Closed-Loop Robot Task Planning Based on Referring Expressions

D. Kuhner J. Aldinger F. Burget M. Göbelbecker W. Burgard B. Nebel

***Abstract*— Increasing the accessibility of autonomous robots also for inexperienced users requires user-friendly and high-level control opportunities of robotic systems. While automated planning is able to decompose a complex task into a sequence of steps which reaches an intended goal, it is difficult to formulate such a goal without knowing the internals of the planning system and the exact capabilities of the robot. This becomes even more important in dynamic environments in which manipulable objects are subject to change. In this paper, we present an adaptive control interface which allows users to specify goals based on an internal world model by incrementally building referring expressions to the objects in the world. We consider fetch-and-carry tasks and automatically deduce potential high-level goals from the world model to make them available to the user. Based on its perceptions our system can react to changes in the environment by adapting the goal formulation within the domain-independent planning system.**

I. Introduction

Service robotics is a field which has received rising interest over the past years. However, many systems are limited regarding the different tasks they can execute and thus are typically applied to specific tasks such as cleaning the floor or mowing the grass. With increasing flexibility, robotic service assistants require various components. In addition to techniques for robot motion generation and perception, they need to be able to determine appropriate high-level plans that split the given task into executable sub-tasks. Furthermore, they require a communication layer that is easy to understand, even for non-expert users, to enable the interaction between the user and the task and motion planning (TAMP) algorithms controlling the robot.

In this paper, we introduce a novel framework that combines components from robotics, planning, computational linguistics and user interface design in order to build a system that allows non-expert users to control a complex robotic system. Fig. 1 depicts the central components of our framework: a non-expert user (left) interacts with an intuitive graphical user interface (GUI, middle) to control the robotic system (right). The focus of this work is the development of a dynamic menu-driven user interface, which assists the user in selecting possible goals. It acts as an abstraction layer between natural language and the TAMP algorithms. The interface adapts available menu entries based on the current world state to close the control loop.

High-level task planning provides an intuitive interface for specifying desires and goals to the service assistant. Whereas high-level planner actions such as *drop* or *move* are typically presented in a human readable fashion, objects are usually assigned internal expressions, such as *ID3185* instead of human readable references, e.g., *red cup on the shelf*. The problem of generating *referring expressions* [1] to objects that can be understood by both, human and machine, is studied in the field of natural language generation (NLG) [2]. Recently, Göbelbecker [3] proposed the foundations for generating references to objects in the context of assisting human users in specifying the *goal* of a planning task. Following this approach, human operators can express desires such as *bring me a glass of water* to the task planner without the need of knowing the underlying system. Our communication goal has a simple structure, namely the action to be executed complemented with an object in the world for each parameter of the action. Therefore, we do not rely on a full natural language component and are satisfied with formal language which is comprehensible for non-expert users. We opt for a menu-driven goal formulation interface which only offers applicable actions and feasible objects to the user. Such an interface is even suitable for control by brain signals decoded from an EEG-cap as shown in our previous work [4]. We evaluate the proposed approach in a real world pick and place scenario and substantiate our system in a user study.

Fig. 1: A user (left) interacts with the goal formulation interface (middle) to control a robotic system (right).

II. Related Work

When humans communicate goals to other humans, they identify objects in the world by referring expressions (e.g., *a red cup on the shelf*). The generation of referring expressions has been subject to computational linguistics research for years as one part of natural language generation [5]. With recent advances in natural language processing, computer vision and the rise of neuronal networks, it is nowadays possible to identify objects in images by building referring expressions generated from features [6]. Spatial references can be used to discriminate similar objects [7]. Robotic systems can furthermore not only be controlled by natural language, they can even learn more complex tasks composed of several primitive actions [8]. NLG has been approached with planning techniques [9] where the goal of the planning

All authors are with Albert-Ludwigs-Universität, Institut für Informatik, 79110 Freiburg, Germany. {kuhnerd, aldinger, burgetf, goebelbe, burgard, nebel}@informatik.uni-freiburg.de. This research was supported by the German Research Foundation (DFG, grant number EXC 1086)

problem is the generation of natural language. However, such systems usually lack knowledge about the actions that can be executed and the objects that can be manipulated.

