To appear in: IEEE Robotics & Automation Letters, 2020.

Increasing Robot Autonomy via Motion Planning and an Augmented Reality Interface

Juan David Hernández*, Shlok Sobti*, Anthony Sciola, Mark Moll, and Lydia E. Kavraki

***Abstract*— Recently, there has been a growing interest in robotic systems that are able to share workspaces and collaborate with humans. Such collaborative scenarios require efficient mechanisms to communicate human requests to a robot, as well as to transmit robot interpretations and intents to humans. Recent advances in augmented reality (AR) technologies have provided an alternative for such communication. Nonetheless, most of the existing work in human-robot interaction with AR devices is still limited to robot motion programming or teleoperation. In this paper, we present an alternative approach to command and collaborate with robots. Our approach uses an AR interface that allows a user to specify high-level requests to a robot, to preview, approve or modify the computed robot motions. The proposed approach exploits the robot's decision-making capabilities instead of requiring low-level motion specifications provided by the user. The latter is achieved by using a motion planner that can deal with high-level goals corresponding to regions in the robot configuration space. We present a proof of concept to validate our approach in different test scenarios, and we present a discussion of its applicability in collaborative environments.**

I. Introduction

Commanding a robot to perform a certain task can be time consuming. For many industrial robots it is common that operators explicitly need to define waypoint configurations for a desired trajectory. Since in many industrial settings the same task needs to be executed the exact same way many times, the time spent on programming the robot can be amortized. However, increasingly robots and humans operate in the same space (which may not be a fully structured environment), and each task is executed only a few times. As a result, there has been significant interest in exploring different modes of human-robot cooperation. For low-level specification of a desired trajectory physical interaction can be used (see, e.g., [1]), whereas for high-level task specification speech and gestures have been explored (see, e.g., [2]).

Early on, the potential of virtual, augmented, and mixed reality (VR, AR, and MR, respectively) as other modalities for human-robot cooperation was recognized [3], [4]. The focus in this paper is on AR, but many of the same ideas could also be applied in a VR/MR setting. AR has been shown to be useful in allowing people to command robots and to visualize a robot's planned actions [5], [4]. In contrast to VR, AR uses the real world as a context for understanding and interacting with virtual objects. AR allows a user to seamlessly integrate coordination of a robot's actions with her own world interactions with the world. This enables a more direct interaction with the world without having to use a symbolic representation (such as natural language or code).

Fig. 1: A user placing virtual cans in desired locations. Inset figure shows the user's point of view. The robot takes this input and computes a plan to place real cans in those locations.

Most of the work on the use of AR in robotics has focused on robot motion programming [5], [6] and teleoperation [7], [8]. In this paper, we take a robot-agnostic and object-centric approach. By this we mean that the AR interface emphasizes the desired *outcomes* of robot actions rather than the specific motions of a robot needed to achieve these outcomes. When commanding highly dexterous robots it is often difficult for a human collaborator to reason about their reachable workspace, or kinematic/dynamic feasibility of a task. In our approach, the user can focus on specifying a sequence of high-level goals and let the robot itself determine a sequence of feasible motions that achieve these goals. This latter not only endows the robot with capability to cope with a wider range of scenarios, but it also increases the robot's autonomy. We call this use of AR *High-level Augmented Reality Specifications (*HARS*)*, to distinguish this work both from other uses of AR and other work on commanding robots via high-level specifications.

An example that encapsulates the capabilities of a HARS is shown in Fig. 1, where a user places two virtual cans on a real table. By placing these virtual cans, a user is requesting a robot to get real cans and to place them as indicated by the virtual counterparts. However, there may be different valid alternative selections that will accomplish the intended task. For example, there could be more available real cans than the ones required by the provided HARS. Also, the user, through the proposed AR interface, can provide a range of possibilities of where to place the objects. We propose to exploit the robot's decision-making capabilities of its motion planner to make a final decision from the given alternatives.

*The authors contributed equally to this paper.

This work has been supported in part by NSF 1830549, NSF 1514372 and Rice University Funds.

J.D. Hernández, S. Sobti, M. Moll and L.E. Kavraki are with the Department of Computer Science, and A. Sciola is with Department of Physics and Astronomy at Rice University, Houston, TX, USA. {juandhv, sobti, ams16, mmoll, kavraki}@rice.edu

The robot then computes motion plans to put each of the real cans, shows a virtual execution of the plan, and—after approval by the user—executes it in the real world.

