Humanoid Mobile Manipulation Using Controller Refinement

Robert Platt, Robert Burridge, Myron Diftler,
Jodi Graf, Mike Goza, Eric Huber
Dexterous Robotics Laboratory
Johnson Space Center, NASA
{robert.platt-1,robert.r.burridge,myron.a.diftler,
jodi.s.graf,s.m.goza}@nasa.gov,eric@roboteyes.com

Oliver Brock
Robotics and Biology Laboratory
Department of Computer Science
University of Massachusetts, Amherst
oli@cs.umass.edu

***Abstract*— An important class of mobile manipulation problems are "move-to-grasp" problems where a mobile robot must navigate to and pick up an object. One of the distinguishing features of this class of tasks is its coarse-to-fine structure. Near the beginning of the task, the robot can only sense the target object coarsely or indirectly and make gross motion toward the object. However, after the robot has located and approached the object, the robot must finely control its grasping contacts using precise visual and haptic feedback. This paper proposes that move-to-grasp problems are naturally solved by a sequence of controllers that iteratively refines what ultimately becomes the final solution. This paper introduces the notion of a *refining sequence* of controllers and defines it in terms of controller goal regions and domains of attraction. Refining sequences are shown to be more robust than other types of controller sequences. In addition, a procedure for converting a refining sequence into an equivalent "parallelized" controller is proposed. Executing this parallelized controller confers all the advantages of iteratively executing the controllers sequentially. The approach is demonstrated in a move-to-grasp task where Robonaut, the NASA/JSC dexterous humanoid, is mounted on a mobile base and navigates to and picks up a geological sample box.**

I. Introduction

A key reason why mobile humanoids are important is their ability to survive harsh environments, and because they can perform physically challenging tasks that require dexterity such as habitat and outpost construction. Indeed, NASA expects robots to be essential to future manned missions to the moon and Mars. By functioning as assistants to astronauts, robots are expected to increase the effectiveness of human extra-vehicular activities (EVAs). In addition, the possibility exists that robots could set up outposts for astronauts before they arrive as well as continuing to function after the crew return to Earth. Humanoid robots are particularly well suited to assist in manned missions because they are physically capable of performing many tasks that astronauts currently perform [1]. However, it is still not clear how to control these robots so that they are able to perform complex mobile manipulation tasks autonomously.

In the literature, mobile manipulation is frequently equated with solving force and/or motion control tasks with one or more mobile manipulators. Important previous work includes work by Chang *et. al.* and Khatib on the augmented object model and Williams and Khatib on the virtual linkage model for controlling object dynamics in operational space and modeling internal forces, respectively [2], [3], [4]. These models were effectively used to program hybrid force-position control tasks that used a mobile manipulator to erase a whiteboard, carry a basket, and sweep off a desk. Tan *et al.* demonstrated an approach to kinematic optimization and hybrid position and force control in the context of a cart pushing task using a mobile manipulator attached to a non-holonomic mobile base [5]. Several researchers have proposed ways of extending or applying behavior-based techniques to mobile manipulators. MacKenzie and Arkin adapted a behavior-based approach to a drum sampling task where a mobile robot needed to locate and approach a barrel and insert a probe into its bung hole [6]. This task was accomplished by executing a sequence of behaviors including *detect_drum*, *move_to_goal*, *detect_bung_hole*, *take_sample*, *transfer_sample*, *etc.* Petersson and Christensen divided the mobile manipulation problem into a mobility portion and a manipulation portion [7]. They proposed that the mobility part is best solved using behavior-based approaches while the manipulation part should be solved using a hybrid dynamical system. Pimentel *et al.* proposed a behavior-based architecture that can be applied to a cooperative carrying task [8].

Instead of addressing mobile manipulation in general, this paper focuses on a class of problems called "move-to-grasp" problems where a mobile humanoid robot must navigate to and pick up a target object. The general notion of "controller funneling" applies to this class of tasks [9]. In controller funneling, the robot executes a sequence of controllers such that the goal configuration of one controller must be contained inside the domain of attraction of the next. Effectively, these controllers "funnel" the state of the robot toward a goal configuration. A major advantage of this approach is that it is unnecessary to design a single, monolithic controller that converges to the task goal and yet has a large enough domain of attraction. Burridge, Rizzi, and Koditschek demonstrated that controller funneling can be an effective approach to dynamic robot juggling tasks [9]. Controller funneling has also been used in grasp synthesis where two grasp controllers execute sequentially to generate an enveloping grasp [10].

