be observed in Fig. 4.

C. Goal Region Resampling

It is difficult to determine a correct number of goal samples. In some cases (*e.g.,* when using numerical inverse kinematics solvers), generating a large number of samples can be computationally expensive. In other cases, a small number of goal samples may fail to correctly describe the feasibility and reachability of the goal regions.

In order to avoid fixing the number of goal samples, we propose to start with a small number, while allowing to generate more samples on demand. To do so, we use the reward-penalty strategy to check if any goal sample reaches a minimum weight value. This minimum threshold is used to trigger the generation of additional goal samples. Such new samples are initialized with a maximum weight of 1.0, thus ensuring that they will be attempted before previous samples. Furthermore, the resampling strategy also sets existing samples' weights to 0.5, which guarantees all samples will be tried before generating more samples. This resampling and initialization strategy also seeks to guarantee probabilistic completeness. An example of how generating more samples not only improves the goal regions approximation, but also leads to an easier solution, is shown in Fig. 5.

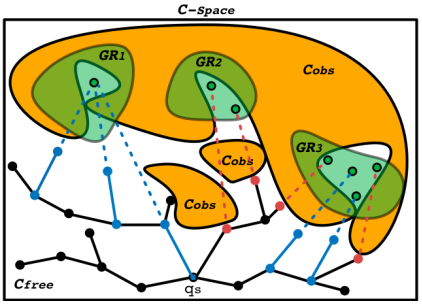


Fig. 4: Tree expansion when solving a start-to-goal-region query. The goal regions and samples are presented in green. Nodes and branches of the tree in blue correspond to successful expansions towards some goal samples. Nodes in red correspond to configuration from which a failed expansion toward a goal sample was attempted. In this example, GR_1 seems to be easier to reach.

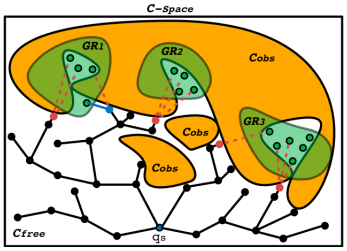


Fig. 5: The tree presented in Fig. 4 continues expanding by using the initial set of goal samples. Additional goal samples are generated. Although some of them are still not reachable, one of them allows to find a solution by connecting to an open area of one goal region.

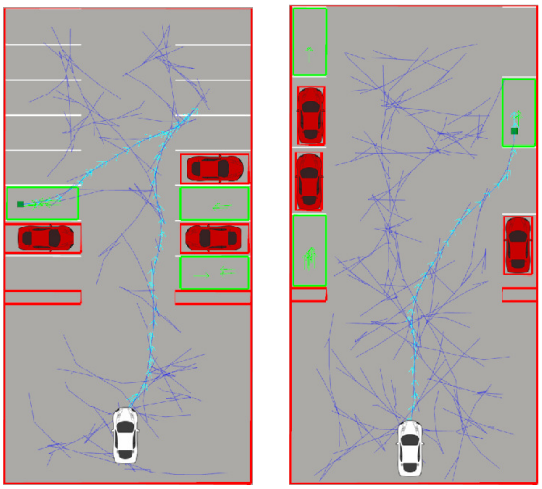
IV. Experiments & Results

This paper has introduced a lazy grounding strategy, which allows us to solve requests that have multiple valid interpretations. This section presents two different test cases: one for automotive applications, and another one for semi-autonomous robotic teammates in collaborative tasks.

A. AVP System with Multiple Available Parking Spots

Let us consider an automated car that requests a parking spot to an AVP system, as it was described in the Introduction. Instead of generating a unique position to park, the AVP system provides multiple available parking spots, each of which is defined by a parking area, thus defining multiple goal regions. The final decision of which parking spot must be used can be taken by an automated car, which uses our proposed lazy grounding approach together with its reward-penalty strategy.