The contributions of this paper are a new high-level AR interface, which focuses the user on *what* it wants from a robot rather than the *how*. The approach relies heavily on motion planning techniques, which use high-level goals that do not correspond to a single state, but, rather, implicitly describe regions in configuration space that achieve a desired task. The work presented is a prototype interface, which combines a head-mounted AR device with a motion capture system and a mobile manipulator to solve a certain type of HARS. We introduce the notion of a virtual inventory of *types* of objects that a user can use to specify object arrangements. We demonstrate the application of HARS in pick and place operations; the scenarios that can be considered are as follows. The user places a virtual object in some region of the workspace (via the augmented reality (AR) interface), after which our motion planner takes over. Each HARS request of such a type requires four robot actions that the planner needs to solve: (1) to move close to the object(s) that are to be manipulated; (2) to pick up the specified object; (3) to move close to a specified placement location; and (4) to place the real world object according to the user specified parameters. Examples of such pick-and-place tasks include complex block stacking arrangements and assembly operations.

The rest of the paper is organized as follows. The next section provides an overview of related work. Section III describes in detail how a HARS can be decomposed into motion planning problems. Section IV describes the complete AR-robot system. Section V describes results obtained with our prototype system on some motivating examples. Finally, in Section VI we discuss the results so far and outline directions for future work.

II. Related Work

Effective collaboration between humans and robots relies on two-way communication, which in the past has been explored via multimodal interfaces (e.g., [9], [10], [11], [12]). Recent advances in AR technologies have allowed for an alternative mode of communication [3]. We focus on the AR work close to the concept of HARS.

AR research in robotics can be primarily classified as belonging to either visual feedback and/or robotic instruction. Visual feedback can used to graphically represent task-plans [4], disambiguate user and robot references [13], or even effectively reduce the skill-barrier to efficient operation [14]. Robotic instruction via AR has mostly been directed towards low-level motion specification. Such frameworks might allow the user to explicitly specify via-points, preview the planned path prior to execution and consequently modify the via-points to address any discrepancies [15], [5], [16], [17], [18]. These types of interfaces, along with visual feedback, have been validated in industrial settings to enhance operator experience particularly in 3-D printing and fabrication [19], [20], [6]. Variations in low-level motion specification might instead rely on the manual definition of collision-free volumes for the generation/optimization of valid trajectories [21], [15], [22], or teleoperation by the user [7], [8], [23]. Low-level motion specification frameworks greatly reduce the hurdles to robot programming; however, they rely heavily on user actions. We instead propose a prototype which further shifts the work load from the user to the robot using AR coupled with motion planning techniques.

The work in [24] introduces the idea of an object-based programming scheme using AR. A context-based multimodal framework that allows the user to interact with the robot is presented based on which the user is able to convey objects of interest and drop locations. Visual feedback including planned paths, selected objects and planned gripper actions are displayed on a 2D monitor for reference. These high-level queries are completed by stitching together preprogrammed low-level actions. The ideas developed, particularly those pertaining to the use of AR and automating low-level actions are very relevant and we build upon them. The interface used in the paper restricts the user to a 2D perspective of a limited virtual environment. Interacting with a 3D world via a 2D interface is limiting and can lead to loss in fluency. As described in Section I HARS uses motion planning techniques and an AR headset to enable an environment/robot agnostic implementation that is physically intuitive.

Another relevant idea is explored in [25] in the context of industrial assembly programming. The paper presents a framework in which the user demonstrates the assembly with virtual objects while the start and end states are recorded. The robot controller then executes a *predefined* pick-place sequence. The paper acknowledges the limitations of such a predefined planning methodology which inhibits the implementation from working in more complex environments. HARS uses ideas such as an always accessible virtual inventory and motion planning to maintain generalization.

The work in [26] explores intention projection to facilitate effective human-robot collaboration by proposing a task-planning paradigm that trades-off cost with the ability to project robot intentions. Similar ideas have been explored by [27]. We acknowledge that intention projection can be very useful in task-planning applications so that the user can plan his actions accordingly. Our paper explores a different yet complementary aspect—that of conveying high-level requests via AR—and deriving geometric constraints from the user-defined high-level goals (e.g., object type, acceptable place regions). A combination of [26] for task-planning and projections [27], and our work for handling individual requests could be considered as a further extension of work presented in this paper.

III. Motion Planning for Solving High-level Augmented Reality Specifications (HARS)

In this paper, we propose a novel approach for collaborating with robots, which allows a user to command a robot by giving a HARS, while letting the robot determine the specific details on how to accomplish the desired task. In order to do so, a user, through an AR interface, manipulates virtual objects that are used to specify the desired state of their

real counterparts. The user can also provide position and orientation ranges around the specific pose that is defined by the virtual object. Such ranges may induce a large number of valid robot motions, which cannot be expressed by a set of preprogrammed robot behaviors. To endow a robot with the capabilities to deal with a HARS in an automated way, we propose to integrate the aforementioned AR interface with a motion planner. Such a motion planner deals with high-level goals that represent regions in the workspace, and hence in the robot configuration space.