Huber and Grupen showed that it is possible to autonomously learn a sequence of controllers that funnels the state of a robot system toward specific goal configurations [11].

This paper focuses on a special case of controller funneling called *controller refinement*. A refining sequence of controllers must satisfy the conditions for controller funneling: the goal region of every controller must be inside the domain of attraction of the next controller in the sequence. In addition, controller refinement requires the goal region of each controller in the sequence to be contained within the goal region of all previous controllers. Solutions with this structure are particularly robust because the robot configuration never leaves the domain of attraction of all previously executed controllers. If a particular controller does cause the robot configuration to leave the goal region of a previous controller, then it can be halted and the robot can be returned to the previous goal region by executing the previous controller. Furthermore, a simple procedure exists for "parallelizing" refining sequences. Instead of executing all controllers in sequence, the same outcome can be achieved by a single composite controller that executes all controllers in parallel. For the "parallelized" controller, the precedence relationship among controllers that was implicit in the temporal order of the sequence is enforced by projecting controllers later in the temporal sequence into the null space of controllers earlier in the sequence.

This approach is characterized as part of a field study involving Robonaut, the NASA space humanoid, and SCOUT, a semi-autonomous rover that can transport two astronauts. In the part of the field study reported on in this paper, astronauts have placed a geological sample box on SCOUT. Robonaut, mounted on a mobile Segway^{TM}Robotic Mobile Platform (RMP) base, navigates to a region around SCOUT, approaches the sample box, and grasps and lifts the box. In Section II, refining sequences of controllers are defined and characterized. Next, Section III-A proposes navigation and hybrid position-force controllers that can be used to solve the move-to-grasp task. Finally, Section III-B experimentally characterizes the approach in the context of the Robonaut-SCOUT field test. Results are presented that show that, for the move-to-grasp task, executing controllers in a refining sequence leads to monotonically decreasing variance in position error relative to the object while localization accuracy correspondingly goes up. In addition, trajectories produced by the refining sequence are compared to trajectories generated by the corresponding parallelized controller.

II. Controller Sequencing

A. Controller Funneling

Refining sequences of controllers are a special case of funneling sequences. In controller funneling, pairs of controllers that execute sequentially must satisfy the *prepares* condition. Φ_i is said to *prepare* Φ_{i+1} when the goal region of Φ_i is inside the domain of Φ_{i+1},

$$g(\Phi_i) \subseteq \mathcal{D}(\Phi_{i+1}). \tag{1}$$

This condition is illustrated in Figure 1(a). The horizontal axis represents the robot configuration space and the vertical axis represents time. The two funnels represent controllers that lead the robot configuration to a goal region at the funnel "spout." Φ_i and Φ_{i+1} are a funneling sequence because they satisfy Equation 1, *i.e.* the goal region of Φ_i is inside the domain of attraction of Φ_{i+1}. This condition guarantees that each controller in the sequence delivers the robot to a configuration within the domain of attraction of the controller that executes next. A discrete control system that uses stable controllers and obeys this constraint is guaranteed to maintain control of the robot [9], [11]. One way to build such a discrete control system is to calculate the acyclic graph over controllers defined by the *prepares* relation. This graph describes all sequences of controllers that satisfy the constraint. Breadth-first-search may be used to search this graph for a sequence of controllers that leads to the goal configuration.

In addition to guaranteeing stability of the discrete system, funneling sequences of controllers also provide the system designer with an easy way to solve complex tasks using different types of feedback at different stages in the task. As will be demonstrated in the experimental section of this paper, move-to-grasp problems can be solved by using visual and haptic feedback at different stages in the task. While it is possible to design a single, monolithic controller that uses both types of feedback at different times, it is simpler to design two separate controllers that execute separately.

