Robonaut 2 and You: Specifying and Executing Complex Operations

William Baker[1], Zachary Kingston[2], Mark Moll[2], Julia Badger[3], and Lydia Kavraki[2]

***Abstract*— Crew time is a precious resource due to the expense of trained human operators in space. Efficient caretaker robots could lessen the manual labor load required by frequent vehicular and life support maintenance tasks, freeing astronaut time for scientific mission objectives. Humanoid robots can fluidly exist alongside human counterparts due to their form, but they are complex and high-dimensional platforms.**

This paper describes a system that human operators can use to maneuver Robonaut 2 (R2), a dexterous humanoid robot developed by NASA to research co-robotic applications. The system includes a specification of constraints used to describe operations, and the supporting planning framework that solves constrained problems on R2 at interactive speeds. The paper is developed in reference to an illustrative, typical example of an operation R2 performs to highlight the challenges inherent to the problems R2 must face. Finally, the interface and planner is validated through a case-study using the guiding example on the physical robot in a simulated microgravity environment. This work reveals the complexity of employing humanoid caretaker robots and suggest solutions that are broadly applicable.

I. Introduction

Robonaut 2 (R2), shown in Figure 1, is a complex humanoid robot capable of dexterous manipulation [1]. It is composed of an anthropomorphic upper body and two leg-like appendages that each feature seven degrees-of-freedom, for a total of 34 degrees-of-freedom in the main body. Its twelve degree-of-freedom hands can execute dexterous manipulation tasks like using human tools and handling fabric surfaces [2]. The end-effector on the legs has a multi-purpose gripper that can attach to the ubiquitous handrails in the International Space Station (ISS). R2 is designed to be an assistant to the crew currently stationed on-board the ISS, as well as a test-bed for robotic caretaker technology that is required in future space exploration [3]. Repetitive and dangerous operations required for space exploration could be accomplished by R2 and other robots, allowing more time for astronauts to achieve scientific mission objectives. For example, the logistics and organization of cargo takes an immense fraction of crew time and is also physically strenuous. In future space exploration, robots like R2 could unload a dormant logistics module for a spacecraft in cislunar orbit in advance of crew arrival. On the ISS today, cargo is stored within a general purpose container called a Cargo Transfer Bag (CTB). R2 could retrieve and manipulate a desired CTB

[1]Oceaneering Space Systems, Houston, TX 77058 `william.c.baker-1 at nasa.gov`

[2]Rice University Department of Computer Science, Houston, TX 77005 `{zak, mmoll, kavraki} at rice.edu`. Work on this paper by ZK, MM, and LK was sponsored in part by NSF IIS1317849

[3]NASA-Johnson Space Center, Houston, TX 77058 `julia.m.badger at nasa.gov`

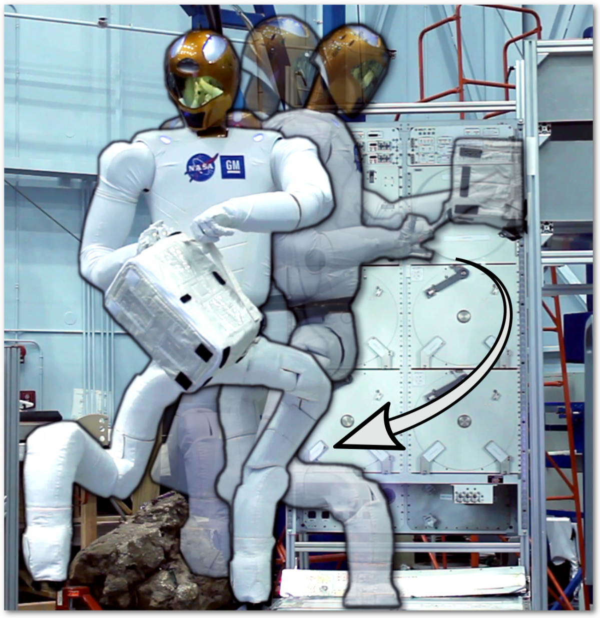


Fig. 1. R2 executing the CTB retrieval scenario. R2 has to respect many different constraints when planning motions along the course of the operation. For example, when holding the CTB with both hands, there is a constraint imposed on the hands as to not drop the CTB. The system allows consideration of constraints such as the example given and provides means for the operator to easily specify novel ones without leaving the high-level interface. There are four stills superimposed in this image, from back to front: R2 approaching the cargo rack, R2 unbuckling the restraint holding the CTB, R2 grabbing the CTB with both hands, and the final image of R2 bimanually manipulating the CTB.

from a logistics module to another location. CTB retrieval will be used as an illustrative example throughout the paper, as it is representative of the complexities that arise when commanding a system like R2. The goal of this work is to create an interface and supporting technology necessary so that an expert operator can quickly and intuitively describe operations, like CTB retrieval, and the motion constraints inherent to the operation's subtasks. The system described simplifies the job of the operator so that it is possible to command a complex robot to do useful work.