Fig. 6 presents two different parking scenarios. In both cases, a car (in white) is required to park in any of the



(a) Scenario 1 (b) Scenario 2

Fig. 6: Automated valet parking (AVP) system. When an automated car requests a parking spot, the infrastructure can provide the available spots (goal regions in green). Our proposed approach can be used to find a collision-free path to park the car in any of the available spots, while avoiding obstacles (in red). Goal samples inside the goal regions are presented as green arrows (position and orientation), while the solution path and the planner's tree branches are shown in light and dark blue, respectively.

TABLE I: Computation time statistics (in seconds) over 1,000 runs for solving the parking scenarios presented in Fig. 6.

Strategy	**Parking Test Scenarios**			
	Scenario 1		**Scenario 2**	
	mean	std	mean	std
Consecutive (bias 5%)	0.08	0.07	0.29	0.22
Random (bias 5%)	0.07	0.06	0.29	0.24
Reward-Penalty (bias 5%)	**0.03**	**0.03**	**0.17**	**0.20**

available spots (in green), while avoiding collision with the surrounding obstacles (in red). To calculate the solution path (in light blue), we used the tree-based planner given in Algorithm 1. We compared our reward-penalty strategy that was presented in Section III-B with another two possible alternatives for dealing with multiple goal regions. While both alternatives consist in using a fixed number of goal samples when the tree is expanding towards the goal (line 5), one alternative follows a consecutive order, *i.e.,* from the first sample to the last one, the other alternative randomly picks one of the samples each time. The three alternatives, *i.e.,* consecutive order, random, and reward-penalty, have been evaluated for both scenarios presented in Fig. 6. Results of this benchmark are given in Table I, where it can be observed that our proposed reward-penalty approach reduces not only the mean of the computation time, but also the standard deviation.

B. Towards Autonomous Teammates in Collaborative Tasks

Let us now consider that we are in a shared and collaborative environment, in which we ask a robot to "pick an object," in an environment in which there are multiple valid

(a) Scenario 1 (b) Scenario 2

Fig. 7: Test environment for the request "pick up a can." (a) The available cans are marked with green circles. The most easily reachable can is the one on the table (left). (b) An example of a solution to the requested command, when the can on the left is not available. In this case, it is not straightforward to determine the easiest or reachable cans, which can only be determined by the motion planner.

objects. For this set of experiments, we used the manipulator of Fetch [19], which is equipped with a single 7-DOF arm, a parallel-jaw gripper, as well as a base-mounted 3D camera for perception (see Fig. 1). We used Gazebo to simulate the manipulator of Fetch over different working environments [20], while our proposed approach was implemented in MoveIt! [21] by extending the open motion planning library (OMPL) [22]. To evaluate our approach, we define three different scenarios, in which we solve planning problems that involve 8 DOF, 7 of the arm plus the vertical motion of the Fetch's trunk.

The first test scenario includes one table and two shelves, in which there are different objects such as boxes and cans (see Fig. 7). In this environment, the manipulator of Fetch responds to the request to "pick up a can." This query can be solved using our proposed approach, where each of the available cans correspond to a goal region. Furthermore, let us consider that the cans can only be grasped from the top, but using any orientation. This last constraint allows us to define n goal regions, one for each can, which consist of 3-dimensional (3D) poses for the gripper, in which the position is constant with respect to the can, but the orientation around the vertical axis of the can can take any value. In this first scenario, there is one can on the table, which seems to be the easiest to be picked up (see Fig. 7a). We defined a second test scenario that consists in solving the same request, however this time we do not include the can on the table (see Fig. 7b). Notice that in this latter case it is not straightforward to determine the easiest or most reachable can, which can only be estimated by the motion planner.

We defined a third test scenario that has a different setup, which includes three shelves in which there are different objects such as boxes, cans, metallic bars, and a trash bin (see Fig. 8). In this environment, we instructed the manipulator of Fetch to "pick up a box." Each box can be grasped at any vertical position along any of its four sides. These constraints establish $4 \times n$ goal regions, four for each box, which consist of 3D poses for the gripper. In each goal region the orientation is constant with respect to the box, but the position along the vertical axis of the box can take a value within a specified range. Fig. 8b shows one possible solution to this task. Once

(a) Scenario 3 (b) Scenario 3 solved

Fig. 8: Test environment for the request "pick up a box." (a) The shelves contain different objects, such as cans, boxes (marked with green rectangles), metallic bars, and a trash bin. (b) An example of a solution to the requested command. In this case, it is not straightforward to determine the easiest or reachable boxes, which can only be determined by the motion planner.

again, notice that it is not straightforward to determine which box is reachable, which can only be determined by the planner.