The *prepares* condition can also be enforced in the context of a state-based discrete control process. This approach requires discrete states to be defined over the robot configuration space. By executing controllers, the system can transition between states. A policy that associates each state with an action can be used to specify the behavior of the discrete control system. A common framework for representing the choices that a discrete control system has available is the Markov Decision Process (MDP). Because the MDP specifies a stochastic transition function, this framework can be used to characterize the stochastic dynamics of the discrete system. When a discrete control problem is framed as an MDP, standard planning and machine learning techniques such as dynamic programming and Reinforcement Learning (RL) can be used to autonomously learn a control policy [11], [12]. When an MDP representation is used, safety constraints such as the *prepares* condition can be enforced simply by pruning actions from the MDP as a function of state [11], [12]. When all "unsafe" actions are eliminated from the MDP, trial-and-error learning algorithms such as RL can be used to explore the space safely.

B. Controller Refinement

In addition to satisfying Equation 1 (the prepares condition), a *refining sequence* requires the goal region of each controller to be a subset of the goal regions of all previous controllers. This composite condition is,

$$g(\Phi_{i+1}) \subseteq g(\Phi_i) \subseteq \mathcal{D}(\Phi_{i+1}), \tag{2}$$

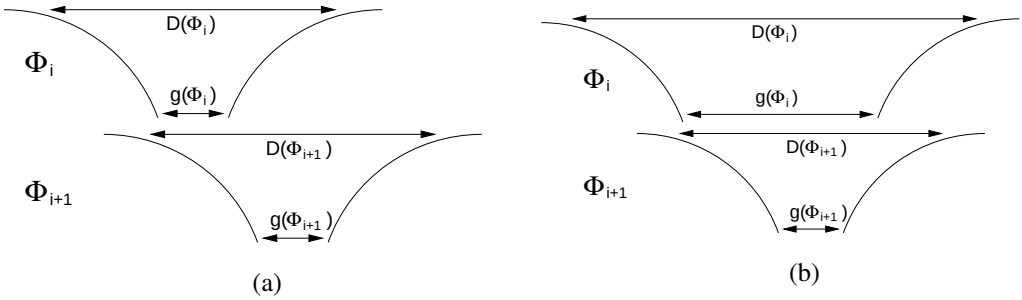


Fig. 1. Controller funneling, (a), compared with controller refinement, (b).

and is illustrated in Figure 1(b). This figure illustrates a refining sequence because the "spout" of Φ_{i+1} is inside the "spout" of Φ_i. Compared with funneling sequences, refining control sequences are important because they can provide an additional measure of robustness by monitoring the robot configuration and executing a previous controller if the currently executing controller fails or becomes unstable. Assume that the discrete control system continuously monitors which controller convergence predicates are satisfied - *i.e.* for all Φ_i, whether the robot configuration is inside $g(\Phi_i)$. Suppose that controller Φ_{i+1} is executing within the goal region of Φ_i. If Φ_{i+1} causes the configuration to depart from $g(\Phi_i)$, then a discrete control system can halt Φ_{i+1} and execute Φ_i until the robot configuration again returns to $g(\Phi_i)$. Provided that this discrete control system is able to monitor system configuration, the above approach allows the discrete control system to make its own stability guarantees even in the presence of potentially unstable controllers. The condition expressed in Equation 2 makes this technique possible. The $g(\Phi_{i+1}) \subseteq g(\Phi_i)$ part of Equation 2 gives the discrete system a way to evaluate whether to terminate Φ_{i+1}. The prepares component of the condition, $g(\Phi_i) \subseteq \mathcal{D}(\Phi_{i+1})$, ensures that the system is able to recover by executing Φ_i.