This paper describes a system to control R2 efficiently and effectively in practice. An interface that enables an expert operator to define and select specific goals and/or path constraints while operating the robot is presented. The minimum requirements of what the interface should offer and the conveniences it affords to operators to save time communicating their commands is discussed. One of the critical facets of commanding a robot is the specification of an operation to execute. An operation is a sequence of subtasks, each composed of a start pose and goal constraints, along with a set of constraints that impose geometric limits on the robot throughout the course of the operation. Operations are distinct from *tasks* as the operator plans the sequence of subtasks to achieve a goal, not the system itself. In the case of R2 and CTB retrieval, an operation that requires bimanual manipulation of the bag induces a constraint between the robot's palms. A novel constraint specification and required techniques to make motion planning tractable in operations such as CTB retrieval are presented. The motion planner takes in a specification of a subtask and outputs a sequence of waypoints, each a full configuration of the robot. To overcome the dimensionality and constraints of the planning problem, the motion planner is a sampling-based, constrained motion planning algorithm [4]. The unique aspects of the motion planner and how the constraint specification is leveraged is also discussed.

The paper is organized as follows. The challenges of making a complex, humanoid robot execute complex operations are described in Section II. The minimum requirements of the interface and the capabilities it provides are covered in Section III. The description of the constraints on the operation, how the constraints are processed, and how they are used for planning is described in Section IV. The system is validated through completing the CTB retrieval operation on the physical robot in a simulated microgravity environment. Figure 1 shows R2 throughout the execution of the CTB retrieval scenario; the details of the study are discussed in Section V. Finally, concluding remarks and a discussion of future directions are left in Section VI.

II. Challenges and Related Work

There are many challenges to consider when designing a system to command a robot like R2. This problem is an extreme case of a very high-dimensional system performing nominally hard tasks in tandem, such as mobile manipulation, constrained motion planning, and dexterous manipulation. Solving each of these together, and having a human operator describe these complex problems is an unresolved problem [5]. At the DARPA Robotics Challenge (DRC), operator control was paramount to the success of the teams, and operator errors were a large source of problems in execution [6]–[9]. This trend can be seen in other robotics competitions and for other form-factors of robotic platforms [10], [11]. The interface must afford the operator the ability to craft commands with a level of complexity suited towards the motion being planned [12]. Complexity arises from the operation at hand and the robot itself, and creates a barrier in usability between the operator and the motion of the robot. The novel system described here is one way to bridge the two together.

The guiding example of CTB extraction is typical of operations R2 performs. Although simple for a human to complete, this operation begets a multitude of problems for a robot to consider. These problems range over the entire operation, starting with approaching the CTB, grasping it, and continuing through the following manipulation subtasks. When approaching the cargo rack that contains the CTB, there is the issue of positioning the torso relative to the CTB. Torso positioning is critical, as it determines the space where the workspace of the arms and the CTB overlap. A small overlap means little room for manipulation. If R2 has both of its gripping end-effectors grasped onto handrails for leverage, then there is a constraint on the motion of the torso, as it cannot move further than the extent of either of the legs. Constraints like these must be taken into account when planning. Locating the pose of the CTB relative to the robot and grasping it is another set of problems; exteroceptive sensors must detect the CTB, and the dexterous hands of R2 must grasp the CTB's pliant canvas straps. The CTB is an unwieldy object; to manipulate it R2 must grab onto both sides of the bag. When grasped, the relative motion of the arms are constrained to one another; they must maintain the same relative position to not drop the CTB. Problems like these generalize to a larger class of issues that are faced when planning for manipulation operations with a complex humanoid robot. Representing these problems and solving them is critical to the usability of the robot.

The old-fashioned and error-prone way of solving the above problem might be something like the following. First, a path representing the motion of the CTB would be generated. Each pose of the CTB along the path also has a desired pose for both the left and right hands relative to itself. A configuration of the arms would be found that satisfies the desired poses for each waypoint. This could potentially generate a valid path for the arms, however this is not likely because there are no guarantees that the identified configurations are connectable. Additional constraints could also not be considered, nor larger plans that involve other parts of R2's body. Any approach must consider the constraints as a whole, and provide a way for the user to specify them in a unified way.

III. Operator Interface

Complex operations like CTB retrieval can be simplified into a sequence of subtasks, in which the operator specifies a set of goal constraints on a set of frames, and a set of constraints that apply through the entire subtask. The job of the system is to abstract away the details of motion planning by providing the ability to quickly define constraints imposed on the desire motion of the robot. In this section the aspects of the interface that enable the operator to efficiently command R2 are covered. This novel solution offers great improvements in the ability to define constraints, update allowable collisions, automate trivial processes, and interact with virtual objects. Constraints are left as an abstraction in this section, they are covered in more detail in Section IV.

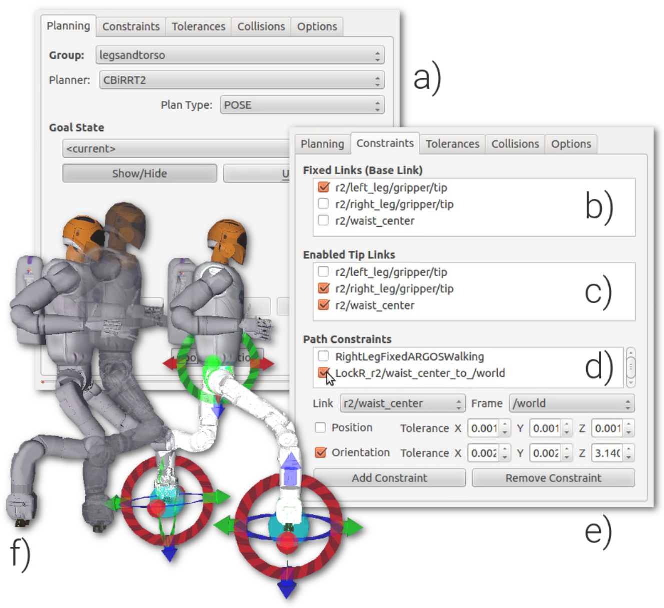
While accomplishing complex operations using *MoveIt!* [13] is possible, it is extremely burdensome on the operator. The operator interface, shown in Figure 2, features a minimalist design as to reduce the number of button and mouse clicks it takes an operator to define, plan, and execute a subtask. Similar to the *MoveIt!* interface, a virtual model of the robot with a drag-and-drop overlay (see Figure 2f) enables the robot's end-effectors to be dragged into a desired pose. The desired pose is generalized by the interface into a constraint, which can be tuned by the operator to define more general goal conditions. The capabilities of this interface have been extended by enabling the virtual model to be toggled with a single button click and the overlay to be customized to enable and disable the various end-effectors. The minimalist design requirement also complements the desire to maximize operator's view of the live sensor data streaming from the robot. Minimizing the size of the motion planning GUI and eliminating complicated menu items creates a bare-bones operator interface designed to quickly and efficiently move the robot from one configuration to another.