A. Translating User Requests into Goal Regions

A HARS for pick-and-place tasks requires four consecutive robot actions (which in the case of a mobile manipulator alternate between moving the robot's base and moving the robot's arm): to get close to some object(s) of the desired type, to grasp one of them, to get close to a placement location, and to place the object as specified by the user. Each of these actions can be translated into a robot motion query. However, such motion queries do not necessarily have a unique robot goal configuration, instead they may have an admissible robot goal region. Therefore, a goal region represents all the valid alternatives that a robot has for completing a desired action.

In this work, the aforementioned robot actions are specified through workspace constraints, which implicitly generate goal regions in the configuration space. Hence, workspace constraints are associated with the specific robot action in consideration. For example, when the robot needs to get close for picking up or placing an object, a workspace constraint is specified by the distance at which the robot is considered to be close enough to the placement location or to the object. For example, in the case of a mobile manipulator, such a distance can correspond to the radius of a sphere of reachable end effector poses. Then, the workspace constraint is "translated" into a navigation goal region, which contains all the valid robot base poses (positions and orientations) at which the robot will be within the reachable sphere.

Another example of robot goal regions can be when the robot has to pick up an object—assuming the robot has already used a navigation goal region to get close enough to the object, a set of workspace constraints contains the different valid ways to pick up the object, e.g., the set of different end effector poses to grasp the object. This information can be available through different entities, such as pose detectors that can predict different ways to manipulate available objects [28]. Then, this set of workspace constraints is "translated" into a grasping goal region, which contains all the valid robot configurations that take the robot (arm and end effector) to grasp the object. The goal regions concept can also be extended to situations in which a user request has multiple valid options, e.g., when the robot needs to pick up one object of a given type and there are several objects of that type available. In this latter case, and also assuming the robot has already moved close enough to the available objects, multiple goal regions that contain the robot's alternatives to pick up one of the objects are obtained from the sets of different end effector poses to grasp each of the available objects.

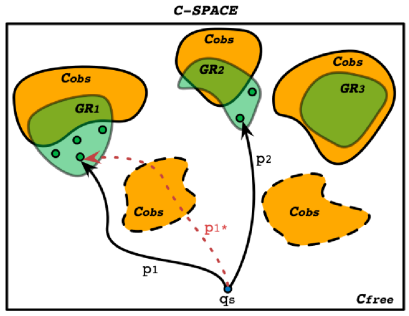

Fig. 2: Start-to-goal-region motion planning problem consists in finding a continuous path from a given start configuration q_s, to any goal configuration q_{g_j} (green circles) that must be contained in any of the provided goal regions GR_{1-3}. The goal configuration q_{g_j} must be not only valid, but also reachable. Possible solution paths to this problem are p_1, and p_2. In other cases such as p_{1*} the path appears to be valid, but it actually collides with initially unknown obstacles.

Once the robot has grasped a real object and has moved close enough to the placement location, in our type of HARS, the user explicitly defines, through the AR interface (examples are given in Figs. 1, 4), a set of workspace constraints with respect to a virtual object, which is used to indicate the preferences for placing a real object. Such constraints can be translated into a placement goal region, which contains all the valid robot configurations that take the robot (arm and end effector) to place the object as specified by the user.

Each of the four robot actions that we need for our type of HARS can be formulated as the solution of a start-to-goal-region motion query. Solving such a motion query consists in finding a continuous path to any of the goal regions. However, we typically do not have an analytic description of the goal regions. We can approximate the goal regions by sampling goal configurations (using, e.g., rejection sampling or an inverse kinematics solver). We can then construct a coarse approximation of the goal regions by generating a set of valid goal samples. As an example, imagine a grasping goal region that contains all the end effector poses to grasp an object from either the top or the side. A valid goal sample corresponds to one specific pose in which the robot, including its end effector, does not collide with any nearby obstacle. Such an approach allows us to reformulate the start-to-goal-region motion query to find a continuous path to any of the goal samples, which approximate a goal region (see Fig. 2).

In order to solve such start-to-goal-region motion queries, we propose to use a tree sampling-based motion planner, since this kind of planner is rooted at a given start configuration, while the branches can be used to attempt connecting to any of the goal samples. Such a behavior is consistent with the formulated planning problem. But more specifically, we propose to use a planner that keeps track of its progress in expanding toward each of the goal samples. If a tree expansion toward a goal sample succeeds, the priority of that state is increased. If it fails, its priority is decreased. Whenever the tree planner is biasing towards a goal region it picks the goal sample with the highest priority, thus prioritizing those goal samples that are more likely to get connected to the tree.