The output of our goal formulation component is a task plan, and we assume that high-level actions (tasks) can be refined to trajectories (motions) of the robot, if the task plan is consistent with the world model in the knowledge base. There are different approaches to solve the TAMP problem, and most of them could be integrated into our framework. Common to most TAMP approaches is a hierarchical decomposition of the problem into task and motion planning layers. Due to the high dimensionality of the TAMP problem the decomposition can be understood as a way to guide the low-level planners based on the high-level plan solution and vice versa. For example, Kaelbling et al. [10, 11] propose an aggressively hierarchical planning method. Such a hierarchical decomposition allows to handle problems with long horizons efficiently. De Silva et al. [12] show an approach based on Hierarchical Task Networks (HTNs) to reason on abstract tasks and combines them with a geometric task planner which works in a discrete space of precomputed grasp, drop and object positions. Recently, Dantam et al. [13] introduce the probabilistically-complete *Iteratively Deepened Task and Motion Planning* (IDTMP) algorithm, which uses a constrained-based task planner to create tentative task plans and sampling-based motion planners for feasibility tests. Srivastava et al. [14] focus on a planner-independent interface layer between task and motion planners. Lozano-Pérez et al. [15] postpone the decision on motion plans to avoid expensive backtracking due to restrictions which might happen, if the low-level planner is queried too early. Instead, they generate a "skeleton" high-level plan and a set of constraints, which need to be satisfied to achieve the goals of the high-level planner. Dornhege et al. [16] integrate task and motion planning by extending the TFD task planner [17] with semantic attachments, i.e., modules which check the feasibility of motion plans on demand to ensure that task plans can be refined to motion plans.

III. High-Level Goal Formulation Planning

Our approach adopts domain-independent planning for high-level control of the robotic system. Whereas automated planning seeks to find a sequence of actions to reach a certain goal, the intended goal of the robotic system is determined by the user. Specifying goals directly requires insight into the internal representation of objects in the knowledge base. In our framework, the knowledge base maintains a world model depending on, e.g., perception input automatically. This obstructs direct user access to the objects. Instead, we allow referring to objects by their *type* and *attributes*. Our automatic goal formulation assistant incrementally creates references to feasible goals in a menu-driven GUI.

A. Domain-Independent Planning

Automated planning is used to transfer a system into a desired goal state by sequentially executing high-level actions. A planning task consists of a planning domain and a problem description. The former describes the object types and predicates. Furthermore, it specifies the preconditions and effects of actions available to manipulate them. The latter models the objects, their initial state and the desired goal. In our experiments, the domain contains a type hierarchy, where for example *furniture* and *robot* are of super-type *base*, and *bottle* and *cup* are of super-type *vessel*. Furthermore, the planning domain specifies relations between objects, e.g., *position* is a relation between objects of type *vessel* and *base*. Finally, the domain also defines the actions, e.g., *grasp* and *move*. The problem description, on the other hand, specifies object instances, such as *cup01* of type *cup* and *shelf02* of type *shelf* as well as relations between them, e.g., the *position* of *cup01* is *shelf02*, as illustrated in Fig. 2.

```
(:objects cup01 cup02 - cup
          shelf01 shelf02 - shelf
          omnirob - robot)
(:init (arm-empty omnirob)
       (at omnirob shelf02)
       (position cup01 shelf02)
       (contains cup01 water))
```

Fig. 2: *Left*: The red cup in the real world, referred to by *cup01*. *Right*: Exemplary PDDL problem description with objects and their initial state.

B. Human and Machine Understandable References

A major challenge when trying to communicate goals to the user is the limited shared vocabulary between the user and the planning system, whose world is described by a PDDL planning task. The planner's most concise representation of the cup in Fig. 2 might be *cup01*, which is not sufficiently clear for the user if there are multiple cups. To solve this problem, the goal generation and selection component uses a set of basic references shared between planner and user. These *shared references* can be combined to create *referring expressions* to objects or sets of objects in the world [1, 3]. Generally, a referring expression ϕ is a logical formula with a single free variable. We say that ϕ *refers* to an object o if $\phi(o)$ is valid in our PDDL domain theory, e.g., $\phi(x) \equiv cup(x) \wedge contains(x, water)$ refers to *cup01*. We restrict ourselves to references that are conjunctions of facts. This is preferable for computational reasons and also allows us to incrementally refine references by adding constraints, e.g., adding $contains(x, water)$ to $cup(x)$ restricts the set of all cups to the set of cups containing water. A reference is *unique* iff it refers to exactly one object. However, it is usually sufficient to create references to sets of objects, e.g., if the user wants a glass of water it might not be necessary to refer to a specific glass as long as it contains water.