A. *Selecting and Appending Constraints*

Throughout a complex operation, such as the removal of the CTB, each subtask implicitly defines constraints on the motion of the robot. For example, approaching the CTB from a distance (shown in Figure 4a) requires the definition of a fixed link, i.e., a link on the robot that remains constant in position and orientation throughout the duration of the motion. Specifically, the constraints for the first motion of the CTB removal operation would be defined as:

- The left leg end-effector is fixed.
- The goal constraint of the torso is such that the object's grasping points are reachable using the upper body's end-effectors.
- The right leg is positioned such that an automated handrail rendezvous procedure can be used to affix the end-effector to the handrail.

Previously, when using the *MoveIt!* interface, there existed no method to quickly define a constraint derived the current state of the robot. In order define a new constraint, the user would first use an external tool to find the geometry of the desired constraint in some reference frame. The user then would programmatically construct the desired constraint using an external application. Moreover, each time the robot moves or grasps an object in a different manner, the constraint would often need to be manually redefined and updated (i.e., repeat the aforementioned process after each motion). This interface combats the tedious constraint re-definition process by implementing a method for toggling fixed link constraints that are automatically updated to the current state of the robot. For example, this is extremely useful during a walking task which requires multiple steps where R2 is switching base frames from one gripper to the other, and each gripper hold occurs at different positions within the environment. This feature is also extended such that any arbitrary link can be constrained in reference to any known coordinate frame (e.g., the torso to world, right hand relative to left hand). For example, during bimanual manipulation, the two grasp points must remain constant to each other, as to not stretch, contort, or destroy the object. This requires a constraint to be defined between the two hands to keep their relative pose constant, which must be updated to the current state of the robot. This is demonstrated with the removal of the CTB



Group: legsandtorso
Planner: CBiRRT2
Plan Type: POSE
<current>

Fixed Links (Base Link):
- [✓] r2/left_leg/gripper/tip
- [ ] r2/right_leg/gripper/tip
- [ ] r2/waist_center

Enabled Tip Links:
- [ ] r2/left_leg/gripper/tip
- [✓] r2/right_leg/gripper/tip
- [✓] r2/waist_center

Path Constraints:
- [ ] RightLegFixedARGOSWalking
- [✓] LockR_r2/waist_center_to_/world

Link: r2/waist_center Frame: /world
[ ] Position Tolerance X 0.001 Y 0.001 Z 0.001
[✓] Orientation Tolerance X 0.002 Y 0.002 Z 3.140

Fig. 2. Elements from the operation planning interface. **a)** The minimalist front-page features a tabbed context with options designed to quickly switch between planning for different parts of the robot, different planners, saved goal states, toggles for the virtual model and path animation (shown in **f**), and the plan and execute buttons (hidden, bottom center). **b)** The *fixed links* selection menu adds constraints that fix each selected link at the link's current pose in the global frame. **c)** The *enabled tip links* selection menu is used to toggle goal constraints for the specified end-effector link. **d)** The *path constraints* selection menu is used to toggle defined constraints that are to be satisfied for the duration of the generated path. **e)** Addition/removal of constraints by defining position and orientation tolerances for a link in a desired reference frame. More information about constraints is given in Section IV. Constraints defined with this interface will be shown in **d**. **f)** The virtual model of the current robot configuration (left), path animation (center), and goal configuration (right).

from the storage rack (Figure 4c); this motion pulls the CTB towards the robot's chest to begin the removal process.

For certain motion planning requests, complex or multiple goal regions and orientations may be allowed. This type of request currently requires an external interface to define the goal constraints, as this is not yet been integrated with the intuitive drag-and-drop method described above. This interface offers the ability to accept a remote plan request from an external application and filter any the requests through the active constraints defined in the GUI. This process will intelligently append (i.e., search for and remove duplicates, etc.) to the constraints already defined in the remote request. This process ensures that any remote motion request will attempt to satisfy the path and fixed link constraints that are enabled in the GUI (Figure 2b, d).