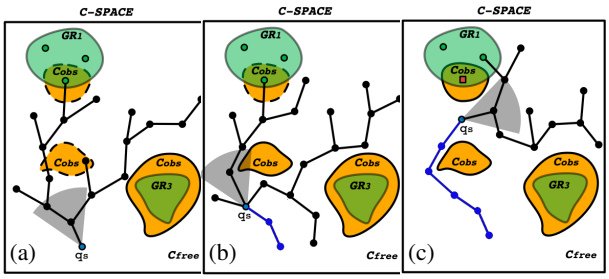

Fig. 3: Online replanning in a partially known environment. (a) The planner must find a solution path to any of the goal regions (in green), while avoiding the obstacles (in orange). Some of the obstacles are initially unknown (dashed line). (b) As the robot follows the path, onboard sensors can detect initially unreported obstacles (sensor's field of view shown in grey), thus discarding part of the previous solution path. (c) Once the robot approaches the goal destination, it discovers that the goal region is partially occluded, thus requiring to find a solution path to an alternative goal sample. The traveled path is shown in blue.

Such a planner was originally presented in our previous work, and we refer the reader to [29] for more detailed information.

B. Online Replanning in Partially Known Environments

In our previous work [29], we assumed the robot has a complete model of the world. However, missing information (e.g., due to occlusion) or changing information (obstacles may change location) require constant replanning. Hence in this paper, we extend our start-to-goal-region planning approach of [29] by using a strategy, which enables fast replanning with minimal computational overhead [30]. At the beginning of each planning cycle, the planner starts from the remainder of the plan of the previous cycle that has not been executed yet. That remainder is rechecked for validity and the valid parts are added to the search tree that the planner maintains. This tree is then grown until a new valid plan is found (which may be instantaneous if nothing the robot's world model has changed).

This strategy encourages the robot to keep using a given goal sample as new information about the environment is sensed. If previously unseen obstacles make the considered goal sample unfeasible (or just very difficult), the planner can automatically discover an alternative valid goal sample *without the need for a higher-level reasoning strategy.* This last part is important, because it avoids having the robot make arbitrary decisions about how much time needs to be spent at different levels of abstraction. Let us consider, for example, that a user request requires a robot to bring an object that is on a table. Even if the robot knows the location of both the object and the table, an initially unknown object that is in the middle of the straight path to the table, e.g., a chair, would require the robot to find an alternative path to get close to the the object. The behavior of this online replanning approach is presented in Fig. 3.

IV. System Overview

The collaboration approach presented in the previous section can implemented in different ways. This section presents one particular implementation, which was used for validation purposes. The following sections will discuss some aspects of such a configuration.

A. Hardware Configuration

In our setup, a Fetch mobile manipulator [31] can move around, while it is tracked by a set of Vicon cameras [32]. Some static elements in the surroundings that are used in manipulation requests, such as manipulation objects (e.g., cans, blocks and cubes) and placement locations (e.g., tables and cabinets), are also tracked and reported to the Fetch. Other elements in the scene can be detected by the robot's onboard perception sensors. A human user who interacts with the robot is equipped with a Magic Leap [33], which is a head-mounted AR device that also includes a portable computer and a remote control. The Fetch, the Magic Leap and the Vicon tracking system are all connected through a WiFi network.

B. HARS Specification and Preview via AR

As it was explained before, in this paper our approach deals with a specific case of HARS, in which virtual objects are used to specify the desired state of their real counterparts. In our AR interface, a user can fetch a virtual object from an inventory (see Fig. 4a), and simply manipulate and place the virtual object in the desired location. Furthermore, our AR interface allows the user to define an admissible placement region where the object can be placed. Such a region is visualized as a cuboid that is built according to the ranges in the x, y and z axes, which are modified by interacting with the Magic Leap's remote control (see Fig. 4b). The user can also indicate orientation preferences, which is done by selecting if a rotation axis is fixed according to the virtual object's pose, or if it can take any value from $[0, 2\pi)$ (see Fig. 4b). All this functionality is intuitive and supported by the Magic Leap. With our proposed work, all user preferences are captured through the notion of goal regions.

Furthermore, the proposed AR interface allows the user to manipulate multiple objects. This is done by sequencing the solution of multiple HARS (see Fig. 4c). In this case, our approach proposes to keep track of the order in which each virtual object was placed by the user, so that the robot can follow the same order. Finally, the AR interface also shows the user a preview of the expected robot motion before its execution. Such a motion preview is superimposed on the real robot, thus giving the user the opportunity of approving or requesting an alternative solution (see Fig. 4d).