To reference objects in planning domains, we need to specify the components that are required to create *shared references*. We distinguish three fundamental reference types. **Individual references** describe objects that can be identified by their name, e.g., the *content* objects *water* or *apple-juice*, and the *omniRob* robot. Additionally, **type-name references** are used to specify objects by their type. They allow referring to unspecific objects as a *shelf* or *cup*. With **relational references** we can refer an object using

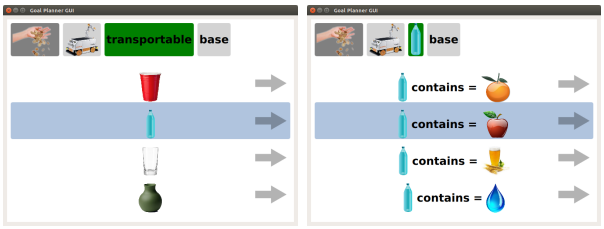

Fig. 3: Graphical user interface of the goal formulation planner: The parameters of a previously selected action are refined step by step to get a final goal.

a predicate in which the object occurs as an argument. In our scenario, most relational references are binary attribute relations whose first parameter is the object that is referred to, and the second parameter is an object in the domain of attribute values. In the example above, a cup can be described using its *content* by the binary relation $contains(x, water)$.

The most natural way for the planner to represent a goal is a conjunction of predicates, e.g., $cup(x) \wedge shelf(y) \wedge position(x, y)$ to put a cup on a shelf. This, however, is a rather unnatural way to refer to goals for humans. We found that it is more natural to use the action that achieves the goal than the goal itself, e.g., $action(put, x, y) \wedge cup(x) \wedge shelf(y)$. Therefore, we include **action references**, a macro reference for all predicates in the action's effect, as additional building blocks to create references to objects in the world and allow the users to specify their goals.

C. Adaptive Graphical Goal Formulation Interface

In our aim for a flexible yet user-friendly control method to set the robot's goals, we use the object references to create a dynamic, menu-driven goal formulation user interface which is depicted in Fig. 3. After the initial selection of a goal type, e.g., *drop*, we have to determine objects for all parameters of the selected goal. We start by populating the goal with the most specific reference that still matches all possible arguments, e.g., *omniRob*, $transportable(x)$ and $base(y)$, assuming that *omniRob* is an individual reference and *transportable* and *base* are type-name references (Fig. 3, left). The current goal reference is displayed in the top row of the GUI. The user interface then provides choices to the user for further refinement of the argument. In our example, the first argument *omniRob* is the only object in the world that fits the parameter type *robot* which is why it does not have to be refined any further. Therefore, we start by offering choices for refining the second argument $transportable(x)$ which yields the selections $cup(x)$, $bottle(x)$, $glass(x)$ and $vase(x)$. This continues until the argument is either unique, it is impossible to further constrain the argument or any remaining option is acceptable for the user. In the example, we refine the first choice $bottle(x)$ based on its *content* (Fig. 3, right) by adding a relation $contains(x, o)$ to the referring expression, where o is an object of type *content*. This procedure is repeated for all parameters of the goal, which will finally result in a single goal or set of goals (if the references are not unique) that are sent to the planner.

Some features cannot be used to partition the remaining objects for one parameter (e.g., not all objects have the attribute *color*), in which case an entry for all *other* objects can be chosen. Additionally, we allow to skip the refinement of the current parameter and use an *arbitrary* object for it. Finally, we provide an entry to go *back* to the previous refinement step.

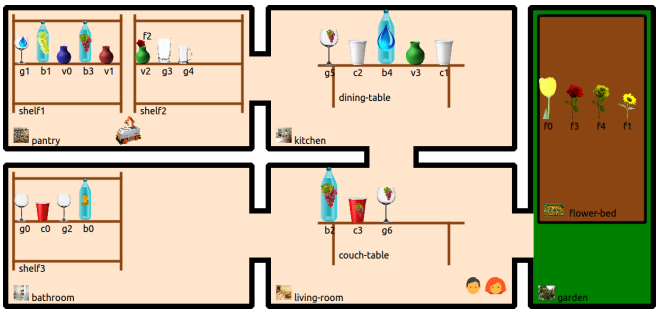

Fig. 4: An exemplary scenario as used in our user experiments with four rooms, a garden, two humans, a robot and multiple objects.