B. *Allowable Collision Matrix*

During object manipulation, the robot will be required to come in contact with objects; these collisions must be considered during planning. There must exist a method to toggle collision checking between various frames and/or objects by modifying the Allowed Collision Matrix (ACM), a matrix whose entries describe which frames are allowed to collide with one another. However, although a programmatic method for accessing and modifying entries in the ACM is available, the *MoveIt!* motion planning interface does not provide a method for an operator to modify the ACM interactively. This interface provides a context menu to easily configure and update the ACM by individually specifying frames or groups of frames that are allowed to collide with any given object (e.g., right index and middle finger may collide with the CTB). Each of these allowable collisions must be specified and updated before and/or after every subtask throughout the entire operation. For example, the ACM must also be updated when R2 releases a gripping end-effector and traverses to another handrail. Note that the release of a handrail signifies that the gripper can be commanded to its next goal. However, collisions with the previously grappled handrail are no longer acceptable.

C. Automation of Trivial Procedures

When generating a motion planning request, an operator is required to specify the starting state of the robot as well as the origin and size of the planning workspace, i.e., the search space in which the planner samples for valid configurations. The dimensions of the planning workspace should be optimized by the operator; specifying a workspace that is too small may limit the ability of the planner to find a valid path, while an overly large workspace will leave the planner searching for valid configurations in inapplicable locations and thus possibly increasing computation time. During operation, as the robot traverses large distances, the planning workspace of the robot must be updated; this often requires re-centering of the workspace about the root of the robot. This novel interface eliminates the repetitive steps of specifying the starting state and origin of the planning workspace for each motion. The start state of the robot is continuously updated with the current state of the robot's joints. The origin of the planning workspace can be configured to always be centered about a desired link (e.g., the base link of the robot, or R2's pelvis). This allows the origin of the workspace to be automatically updated prior to passing the motion request to the planner, eliminating the need to manually update the origin and or provide arbitrarily large dimensions for the workspace.

D. Virtual Objects

The use of a virtual object greatly augments our interface by further eliminating repetitive tasks and implementing new capabilities. Virtual objects, such as those described in [14] can be used to define a set of goal and/or path constraints as well as any allowable collisions (shown in Figure 4b, c). This virtual object has a series of goal constraints, i.e., waypoints, marked by floating end-effectors. These waypoints contain embedded information not only limited to the position and orientation for a desired end effector, but also details about the allowable collisions. For instance, the waypoints that are in contact with the virtual CTB object automatically update the ACM to allow for collisions between R2's fingers and the CTB. Use of these virtual objects can further reduce the use of the operator interface, therefore enabling the automation of the subtasks.

IV. Specifying and Solving Constraints

When specifying a subtask in an operation for a complex system like R2, constraints naturally arise from high-level requirements and low-level physical limits. Both requirements and limits are represented in a unified constraint framework that translates constraints into geometric limits on the robot. As detailed in Section III, the developed interface provides a way for users to enable/disable these constraints on the fly in order to accomplish whatever subtask is at hand. This section describes in more detail the types of constraints that arise for many humanoid robot systems like R2. Constraint ordering is discussed in the context of efficiently computing solutions that satisfy all constraints. Finally, the novel planner and how the constraint specification is used to enforce constraints along the planned path is presented.

The CTB extraction scenario contains common, archetypal examples of constraints that can be imposed on humanoid systems. For example, in order to be stable in microgravity when manipulating the CTB, it is necessary to attach both of the grippers to handrails to take a strong stance. This fixes the pose of the grippers, which creates a dependency in the configuration of the legs when moving the torso. Similarly, bimanual manipulation of the CTB creates a constraint between the palms of R2's hands, as they must maintain a fixed relative pose with respect to each other to continue to grip the bag. These constraints can be broken up into two broad categories: global coordinate frame constraints and relative coordinate frame constraints. Global constraints are attached to the world frame and are invariant to the robot's configuration; the robot's current configuration has no bearing on the meaning of the constraint. Relative constraints are attached to a frame on the robot's structure, and change their meaning in world coordinates as the robot changes configurations. By defining a constraint of one frame on the robot to another, an implicit linkage is defined that binds the movement of the frames together. This posits a harder problem to solve, but some tasks are impossible to specify using only global frame constraints. There is no distinction between the two types of constraints in the interface as they are defined by the same set of parameters; methods that manipulate constraints have no notion of whether the constraints are relative or global. The semantic difference between global and relative constraints occurs when projecting onto the constraint manifold, discussed later in Section IV-B.

We define a constraint c_i^j as a set of bounding volumes on the position and orientation of a frame j attached to another frame i of the robot. These constraint volumes are defined as a subset of the robot's configuration space where, at these configurations, the constrained frame of the robot is within some geometric bounds. The geometric bounds are defined as either position or orientation volumes. Position volumes are defined as geometric primitives such as rectangular prisms or spheres, or arbitrary meshes. Orientation volumes are defined as angular tolerances about the X-, Y-, and Z-axes of the base frame. These volumes are defined within some base frame as detailed above, either the world frame or a frame on the robot. For the example constraint imposed during bimanual manipulation, a tightly bounded orientation and position volume is created from one palm of the hand to the other's grasped location. This keeps their relative pose consistent throughout the path. The definition of a constraint is one way to specify general geometric limitations on a frame of the robot. It has the benefits of not prescribing the implementation of the constraints, and allowing for definition of relative and global constraints using the same set of parameters.