C. Coordinate Systems Consistency

To be able to convey information, such as goal queries and robot pose, back and forth between the AR device and the robot they need to share a common frame of reference. This is particularly crucial since the Magic Leap, running Unity [34], assigns itself an arbitrary frame of reference upon boot, while the reference frame of the robot (Fetch) depends on the localization system used (Vicon, in our setup). For this reason the user needs to step through a calibration step each session to align the virtual and real worlds. This requires the

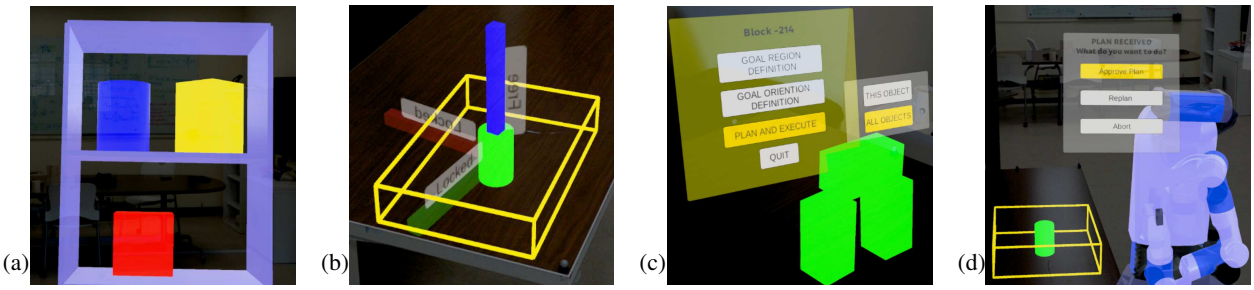

Fig. 4: Proposed AR interface. (a) Inventory of virtual objects. (b) A virtual cylinder is placed over a real table in order to command a robot to place a real wooden cylinder. The preferences in position are given by a cuboid, whose size can be modified through the Magic Leap's controller. The orientation preferences are indicated in the rotation axes. (c) A sequence of HARS can be defined by placing multiple virtual objects, thus commanding the robot to arrange multiple objects. (d) A virtual robot is superimposed on the real robot in order to visualize the computed motions.

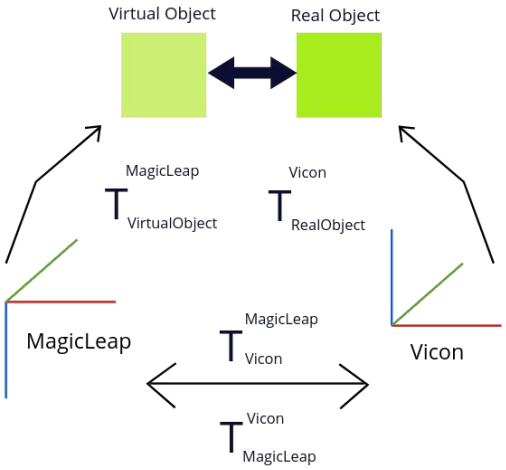

Fig. 5: Calibration to align the virtual Magic Leap (ML) world and real world. This is done via a real object (RO) and a virtual counterpart (VO).

user to manipulate a virtual object and overlay it on its real Vicon tracked counterpart. Once satisfied with the overlay, the user can request to calculate and save the transform that maps the difference in frames. Onwards, all information pertaining to poses is pumped through this transform or its inverse. Figure 5 explains this calibration setup. We are interested in computing T_{VICON}^{ML} and we can assume knowledge of T_{VO}^{ML} and T_{RO}^{VICON}. Once the real object is overlaid with the virtual object, we can perform basic matrix operations to calculate T_{VICON}^{ML} according to the equation, $T_{VICON}^{ML} = T_{VO}^{ML}(T_{RO}^{VICON})^{-1}$.

V. Test Scenarios

This section presents different test scenarios, which seek to demonstrate the capabilities of our approach in cases that are relevant for common human–robot collaboration tasks.

A. Single-object Request

Let us consider a collaborative scenario, which includes manipulation objects such as wooden cylinders and wooden blocks initially placed on a shelf. In this first test scenario, a HARS allows a user to command a robot to fetch an object from its initial position on the shelf (see Fig. 6), then and place the object in a desired location, which in this particular example corresponds to a table (see Fig. 7). In order to do so, the user, through the AR interface explained in Section IV, retrieves a virtual object (a cylinder or a block) and places it over the table, while also defining an admissible placement goal region (see Fig. 4b). Such interaction with the virtual object allows the user to provide the robot with a high-level specification for placing a real-world object, without providing low-level motion details on how to accomplish this task.

In order to solve this HARS, the robot first navigates close to were the wooden cylinders are, then the robot grasps one from the shelf. The location to navigate to is chosen heuristically to maximize the odds of being able to grasp several objects of the same type.

After getting close to the objects, the Fetch uses its sensors to inspect the available objects and their surroundings, so that the robot can detect any unreported obstacle (see Fig. 6c). The robot creates a grasping goal region that includes the gripper poses for grasping each object. In some cases, some of the gripper poses to grasp the available objects can be difficult to reach due to other nearby objects, which can create occlusions and narrow passages. Nonetheless, the approach presented in Section III provides a mechanism to prioritize those objects that are easier to reach. It should be noted that our framework allows maximum flexibility to the user, and rids the user from the burden to specify exactly which object will be picked up.However, if the user wants a specific object, this is also allowed by our approach (e.g., the user can specify and require an object-type which includes only the object of interest).