The decision on which feature to use for refining the current selection is based on maximizing the resulting partition's information content, which is similarly computed as in decision tree learning [18]. This strategy prefers to split the remaining objects in a way that reduces the total number of refinement steps. Moreover, the method allows to split the referable objects more equally, thus offering the user a meaningful choice at every step. During the refinement process, we only offer choices that can result in an achievable goal, where goal reachability is efficiently approximated by *delete relaxation* [19]. For example, if all cups were out of reach of the robot, the choice $cup(x)$ would be removed from the selection above. This might result in a completely different selection being preferred, e.g., one that uses the transportable's color or position for distinction. If several objects can satisfy the specified goal, the planner resolves this ambiguity by picking an arbitrary object among them.

IV. Experiments

In the following, we present the evaluation of our planning framework regarding its performance and user compliance on several virtual scenarios of varying complexity and a fetch-and-carry task carried out by a robot in the real world.

A. Implementation

We employ the robotic operating system ROS [20] to connect the central knowledge base, which stores the current world state, to the proposed menu-driven user interface. The back-end of the goal formulation GUI uses Fast Downward [21] to determine a task plan for the selected goal. Changes in the knowledge base automatically trigger updates of the front-end. Furthermore, the user controls the front-end by keyboard or mouse but is not restricted to these [4].

B. Scenario Setup

We created a virtual scenario with five rooms as depicted in Fig. 4: a kitchen with a dining table, a living room with a couch table, a pantry with two shelves, a bathroom with one shelf and a garden containing a flowerbed. Bottles, cups,

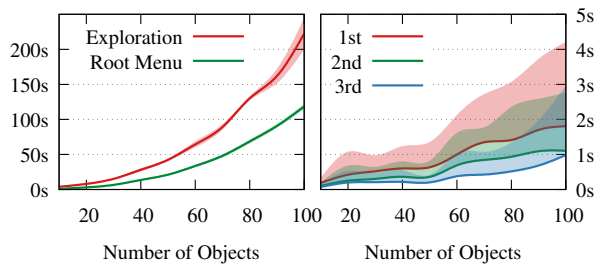

Fig. 5: Evaluation of the computation time for different numbers of objects in the environment averaged over random actions. *Left:* The plot shows the mean and standard deviation of building the menu structure at the beginning and includes initial exploration and root menu creation. *Right:* Refinements of a goal can be done efficiently. It shows the mean and positive standard deviation times of the first three refinements.

glasses and vases are distributed among the furniture. There are three types of flowers (e.g., *rose*), seven drinking contents (e.g., *red-wine*), five colors (e.g., *red*) for cups and vases and three for flowers and finally, four glass shapes (e.g., *balloon*). Flowers can be put into vases but may also be placed directly on furniture. A robot (*omniRob*) has the ability to move between the rooms and serve the two persons (*me* and *friend*). Finally, the available actions are: *arrange* a flower in a vase, *pick* a flower out of a vase, *grasp* a transportable object, *drop* a transportable object on a furniture, *give* a transportable object to a human, *pour* a liquid from one vessel to another, *drink* to assist a human with drinking a drink, *move* the robot between rooms and *approach* a furniture or human for further interaction.

C. Goal Formulation Performance

In our first experiment we evaluated the performance of the goal formulation interface. We used a scenario generator which randomly creates instances of the planning problem with an increasing number of objects. To assess the performance, we measured the time required to start the user interface and select parameters of random actions. The experiment was repeated 100 times and averaged to retrieve reliable results. Fig. 5 illustrates the run times needed for several operations as a function of the number of objects present in the world. The most time-consuming component is given by the initial object exploration, where potentially reachable goals are determined based on relaxed exploration (left, red). Another computationally expensive operation is the root menu generation, where initial partitions are chosen for all actions (left, green). In contrast, the reference refinements for the current parameter of an action requires in average less than 2 s even for scenarios containing numerous objects (right). However, this assertion only holds as long as the world and thus the references do not change. Considering dynamic environments, changes of the world are frequently triggered by actions taken by the robotic service assistant or other robotic and human agents. For example, when the robot has grasped a cup, the system should no longer refer to the cup as *the cup on the table*. Instead, the reference must be rebuilt given the updated environment state yielding *the cup at the robots gripper*. For simplicity, our approach rebuilds all object references when an environment change has been detected. In the future, only obsolete references should be recomputed in order to scale well on larger scenarios.