A. Ranking and Sorting Constraints

In order to solve a subtask with constraints, the configuration of the robot must be made to satisfy each of the defined limitations. Solving all constraints simultaneously is very difficult, in part because the relative pose constraints create dependencies between the constraints. To solve the problem of planning with multiple constraints, the constraints must first be ordered in a way that respects these dependencies. The sorting step takes place after constraints have been selected for the operation, but before the planner starts using the constraints to solve the posed subtask. Simple conflicts in the proposed set of constraints are detected during this step. When sorting the constraints and building the structure of the problem, over-constrained frames and redundant constraints are found and then returned to the user so they can revise their specification. Cyclic dependencies are not considered and are explicitly disallowed when a subtask is specified. It is left to future work to consider compositions of constraints which could potentially affect each other.

The robot, when unconstrained, is modeled as a kinematic tree. A kinematic tree is constructed of frames attached to the robot and the relative transformations that chain successive frames together. A kinematic tree has a root frame, which all other frames in the tree descend from. For the purpose of constraint sorting, the kinematic tree is formalized as $K = (F, R)$, where F is the set of all the robot's frames and $(u, v) \in R$ is the relative transformation of a frame u to frame v. It is assumed all frames in the tree are descended from the root frame f_R. A subtask for the robot is specified as a set of goal constraints for frames from the kinematic tree of the robot. Any number of frames can be selected to have goal constraints. A goal configuration is a configuration of the robot so that all goal constraints are satisfied. For R2 to move through the environment, R2 must be affixed to the structure of the ISS. This is modeled as a global frame constraint with a tightly bounded constraint volume, and the constrained frame becomes the root frame of the kinematic tree. Generally, the root frame will be one of R2's grippers if executing a full body motion. The torso can also be used as the root frame if executing exclusively upper-body movement.

Relative frame constraints depend on the configuration of the robot, as their constraint volumes are defined with respect to a frame of the robot. If a solution that satisfies a relative constraint is found, and then another constraint is solved for, the solution configuration may no longer satisfy the relative constraint. Constraints are ordered so that more central constraints to the kinematic structure of the robot are solved for first, and that global constraints are satisfied before relative ones. For relative constraints, this means that frames that could affect the pose of later frames are locked in before solving dependent frames. The procedure for constraint sorting is shown in Figure 3. The ordered constraints are now used to assist the planner in exploring the constrained configuration space of the robot.

```
1:  **procedure** ConstraintSort($K, f_R, C_G, C_R$)
2:      $K_B \leftarrow (F, R \cup \{$ All edges in $R$ reversed $\})$
3:      $D_{f_R} \leftarrow$ Map of $f \in F$ to distance from $f_R$ in $K_B$
4:      Sort $C_G$ in ascending order based on $D_{f_R}$

5:      $K_R \leftarrow K_B$ with an additional edge from $f_i$ to $f_j$ for
            each relative constraint $c_i^j \in C_R$
6:      $k \leftarrow 0$
7:      $D_R \leftarrow$ Map of $f \in F$ to $\infty$
8:      **for** each global constraint $c_i^j \in C_G$ in order **do**
9:          $D_c \leftarrow$ Map of $f \in F$ to distance from $f_i$ in $K_R$
10:         $D_R \leftarrow$ Map of $f \in F$ to the minimum value
                of $D_R(f)$ and $D_c(f) + k$
11:         $k \leftarrow k + 1$
12:     Sort $C_R$ in ascending order based on $D_R$
13:     **return** $C_G + C_R$
```

Fig. 3. The constraint sorting procedure. This procedure takes as input the kinematic tree of the robot K, the rooted frame f_R, and the global and relative constraints C_G and C_R. Line 2 builds K_B, a bidirectional graph with all edge from K, as well as those edges reversed. Line 5 builds K_R, which adds an edge to represent the constrained transformation for each relative constraint. In the for loop starting on line 8, a map D_R is computed which contains the minimum distance of each frame $f \in F$ to a globally constrained frame. k is an index in the sorted global constraint list that gives precedence to to frames closer to the root constraint.

B. Projection onto the Constraint Manifold

A key requirement of any constrained motion planner, including the one used here, is the ability to find configurations that satisfy the constraints. This is accomplished through a projection routine offered by the constraints. Given a set of constraints and a starting configuration, a nearby configuration that satisfies the constraints using Jacobian Inverse Kinematics (IK) is determined. Given a constrained frame and its desired constraint volume, the frame is projected into the constraint volume by first creating a twist vector in the workspace from the frame's current position to the center of the constraint volume. The configuration of the robot is iteratively updated until a solution is found using constrained frame's Jacobian to follow the twist vector. To account for multiple active constraints, a hierarchical IK solver is used [15]. In the order according to the sorting method detailed in Section IV-A, the joint velocities from the constraints are projected into the preceding constrained frame's Jacobian's null space. Null space projection into the Jacobian adds joint velocities that do not affect the result of the preceding velocity, which is satisfactory under the assumption that the sorting has ordered the constraints in such a way that each successive constraint does not affect the previous solution. Singularity robustness is maintained using the damped least-squares Jacobian pseudo-inverse [16], [17]. This method is not complete, as during the gradient descent in Jacobian IK the procedure can be caught in local minima and fail to find a valid solution. An attempt to avoid this is implemented by projecting random velocities into the null space of the final Jacobian, and randomly restarting gradient descent if little progress towards a solution is being made.