Once the Fetch has picked up an object (see Figs. 6d), another motion query is required to move close enough to the table. Such a motion query is similar to the initial one to approach the objects on the shelf, and since the robot must deal with a partially known environment. After getting close to the table, the robot needs to inspect the table while looking for unreported objects (see Figs. 7a, 7b). Then, the manipulator of the Fetch robot must place the object on any available spot in its placement goal region. For this last motion query, the placement goal region corresponds to the position and orientation constraints specified by the user via the AR interface (see Fig. 6a). Fig. 7c depicts the Fetch after

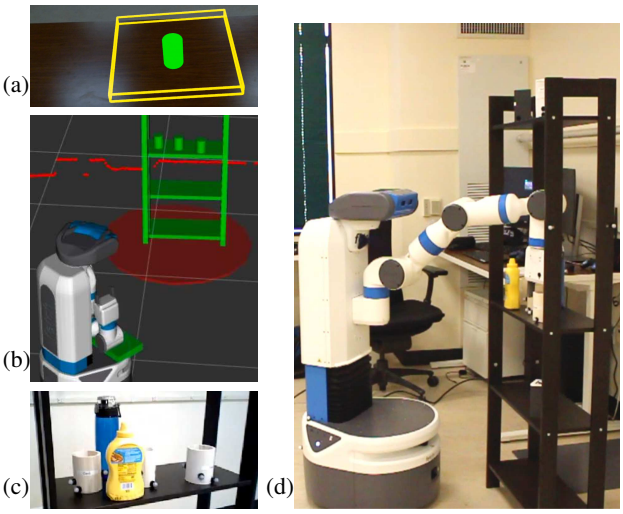

Fig. 6: (a) A user specifies a HARS by placing a virtual cylinder and defining a placement goal region. (b) A navigation goal region is established to approach the shelf where the real cylinders are. The shelf and cylinders are tracked and reported by the Vicon cameras. (c) The Fetch inspects the available cylinders in order to detect other unreported obstacles. (d) The Fetch grasps one of the cylinders in order to continue solving the HARS as shown in Fig 7.

solving and executing this latter start-to-goal-region motion query to place a wooden cylinder on the table.

In summary, upon receiving the initial high-level specification, the Fetch generates each of the described motion queries without additional input from the user. Once the Fetch finds a valid motion plan for a given query, an animated preview of that motion is displayed to the user. The user then has the option to approve the current motion plan, tell the Fetch to recalculate the motion in order to find an alternative solution, or to abort the current action entirely. While this level of interactivity can take place for each individual motion query, the option also exists to allow the Fetch to complete the entire high-level task without interruption. This gives the user the ability to provide various levels of supervision over the Fetch's completion of the high-level task, which is especially useful when the Fetch is given a sequence of HARS to perform, as described below.

B. Multi-object Requests

Let us now consider a second scenario, which also includes wooden cylinders and wooden blocks that are initially placed over a cabinet. This time, however, the user wants to command the robot to arrange a set of objects (e.g., to stack them). This particular task can be specified by sequencing multiple HARS. Similar as in our previous test scenario, the user retrieves and places virtual objects through the AR interface, which also keeps track of the specific order in which the virtual objects are manipulated and placed. Although in this case the user decides the placement order for each object, the HARS provides no constraints on how to get and place the real objects.

To attend this user request, the robot follows the strategy explained in the previous example, i.e., the robot approaches

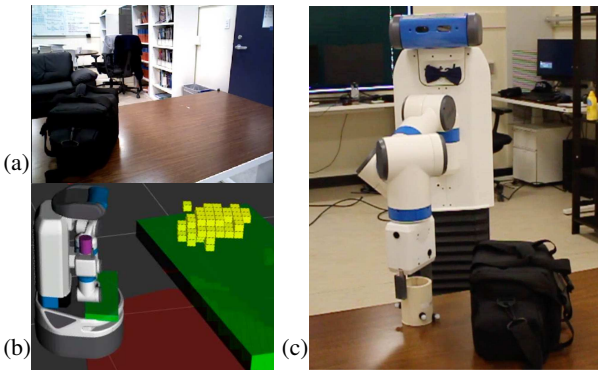

Fig. 7: In order to complete the HARS specified by the user (as shown in Fig. 6a), a navigation goal region is established to approach the placement location. (a), (b) The robot Fetch uses its onboard cameras to inspect the table to detect any initially unknown object. (c) The robot places the wooden cylinder on the table. In this particular case, a bag has been placed on the table, partially occluding the placement goal region.