Finally, as we shown in [4] the system is adaptive to other tasks and environments. To improve practicability of implementing new robotic behavior we could apply learning strategies to our system, e. g., [22].

D. Usability Study

With this preliminary investigation we want to assess the user-friendliness and intuitiveness of the system and how humans use references to objects.

a) Participants: A total of 20 participants (3 female, 17 male, 25 – 45 years) took part in the user study and gave their consent for the anonymized processing of the collected data. The participants were students in computer science and administrative employees of the university. They used our system the first time and were not familiar with it.

b) Data Collection and Measures: The participants had to use our system to accomplish tasks in five simulated scenarios, which were generated beforehand to get comparable results. The five scenarios with increasing complexity were: (S1) Move the robot to the garden, (S2) Drink beer using a beer mug, (S3) Arrange a red flower in a red vase, (S4) Place a red rose on the couch table, and (S5) Give a red wine glass with red wine to your friend. After introducing the user interface by explaining the individual components of the system, the participants had to accomplish the five tasks using the GUI. Since there were no time constraints and sub-optimal strategies were allowed, all users managed to reach the requested goal states. We counted the number of *steps* the participants required to finish the predefined tasks successfully, where a step is either a refinement of an attribute or the selection of the *back* entry in the menu.

For each scenario the participants had to rate if the displayed control opportunities offered by the user interface *comply* to their expectations in a questionnaire, where the *compliance* levels ranged from 1 (unreasonable) to 5 (fully comply). Moreover, we asked the participants to rate the overall *intuitiveness* of the GUI in the range of 1 (not intuitive) to 5 (excellent). We then asked whether the participants prefer to describe objects using references or via internal names (e.g., *v2*). Additionally, we evaluated the *subjective quality* of object references ranging from 1 (not prefer at all) to 5 (highly prefer). We proposed four references to objects depicted in Fig. 4 and let the users rate how well each of those references describes the corresponding object. Moreover, subjects were asked to generate references to these objects in natural language themselves in the way they would tell a friend to find an object. In particular, we considered the green vase with the red rose located in the pantry (*v2*) and the glass, filled with red wine (*g6*), located on the couch table in the living room. The proposed references ranged from under-determined to over-determined descriptions, e.g., *the green vase* vs. *the green vase located in the right shelf in the pantry which contains a red rose*.

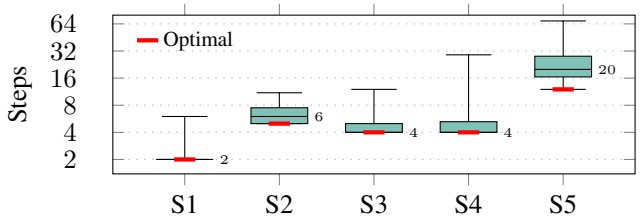

Fig. 6: The box plots illustrate the number of steps required by our participants to achieve a given goal in five different scenarios S1-S5 (optimal number of steps indicated in red, numbers denote the median)

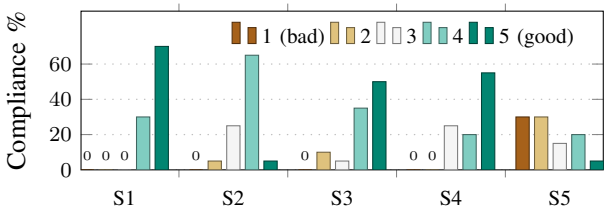

Fig. 7: Compliance of the offered choices with the users' expectation for five tasks in different scenarios. The participants had to select compliance levels from 1 (unreasonable) to 5 (fully comply).

c) Result: Fig. 6 shows the quantitative result of the user study. We counted the number of steps performed by each of the participants to achieve the predefined tasks successfully. The figure shows box plots for each scenario. Additionally, the plot contains the optimal number of steps which are required to successfully achieve the goal.

Most of the participants were able to find a near-optimal strategy to solve the task. The outliers in the first four scenarios are mainly caused by the user exploring the possibilities of the user interface. The increased number of steps in the last scenario can be traced back to the following reasons. First, the scenario required two actions to be able to achieve the task: fill a balloon shaped glass with red wine and give this glass to the friend. Only a few users were able to determine this fact at the beginning. Therefore, the participants had to correct their decisions which results in a higher number of steps in the fifth scenario. Second, the pour action as defined in our scenarios required to specify three parameters: the vessel to pour from, the vessel to pour to and the liquid that is poured. Our system usually refers to the first vessel by its content, so the redundant refinement of the liquid as last parameter is not intuitive to the users. Finally, we split a partition based on its information content to reduce the number of refinements. This strategy can lead to unexpected refinements of object attributes since the user might prefer these in a different order.