Accounting for relative frame constraints is trickier, as they are dependent on the configuration of the robot. To accommodate these constraints, the IK solver satisfies relative constraints incrementally. Due to constraint sorting, valid configurations that satisfy each relative constraint successively are found, as motion to solve the previous constraints does not affect the constraints proceeding them. The projection procedure first finds a configuration that satisfies all global constraints using the IK solver described above. Iteratively, one relative constraint is added to the IK problem, using the current solution configuration as the starting point, and the next configuration is found. For example, when bimanually manipulating the CTB, the goal configuration is specified by a global constraint on the right palm, and a relative constraint from the right palm to the left palm. First, a valid configuration of the robot that satisfies the global constraint of the right palm is found. Other global constraints specified are solved for simultaneously at this time. Next, the pose of the left palm is found using the current pose of the right palm. This is then repeated done for any number of other relative constraints that may follow.

C. Subtask Planning

An operation such as the CTB extraction is composed of many subtasks, each specified by the initial configuration of the robot, the goal constraints on a set of frames, and a set of path constraints that must be satisfied on every waypoint in the path. Subtasks can be specified through the interface by the operator, or scripted for longer sequences. The sequence of waypoints in the path corresponds to successive configurations of the robot from the initial configuration to a goal configuration that satisfies the goal constraints. The waypoint configurations satisfy the constraints, are collision-free with obstacles in the workspace, and are within the joint limits of the robot. R2 is assumed to be quasi-static, with a nominal load capacity of nine kilograms. This frees the planner from reasoning over the dynamics of the system, so assumptions from geometric planning hold. R2 also operates in a microgravity environment, and does not need to consider problems like stability that are found in other walking humanoids. Waypoints are not time-parameterized; they are fed into a trajectory generator after planning to create a continuous trajectory that the robot can execute which then takes the dynamics of the robot into account. The controller architecture for R2 is discussed in [18].

In order to plan under task constraints, a sampling-based constrained motion planning algorithm is used [4]. [19] presents prototypical components of constrained sampling-based planners. The approach presented here is a modification to the *CBiRRT2* algorithm [20], using the described notion of constraints. The planner only has access to one primitive operation offered by the constraints, projection. The projection routine takes a configuration of the robot and modifies it so that the configuration satisfies the constraints, while still preserving some of the locality of the configuration relative to other configurations. Projection is used both in exploration of the space and in the growth of the trees in *CBiRRT2*. An interpolation method is used in order to generate additional waypoints along the path which is critical to the success of path generation. The path generated by the algorithm is a rather coarse approximation of a complete path, and it is possible for the trajectory generation algorithm to bridge the gaps between waypoints with configurations that are invalid. The splining or blending method employed by the trajectory generation algorithm does not have knowledge of the constraints, and could invalidate them when attempting to smoothly connect waypoints. Interpolation refines the path as a post-processing step, adding waypoints until a continuous path is sufficiently approximated, while still respecting the task constraints. The interpolated path is then fed to the trajectory generator to create a continuous, valid trajectory.

V. Case Study

In this section, analysis from the case-study of executing the CTB retrieval scenario on the R2 hardware is presented. The interface and execution is compared to previous approaches, and improvements made and future work are discussed. The software uses ROS, the Robot Operating System [21], as communication middleware for the higher level planning processes. The planner is implemented with the MoveIt! [13] framework, using a planner implemented in OMPL, the Open Motion Planning Library [22]. As R2 is designed for microgravity environments, it cannot operate fully when under Earth's gravity. R2 is connected by a gimbal to the Active Response Gravity Offload System (ARGOS) [23] at NASA-Johnson Space Center. ARGOS allows R2 to achieve almost full mobility and to traverse the simulated ISS handrail test mock-up, shown in Figure 4. The ARGOS system allows mobility along all three translation planes (the X-, Y-, and Z-axes) but only one rotational degree-of-freedom about the Z-axis, or yaw. The details of ARGOS relevant to the example are discussed in Section V-B.

A. Cargo Transfer Bag (CTB) Manipulation

Figure 4 shows stills from a single execution of the CTB scenario. The first part of the scenario is the approach phase, where R2 moves from a stow position to in front of the CTB (shown in Figure 4a). Planning for the motion to move from start to a manipulation-ready pose has one of the most common constraints used regularly by operators, as it must be satisfied to move about in ARGOS. The relative constraint is defined from the foot to the torso, which prevents the torso from rotating about the X- and Y-axes. This satisfies the constraint imposed on R2 by ARGOS so it can operate on Earth. R2 next grasps another handrail with its right leg,