In these three scenarios (Figs. 7, 8), we compared our reward-penalty strategy with one alternative that follows a consecutive order, as it was as explained in the previous section, *i.e.,* from the first goal sample to the last one. For these experiments, we first set the goal bias to 5%. Results of this benchmark are given in Table II, where it can be observed that our proposed reward-penalty approach behaves similarly in cases in which there is an easy and accessible option, like grasping the can on the table (see Fig. 7a). However, our reward-penalty strategy shows a considerable improvement when it has to deal with more difficult scenarios, where it is not possible to establish the most reachable goal region.

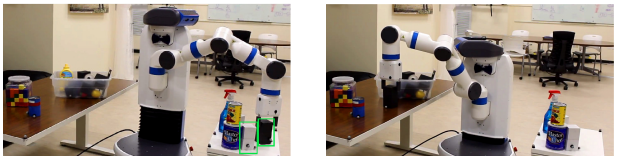
For this set of experiments, where the planner has to deal with a high-dimensional C-SPACE (8 DOF, 7 of the arm plus the vertical motion of the Fetch's trunk), we observed that increasing the goal bias percentage improves the computation times. This means that more tree expansions are attempting to connect to any goal region. However, this increase of the goal bias percentage is only effective if the correct goal sample is being used, for example by using our reward-penalty strategy. Results that show such an improvement are also given in Table II.

TABLE II: Computation times statistics (in seconds) over 100 runs for solving the manipulation problems presented in Figs. 7 and 8 for different values of goal bias percentage.

Strategy	**Bias%**	**Fetch Test Scenarios**					
		Scenario1		**Scenario2**		**Scenario3**	
		mean	std	mean	std	mean	std
Consecutive	5	0.89	0.33	15.59	17.48	64.34	42.62
Reward-Penalty	5	0.97	**0.29**	9.90	7.62	47.18	29.91
Consecutive	50	0.86	0.34	24.40	25.20	55.93	43.03
Reward-Penalty	50	**0.82**	0.34	**6.91**	**4.51**	**25.98**	**15.02**

C. Real-world Tests

We also integrated our proposed approach on the real-world manipulator of Fetch. We defined a scenario that includes a table and a cabinet, over which there are different objects (see Fig. 9), where we tested several requests. One example of those requests corresponds to a task which requires the manipulator of Fetch to "put a box on the table." In this



(a) Picking any box (b) Placing a box on the table

Fig. 9: The manipulator of Fetch is instructed to "put a box on the table." Available boxes are prismatic objects (marked in green).

scenario, we consider a box as a rectangular prismatic object (see Fig. 9a). Each of the available boxes can be grasped from the top, with four different orientation, one for each side. These constraints establish $4 \times n$ goal regions, four for each box, which consist of 3D poses for the gripper. In each goal region the orientation is constant with respect to the box, but the position along the longitudinal axis of the box can take a value within a specified range. Fig. 9a shows one possible solution to this task. Once the box has been grasped, our approach was also used to find a valid and feasible placing location on the table. In this latter case, the entire table surface was define as a goal region, and the planner found an empty space over the table (see Fig. 9b).

V. Discussion and Future Work

In this paper, we introduced a lazy grounding strategy, which allows us to solve requests that have multiple valid interpretations or groundings. We proposed to formulate these multi-interpretation requests as a motion planning problem, in which the alternative groundings are represented with goal regions. Such a formulation permits a single planning problem to consider and evaluate multiple interpretations, without having to exhaustively and separately evaluate each of them. As part of the proposed approach, we also presented a reward-penalty strategy, which seeks to lead the planner to prioritize those goal regions that are more promising to provide a final solution.