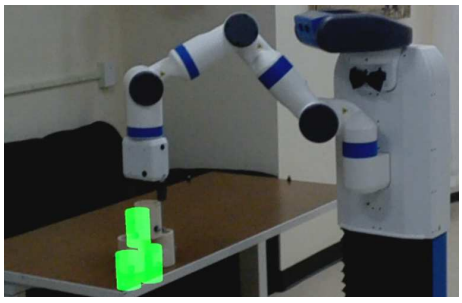

Fig. 8: The Fetch received a HARS to stack three wooden cylinders on a table. The arrange of virtual cylinders can be observed superimposed the real cylinders.

the objects, picks up one of the objects, then the robot moves close to the placement region, and finally the Fetch places the object. These consecutive motion queries are repeated n times, where n corresponds to the number of virtual objects set via the AR interface. Fig. 8 depicts the Fetch after solving and executing the start-to-goal-region motion queries to arrange a group of objects on the table.

VI. Discussion and Future Work

We have presented a new high-level AR interface to command and collaborate with robots. This new interface allows the user to specify high-level goals, which we have called High-level Augmented Reality Specifications (HARS), while letting the robot determine the feasible motions that achieve these goals. We have also proposed to use motion planning techniques, which can solve robot motion queries with high-level goals that correspond to regions in the configuration space. This latter characteristic increases the autonomy and autonomous decision making capabilities of a robot, thus allowing the user to provide more general and open requests.

To evaluate the proposed approach, we presented a proof of concept for attending one case of HARS, in which a user can manipulate a virtual object in order to command a robot to fetch and place a real object. We also demonstrated that

such a type of HARS can be sequenced, thus allowing the user to give robots more complicated tasks such as arranging and stacking multiple objects.

This paper lays the foundation for further investigations of ways that AR and planning can be integrated in human-robot collaboration applications. One direction that will be investigated in future work is the perceived utility of our approach through user studies. Besides considering potential efficiency improvements in collaborative tasks, it would also be interesting, for example, to assess the combination of the proposed approach with other interaction modalities, such as speech and gestures.

ACKNOWLEDGMENT

This work would have not been possible without the support of Zachary Kingston in extending MoveIt [35] and OMPL [36] for the presented motion planning strategies. The authors would like to thank Samantha Gilmore for her initial exploration and development over the Magic Leap as part of the COMP 650 (Physical Computing) class at Rice University. The authors would also like to thank Vladimir Vincan for his insight and discussion in the coordinate systems consistency.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] 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, no. 2, pp. 129–147, Feb. 2012.

[3] S. A. Green, M. Billinghurst, X. Chen, and J. G. Chase, "Human-Robot Collaboration: A Literature Review and Augmented Reality Approach in Design," *Intl. J. of Advanced Robotic Systems*, vol. 5, no. 1, 2008.

[4] S. Charoenseang and T. Tonggoed, "Human–Robot Collaboration with Augmented Reality," in *C. Stephanidis (ed) HCI 2011 – Posters' Extended Abstracts.* Springer-Verlag, 2011, vol. 174, pp. 93–97.

[5] C. Perez Quintero, S. Li, M. K. Pan, W. P. Chan, H. Machiel Van der Loos, and E. Croft, "Robot Programming Through Augmented Trajectories in Augmented Reality," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. IEEE, Oct 2018, pp. 1838–1844.

[6] S. Ong, A. Yew, N. Thanigaivel, and A. Nee, "Augmented reality-assisted robot programming system for industrial applications," *Robotics and Computer-Integrated Manufacturing*, vol. 61, 2020.

[7] D. Whitney, E. Rosen, E. Phillips, G. Konidaris, and S. Tellex, "Comparing Robot Grasping Teleoperation across Desktop and Virtual Reality with ROS Reality," in *Intl. Symp. on Robotics Research*, 2017, pp. 1–16.

[8] D. Whitney, E. Rosen, D. Ullman, E. Phillips, and S. Tellex, "ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2018, pp. 5018–5025.

[9] S. Tellex, P. Thaker, R. Deits, T. Kollar, and N. Roy, "Toward Information Theoretic Human-Robot Dialog," in *Robotics: Science and Systems (RSS)*. 2012.

[10] R. A. Knepper, S. Tellex, A. Li, N. Roy, and D. Rus, "Recovering from failure by asking for help," *Autonomous Robots*, vol. 39, no. 3, pp. 347–362, Oct 2015.

[11] D. Whitney, E. Rosen, J. MacGlashan, L. L. S. Wong, and S. Tellex, "Reducing errors in object-fetching interactions through social feedback," in *IEEE Intl. Conf. on Robotics and Automation*. IEEE, May 2017, pp. 1006–1013.