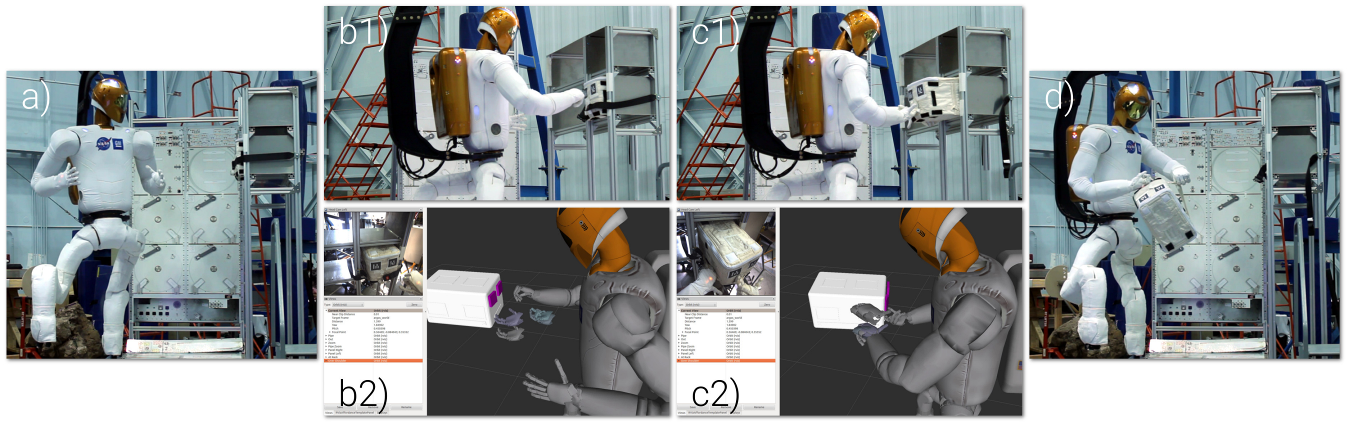


Fig. 4. A sequence of images showing R2 completing the CTB extraction scenario while in ARGOS. Hardware tests with R2 are done in the simulated ISS handrail test mock-up at NASA-Johnson Space Center. **a)** R2 approaches the logistics module rack mockup that contains the CTB. It first must attach its right leg to another handrail (below image cut) so it has sufficient leverage to undo the strap holding the CTB. **b1)** R2 mid-execution of a trajectory planned to a goal pose relative to the CTB. **b2)** Operator's view in the interface of the sensed location of the CTB and R2's reported configuration. The purple squares represent the fiducial tags on the CTB. The white box is the virtual object representing the CTB. The floating hands are a sequence of goal poses for the hand to accomplish the task of grabbing the CTB. **c1)** R2 after finishing execution of a trajectory grabbing the CTB with its left hand. **c2)** Operators view of the same moment as **c1**. **d)** R2 holding the retrieved CTB bag, returning to start.

creating a constraint on its motion. To begin the grasping process, a goal is specified as a desired pose above the handrail for the right foot, with the ARGOS constraints still active. An autonomous docking script then controls the descent of the grippers onto the handrail. Grasping both handrails is done so R2 can have sufficient leverage to unbuckle the strap containing the CTB, as well as stability when manipulating the CTB. Without both legs down, the force from pressing the button on the CTB is sufficient to bend the leg slightly due to its compliance. Further motions moving the arms around change the center of R2's mass, which causes drift in the pose of the torso as ARGOS compensates for the change. Any further torso movement requires planning both with the relative constraint imposed by the gimbal and the constraint imposed by having both feet in fixed poses.

The CTB is secured in place by a five centimeter nylon band and push-button mechanism akin to an automotive seat-belt. The robot must first unclasp the restraint and then withdraw the CTB from the rack. After unclasping, R2 must plan to grab the forward facing strap on the CTB, shown in Figure 4(b1) and (b2). The plans are done by using an extension to [14], which defines goal poses relative to a frame in the world. In this case, the desired series of motions is specified in reference to the virtual CTB object shown in Figure 4(b2). After grabbing the strap and pulling the CTB out, the other hand grasps another strap, as shown in Figure 4(c1) and (c2). After this point, R2 must bimanually manipulate the CTB, introducing the problematic constraint first discussed in Section II. Finally, R2 moves away from the logistics module rack mockup to the starting position, this time with the CTB in tow, as shown in Figure 4d.

B. *Simulated Microgravity Climbing*

Use of ARGOS [23] is essential for R2 to operate in gravity, simulating the conditions of microgravity in space on-board the ISS. ARGOS is coupled to R2 with a rigid gimbal, yielding the requirement for the center of mass of R2 to continuously maintain an almost-constant orientation about the X- and Y-axes. A large movement about the X- and Y-axes will exert excessive force on R2 through the connection to ARGOS, causing a safety fault on the robot to prevent further damage [24]. The requirement is satisfied by creating a constraint on the torso with an orientation volume with tight bounds about the X- and Y-axes, but allowing free rotation about the Z-axis. Before the improvements to the planner and specification to allow relative constraints, the torso was orientation locked to the global X- and Y-axes. This was brittle as slight orientation changes due to the gimbal and ARGOS would cause invalid poses, and planning could not be done unless manual calibration of the current offset was done, a humongous time-sink for operators. The infrastructure of this system allows operations in ARGOS without manual specification of the constraint imposed. Moving away from ARGOS into an actual microgravity environment, the constraint can now simply be removed and everything will operate as expected. This greatly improves usability as constraints can be phrased in the unified specification of the system.

C. *Operator Usage*

The interactivity of the system is key to the improved ability of the operator to command the robot. Global and relative constraints definition in interface, interactive positioning of the goal poses, and automatic updating of constraints, poses, and workspace based on state all require less effort by the operator to define their mission. This process is opposed to the manual steps that had to be taken without the interface. Each subtask in an operation was individually crafted in order to account its specific constraints or requirements, and movements in ARGOS required careful hacks to avoid faulting R2. Each constraint was crafted offline by using a script to lookup frames, and required knowledge *a priori* of the relative poses of the frames if a relative constraint was made. Goal poses would be specified without knowledge of the robot state, and feasibility is undetermined until the planning problem fails. Using the current system and what it affords, the CTB scenario can be completed on the order of five minutes. The subtasks in the operation can be specified using constraints, and utilization of the interface provides an interactive, quicker means to develop valid plans rather than scripting in the dark. An operation that would have required hours of near-manual command by an operator can now be accomplished in minutes.