Fig. 7 shows the results on how well the choices offered by the high-level planning GUI actually comply with the expectations of the users. A large percentage of them comply with the refinements provided by the GUI in the scenarios S1 to S4. Due to the previously mentioned problems however, S5 has been rated worse. A short training period of the users to get familiar with the interface might help to improve the compliance in S5. Overall, 80% of the participants rated the GUI as intuitive, i.e., according to the aforementioned metric they rated the intuitiveness with at least 3 (acceptable). In particular, 85% of the participants preferred referring to objects by incremental referencing over internal names (e.g., *green vase on the couch table* vs. $v1$).

In the last user experiment, we evaluated the *subjective quality* of object references. According to our results, preferred references highly depend on whether the spatial context of the agents in the world is considered or not. One group of users only preferred references that uniquely identify the objects independent from the location of the agents. This group preferred references such as *the vase containing a rose* or occasionally also *the vase in the right shelf* for $v2$ and *the red wine glass on the couch table* for $v6$. Another group preferred under-determined references as they considered the spatial context of the agents. This group preferred references such as *the green vase* for $v2$ and *the red wine glass* for $v6$. Interestingly, the capability of users to impersonate the acting agent has also a strong influence on the references preferred by the second group. For referring to $v2$, some users of the second group additionally specified the room or the content of the vase, assuming that the assisting agent is also located in the living room and therefore requires a more detailed object description, while they preferred under-specified references for objects on the couch table. Detailed over-specified references were refused by all participants, but more firmly by the second group. Summarizing, our evaluation revealed that incrementally building object references is suitable to describe objects precisely. Room for improvement was identified in updating object references that change during plan execution and in the consideration of temporal and spatial context.

E. Real World Experiments

In order to evaluate our framework in real world experiments, we consider the environment depicted in Fig. 8 (left) which contains two shelves and a table as potential locations for pick and place actions. An *omniRob* omnidirectional mobile manipulator platform by KUKA Robotics is used as an autonomous service assistant that supports the human operator in fetch-and-carry tasks. The robot needs to be able to execute the following actions autonomously: (1) approach a location, (2) grasp an object, and (3) drop an object. For that, we employ sampling-based motion- and manipulation-planning techniques based on rapidly-exploring random trees [23] and probabilistic roadmaps. To be able to react to changes in the environment, as adding or removing objects, we use two statically mounted RGBD cameras. They observe the shelves and report detected changes to the knowledge base using the simtrack object detection framework [24]. In addition, the robot carries an on-board camera which is used to perform collision checks in manipulation planning.

We performed the experiments in a way that unexpected world changes may occur at any time through actions taken by another unknown agent. In practice, this agent could refer to a human taking actions that directly affect the execution of the current high-level plan. Therefore, we initially placed a cup on one of the shelves and queried the goal formulation assistant to generate a sequence of actions leading to the

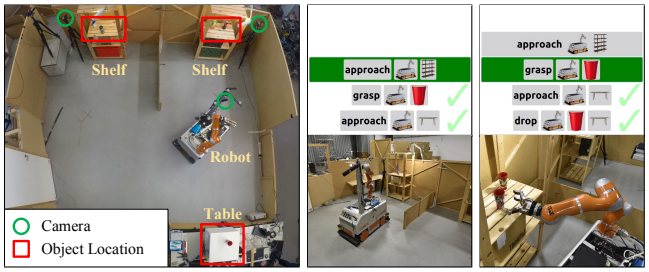

Fig. 8: *Left:* The experimental environment with a service assistant, two shelves and a table. *Right:* Snapshots of the actions *approach* and *grasp*. Top row: user interface, bottom row: real world.

goal state *cup on table*, i.e., $approach(shelf\ with\ cup)$, $grasp(cup)$, $approach(table)$, $drop(cup)$. Once the robot arrived at the corresponding shelf in the execution phase of the plan, a human agent took the cup while the robot was about to grasp it and transferred it to the other shelf. In order to obtain quantitative results on the performance of our framework in such a scenario, we run this experiment 10 times with different initial cup placements and evaluated its ability to generate the goal state in the real world despite the external disturbance introduced by the human agent. For all runs, our perception system correctly updated the information on the cup in the knowledge base, in turn triggering the goal formulation assistant to perform a re-planning step. The updated action sequence always contained two additional actions, namely moving to the shelf where the human agent dropped the cup and grasping the cup. In total, 59 out of 60 (98.33%) scheduled actions were successfully executed and thus 90% of the runs succeeded in generating the goal state despite the disturbance. Only one run failed in the action execution phase due the inability of the low-level motion planning algorithm to generate a solution path for the mobile base within the prescribed planning time. On average, our system required an overall time of 258.7±28.21s SD for achieving the goal state in the real world.