We evaluated our lazy grounding approach in two different scenarios. The first scenario presented an application in an AVP system, in which an automated car must decide where to park when provided with multiple available spots. The second scenario was oriented to manipulation tasks, in which a robotic agent was requested to pick up a specific type of object (can or box), when multiple options were available. In both cases, our approach not only solved the request, but its reward-penalty strategy also excelled when it was compared with alternative mechanisms. Furthermore, in the scenario of manipulation tasks, we see the possibility of using our approach to develop multimodal interfaces for commanding robots at a high level, which will be pursued in future work.

Acknowledgment

This work would have not been possible without the help and support of Bryce Willey for setting up the software and hardware of both the Fetch Robot and the Vicon Cameras. The authors would also like to thank Zachary Kingston for help with the benchmarking software infrastructure.

REFERENCES

[1] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, pp. 335–346, jun 1990.

[2] H. Grimmett, M. Buerki, L. Paz, P. Pinies, P. Furgale, I. Posner, and P. Newman, "Integrating metric and semantic maps for vision-only automated parking," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2159–2166, IEEE, may 2015.

[3] H. Banzhaf, D. Nienhuser, S. Knoop, and J. M. Zollner, "The future of parking: A survey on automated valet parking with an outlook on high density parking," in *IEEE Intelligent Vehicles Symposium*, pp. 1827–1834, IEEE, jun 2017.

[4] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," *KI - Künstliche Intelligenz*, vol. 24, pp. 345–348, nov 2010.

[5] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp Pose Detection in Point Clouds," *The International Journal of Robotics Research*, vol. 36, pp. 1455–1473, dec 2017.

[6] B. Burger, I. Ferrané, F. Lerasle, and G. Infantes, "Two-handed gesture recognition and fusion with speech to command a robot," *Autonomous Robots*, vol. 32, pp. 129–147, feb 2012.

[7] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy, "Asking for Help Using Inverse Semantics," in *Robotics: Science and Systems (RSS)*, vol. 2, p. 3, Robotics: Science and Systems Foundation, jul 2014.

[8] D. Whitney, M. Eldon, J. Oberlin, and S. Tellex, "Interpreting multimodal referring expressions in real time," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3331–3338, IEEE, may 2016.

[9] D. Whitney, E. Rosen, J. MacGlashan, L. L. S. Wong, and S. Tellex, "Reducing errors in object-fetching interactions through social feedback," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1006–1013, IEEE, may 2017.

[10] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner, "Manipulation planning with workspace goal regions," in *IEEE Intl. Conf. on Robotics and Automation*, pp. 618–624, May 2009.

[11] A. D. Dragan, N. D. Ratliff, and S. S. Srinivasa, "Manipulation planning with goal sets using constrained trajectory optimization," in *IEEE Intl. Conf. on Robotics and Automation*, pp. 4582–4588, May 2011.

[12] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, "Multi-heuristic A*," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[13] T. Lozano-Pérez, "Spatial Planning: A Configuration Space Approach," *IEEE Transactions on Computers*, vol. C-32, pp. 108–120, feb 1983.

[14] J. H. Reif, "Complexity of the mover's problem and generalizations," in *20th Annual Symposium on Foundations of Computer Science*, pp. 421–427, IEEE, oct 1979.

[15] J. Canny, "Some algebraic and geometric computations in PSPACE," in *Twentieth Annual ACM Symposium on Theory of Computing (STOC)*, (New York, New York, USA), pp. 460–469, ACM Press, 1988.

[16] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *The International Journal of Robotics Research*, vol. 20, pp. 378–400, may 2001.

[17] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," *International Journal of Computational Geometry & Applications*, vol. 09, pp. 495–512, aug 1999.

[18] L. E. Kavraki, P. Švestka, 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.

[19] Melonee Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch & Freight: Standard Platforms for Service Robot Applications," in *Workshop on Autonomous Mobile Service Robots, held at the 2016 International Joint Conference on Artificial Intelligence*, 2016.

[20] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, pp. 2149–2154, IEEE, 2004.

[21] I. A. Sucan and S. Chitta, "MoveIt!." http://moveit.ros.org.

[22] I. A. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, dec 2012.