VI. Conclusion and Future Work

A system that allows human operators to efficiently specify complex operations for Robonaut 2, a dexterous humanoid robot, was presented. The system's interface enables the operator to interactively define constraints that automatically update with the state of the robot, preview motion plans that satisfy the constraints, and monitor the execution on the actual hardware. When planning for subtasks in an operation, the system ranks and sorts the constraints for an effective projection routine that enables efficient motion planning. This system is a first step towards the creation of co-robotic assistants in space, helping astronauts on the International Space Station and on future exploration missions.

There are several directions in which the existing work will be expanded. In the near future the capability of virtual objects in the interface will be increased through affordance templates [14]. This will simplify operations that require manipulation of objects. A longer-term goal is to increase the autonomy of R2. Planning long operations for R2 is currently tasked to the human operator, who plans the sequence of subtasks necessary to complete an operation. In the future, techniques from task planning could be employed to automatically solve problems of discrete reasoning, such as the path of handrails to walk on to reach a desired pose.

References

[1] M. A. Diftler, J. S. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. A. Permenter, *et al.*, "Robonaut 2—The first humanoid robot in space," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 2178–2183.

[2] L. B. Bridgwater, C. A. Ihrke, M. A. Diftler, M. E. Abdallah, N. A. Radford, J. M. Rogers, S. Yayathi, R. S. Askew, and D. M. Linn, "The Robonaut 2 hand—Designed to do work with tools," in *IEEE Intl. Conf. on Robotics and Automation*, 2012, pp. 3425–3430.

[3] M. A. Diftler, T. D. Ahlstrom, R. O. Ambrose, N. A. Radford, C. A. Joyce, N. De La Pena, A. H. Parsons, and A. L. Noblitt, "Robonaut 2—Initial activities on-board the ISS," in *IEEE Aerospace Conference*, 2012, pp. 1–12.

[4] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, June 2005.

[5] J. James, Y. Weng, S. Hart, P. Beeson, and R. Burridge, "Prophetic goal-space planning for human-in-the-loop mobile manipulation," in *IEEE-RAS Intl. Conf. on Humanoid Robots*, 2015, pp. 1185–1192.

[6] C. G. Atkeson, B. P. W. Babu, N. Banerjee, D. Berenson, C. P. Bove, X. Cui, M. DeDonato, R. Du, S. Feng, P. Franklin, *et al.*, "No falls, no resets: Reliable humanoid behavior in the DARPA robotics challenge," in *IEEE-RAS 15th Intl. Conf. on Humanoid Robots*, 2015, pp. 623–630.

[7] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, *et al.*, "Team IHMC's lessons learned from the DARPA robotics challenge trials," *Journal of Field Robotics*, vol. 32, no. 2, pp. 192–208, 2015.

[8] P. Hebert, M. Bajracharya, J. Ma, N. Hudson, A. Aydemir, J. Reid, C. Bergh, J. Borders, M. Frost, M. Hagman, *et al.*, "Mobile manipulation and mobility as manipulation—Design and algorithms of RoboSimian," *Journal of Field Robotics*, vol. 32, no. 2, pp. 255–274, 2015.

[9] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. P. D'Arpino, R. Deits, M. DiCicco, D. Fourie, *et al.*, "Affordance-based perception and whole-body planning in the DARPA robotics challenge," MIT, Cambrige, MA, Tech. Rep. MIT-CSAIL-TR-2014-003, 2014.

[10] H. A. Yanco, J. L. Drury, and J. Scholtz, "Beyond usability evaluation: analysis of human-robot interaction at a major robotics competition," *Human–Computer Interaction*, vol. 19, no. 1-2, pp. 117–149, 2004.

[11] H. A. Yanco and J. L. Drury, "Rescuing interfaces: A multi-year study of human-robot interaction at the AAAI robot rescue competition," *Autonomous Robots*, vol. 22, no. 4, pp. 333–352, 2007.

[12] G. Dorais, R. P. Bonasso, D. Kortenkamp, B. Pell, and D. Schreckenghost, "Adjustable autonomy for human-centered autonomous systems," in *Working notes of the Sixteenth Intl. Joint Conf. on Artificial Intelligence Workshop on Adjustable Autonomy Systems*, 1999, pp. 16–35.

[13] I. A. Şucan and S. Chitta. Moveit! [Online]. Available: http://moveit.ros.org

[14] S. Hart, P. Dinh, and K. Hambuchen, "The affordance template ROS package for robot task programming," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 6227–6234.

[15] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005.

[16] Y. Nakamura and H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *Journal of dynamic systems, measurement, and control*, vol. 108, no. 3, pp. 163–171, 1986.

[17] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[18] J. Badger, A. Hulse, R. Taylor, A. Curtis, D. Gooding, and A. Thackston, "Model-based robotic dynamic motion control for the Robonaut 2 humanoid robot," in *IEEE-RAS Intl. Conf. on Humanoid Robots (Humanoids)*, 2013, pp. 62–67.

[19] M. Stilman, "Task constrained motion planning in robot joint space," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2007, pp. 3074–3081.

[20] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *Intl. J. of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, October 2011.

[21] M. Quigley, K. Conley, B. P. 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*, 2009.

[22] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[23] P. Valle, L. Dungan, T. Cunningham, A. Lieberman, and D. Poncia, "Active response gravity offload system," NASA-Johnson Space Center, Houston, TX, NASA Tech Brief MSC-24815-1/24-1, Sep 2011.

[24] J. Badger, A. Hulse, and A. Thackston, "Advancing safe human-robot interactions with Robonaut 2," in *International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2014.