V. Conclusions

In this paper, we presented a novel system unifying high-level task planning, low-level motion planning and perception of the environment with an intuitive graphical user interface to enable non-expert users to formulate feasible goals based on object references to command a robotic service assistant. As shown in the experiments, our referencing system is capable of dealing with environments populated by numerous objects. Moreover, the provided graphical user interface and the navigation therein has been perceived as intuitive and compliant with respect to the users' expectation. This observation is also reflected by the low number of steps required by users in the GUI in order to command complex tasks to the robotic service assistant. Furthermore, we proved that our system is capable of reliably accomplishing the desired goal states in the real world.

References

[1] R. Dale and E. Reiter, "Computational interpretations of the gricean maxims in the generation of referring expressions," *Cognitive science*, vol. 19, no. 2, pp. 233–263, 1995.

[2] E. Reiter and R. Dale, *Building Natural Language Generation Systems*. Cambridge, UK: Cambridge University Press, 2000.

[3] M. Göbelbecker, "Assisting with Goal Formulation for Domain Independent Planning," in *KI 2015: Advances in Artificial Intelligence*. Springer, 2015, pp. 87–99.

[4] F. Burget, L. Fiederer, D. Kuhner, M. Völker, J. Aldinger, R. T. Schirrmeister, C. Do, J. Boedecker, B. Nebel, T. Ball, and W. Burgard, "Acting thoughts: Towards a mobile robotic service assistant for users with limited communication skills," in *Proc. of the IEEE European Conference on Mobile Robotics (ECMR)*, Paris, France, 2017.

[5] E. Krahmer and K. Van Deemter, *Computational generation of referring expressions: A survey.*, 2012.

[6] M. Shridhar and D. Hsu, "Grounding spatio-semantic referring expressions for human-robot interaction," *CoRR*, vol. abs/1707.05720, 2017.

[7] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg, "Modeling context in referring expressions," *CoRR*, 2016.

[8] P. Lindes, A. Mininger, J. R. Kirk, and J. E. Laird, "Grounding Language for Interactive Task Learning," 2017.

[9] A. Koller and R. P. Petrick, "Experiences with planning for natural language generation," *Computational Intelligence*, vol. 27, 2011.

[10] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 1470–1477.

[11] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *I. J. Robotics Res.*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[12] L. De Silva, A. K. Pandey, M. Gharbi, and R. Alami, "Towards combining HTN planning and geometric task planning," *CoRR*, vol. abs/1307.1482, 2013.

[13] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, "Incremental task and motion planning: A constraint-based approach," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, 2016.

[14] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 639–646.

[15] T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2014, pp. 3684–3691.

[16] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic Attachments for Domain-independent Planning Systems," in *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, 2009.

[17] P. Eyerich, R. Mattmüller, and G. Röger, "Using the Context-enhanced Additive Heuristic for Temporal and Numeric Planning," in *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 2009, pp. 130–137.

[18] J. Quinlan, "Induction of decision trees," *Machine Learning 1*, pp. 81–106, 1986.

[19] B. Bonet, G. Loerincs, and H. Geffner, "A Robust and Fast Action Selection Mechanism for Planning," in *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI 1997/ IAAI 1997)*, July 27–31 1997, pp. 714–719.

[20] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[21] M. Helmert, "The Fast Downward Planning System," *Journal of Artificial Intelligence Research 26 (JAIR 2006)*, pp. 191–246, 2006.

[22] T. Welschehold, C. Dornhege, and W. Burgard, "Learning mobile manipulation actions from human demonstrations," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, 2017.

[23] F. Burget, M. Bennewitz, and W. Burgard, "BI^{2}RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea, 2016.

[24] K. Pauwels and D. Kragic, "Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking," in *IROS*, Hamburg, Germany, 2015.