[13] E. Sibirtseva, D. Kontogiorgos, O. Nykvist, H. Karaoguz, I. Leite, J. Gustafson, and D. Kragic, "A Comparison of Visualisation Methods for Disambiguating Verbal Requests in Human-Robot Interaction," in *IEEE Intl. Symp. on Robot and Human Interactive Communication*. 2018, pp. 43–50.

[12] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, Nov 2018.

[14] H. Hedayati, M. Walker, and D. Szafir, "Improving Collocated Robot Teleoperation with Augmented Reality," in *ACM/IEEE Intl. Conf. on Human-Robot Interaction*. 2018, pp. 78–86.

[15] H. Fang, S. Ong, and A. Nee, "Interactive robot trajectory planning and simulation using Augmented Reality," *Robotics and Computer-Integrated Manufacturing*, vol. 28, pp. 227–237, 2012.

[16] A. Gaschler, M. Springer, M. Rickert, and A. Knoll, "Intuitive robot tasks with augmented reality and virtual obstacles," in *IEEE Intl. Conf. on Robotics and Automation*. IEEE, May 2014, pp. 6026–6031.

[17] M. Ostanin and A. Klimchik, "Interactive Robot Programing Using Mixed Reality," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 50–55, 2018.

[18] M. E. Walker, H. Hedayati, and D. Szafir, "Robot Teleoperation with Augmented Reality Virtual Surrogates," in *ACM/IEEE Intl. Conf. on Human-Robot Interaction*, 2019, pp. 202–210.

[19] H. Peng, J. Briggs, C.-Y. Wang, K. Guo, J. Kider, S. Mueller, P. Baudisch, and F. Guimbretière, "RoMA: Interactive Fabrication with Augmented Reality and a Robotic 3D Printer," in *CHI Conf. on Human Factors in Computing Systems*. 2018.

[20] J. Neves, D. Serrario, and J. N. Pires, "Application of mixed reality in robot manipulator programming," *Industrial Robot*, vol. 45, no. 6, pp. 784–793, 2018.

[21] J. Chong, S. Ong, A. Nee, and K. Youcef-Youmi, "Robot programming using augmented reality: An interactive method for planning collision-free paths," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 3, pp. 689–701, 2009.

[22] Y. Sarai and Y. Maeda, "Robot programming for manipulators through volume sweeping and augmented reality," in *IEEE Conf. on Automation Science and Engineering*. 2017, pp. 302–307.

[23] P. M. Grice and C. C. Kemp, "In-home and remote use of robotic body surrogates by people with profound motor deficits," *PLOS ONE*, vol. 14, no. 3, p. e0212904, Mar 2019.

[24] B. Akan, A. Ameri, B. Cürüklü, and L. Asplund, "Intuitive industrial robot programming through incremental multimodal language and augmented reality," in *IEEE Intl. Conf. on Robotics and Automation*, May 2011, pp. 3934–3939.

[25] S. Blankemeyer, R. Wiemann, L. Posniak, C. Pregizer, and A. Raatz, "Intuitive robot programming using augmented reality," *Procedia CIRP*, vol. 76, pp. 155–160, 2018.

[26] T. Chakraborti, S. Sreedharan, A. Kulkarni, and S. Kambhampati, "Projection-Aware Task Planning and Execution for Human-in-the-Loop Operation of Robots in a Mixed-Reality Workspace," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. 2018, pp. 4476–4482.

[27] E. Rosen, D. Whitney, E. Phillips, G. Chien, J. Tompkin, G. Konidaris, and S. Tellex, "Communicating Robot Arm Motion Intent Through Mixed Reality Head-mounted Displays," in *Intl. Symp. on Robotics Research*, Aug 2017, pp. 1–16.

[28] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp Pose Detection in Point Clouds," *The Intl. J. of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, Dec 2017.

[29] J. D. Hernández, M. Moll, and L. E. Kavraki, "Lazy Evaluation of Goal Specifications Guided by Motion Planning," in *IEEE Intl. Conf. on Robotics and Automation*. May 2019, pp. 944–950.

[30] J. D. Hernández, M. Moll, E. Vidal, M. Carreras, and L. E. Kavraki, "Planning feasible and safe paths online for autonomous underwater vehicles in unknown environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*. 2016, pp. 1313–1320.

[31] M. 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 Intl. Joint Conf. on Artificial Intelligence*, 2016.

[32] Vicon. Vicon Motion Capture System. [Online]. Available: https://www.vicon.com

[33] Magic Leap, Inc. Magic Leap One. [Online]. Available: https://www.magicleap.com/

[34] Unity Technologies. Unity. [Online]. Available: https://unity.com/

[35] I. A. Şucan and S. Chitta, "MoveIt!" [Online]. Available: http://moveit.ros.org

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