An interesting characteristic of refining control sequences is that the robustness of the discrete control system described above can be replicated by a single "parallelized" controller that executes all of the controllers in the refining sequence concurrently. Assume that each controller, Φ_i, in a refining sequence descends the gradient of its associated artificial potential function, $\nabla\Phi_i$. The parallelized controller projects the output of controllers late in the sequence into the null space of the output of "earlier" controllers. In particular, given a refining sequence, $R = (\Phi_1, \Phi_2, \ldots, \Phi_k)$, that executes Φ_1 first, Φ_2 second, *etc.*, the corresponding parallelized controller executes the following,

$$\Phi_{12\ldots k} = \Phi_k \triangleleft \Phi_{k-1} \triangleleft \ldots \Phi_1, \tag{3}$$

where $\triangleleft$ is the *subject-to* operator. The subject-to notation derives from the *control basis* framework and denotes that the controller directly to the left of $\triangleleft$ executes in the null space of all controllers to its right [13], [14]. Let

$$\mathcal{N}\begin{pmatrix} \nabla\Phi_1^T \\ \vdots \\ \nabla\Phi_{k-1}^T \end{pmatrix}$$

be a square matrix that projects arbitrary vectors into the null space of $\nabla\Phi_1 \ldots \nabla\phi_{k-1}$. Then the output of $\Phi_{12\ldots k}$ is

$$\nabla\Phi_{12\ldots k} = \nabla\Phi_1 + \mathcal{N}(\nabla\Phi_1)\nabla\Phi_2 + \ldots + \mathcal{N}\begin{pmatrix} \nabla\Phi_1^T \\ \vdots \\ \nabla\Phi_{k-1}^T \end{pmatrix}\nabla\Phi_k. \tag{4}$$

This sum projects the gradient of lower priority controllers into the null space of higher priority controllers. After projecting lower priority controller gradients into the null space, this composite controller executes the resulting sum. It is assumed that when the system is within the goal region of a controller, its gradient is zero and it does not change the associated null space projection matrix. For example, if the robot is inside the goal region of controllers Φ_1 through Φ_i, then the null space projection matrix, $\mathcal{N}\left((\nabla\Phi_1, \ldots, \nabla\Phi_i)^T\right)$ is the identity matrix.

Because the output of controllers late in the refining sequence are projected into the null space of the output of "earlier" controllers, the "later" controllers are constrained not to ascend the artificial potential function of "earlier" controllers. Essentially, this prevents "later" controllers from interfering with the objectives of the "earlier" controllers. This process of parallelization converts the temporal nature of the refining sequence into a prioritization of concurrently executing controllers. Note that the net effect of the parallelized controller is the same as that of the hybrid discrete-continuous controller described earlier. The discrete control system described earlier essentially constrains each controller in the sequence to execute within the goal region of all previous controllers. If the robot configuration leaves this region, then the current controller is halted and a previous controller executes so as to return the system to the goal region. Similarly, the parallelized version requires each controller to execute in the null space of all higher priority controllers. Suppose that the robot configuration is within the goal region of the controllers Φ_{i-1} through Φ_1. Then the null space projection matrix applied to the output of controller Φ_i is

State	Condition	Action
1	$r \geq R$ and $\beta \geq B$	$\Phi_{rot}(\beta)$
2	$r \geq R$ and $\beta < B$	$\Phi_{lin}(r)$
3	$r < R$ and $\alpha \geq A$	$\Phi_{rot}(\alpha)$

TABLE I

TURN-DRIVE-TURN CONTROL POLICY.

identity and does not effect its output. However, if Φ_i should push the system outside of this goal region, then the nullspace applied to Φ_i restricts Φ_i to the manifold of configurations tangent to Φ_{i-1} and allows Φ_{i-1} to return the system back into $g(\Phi_{i-1})$.

III. EXPERIMENTS

Controller refinement was explored in the context of the Robonaut-SCOUT field study. This section first describes the set of controllers that were used to solve this task. These include a navigation control policy that moved Robonaut's mobile base into range of the box and hybrid force-position controllers that moved Robonaut's hands into a grasp configuration. Next, experiments are described that execute sequential and parallelized versions of a refining sequence. Results are presented that characterize the trajectories taken by the robot in these two scenarios. As controllers in the refining sequence execute, the robot is shown to be confined to iteratively smaller and smaller regions of configuration space. Simultaneously, box localization error is shown to become smaller and smaller as the sequence proceeds.

A. Controllers

1) Turn-Drive-Turn Control Policy: After reaching a region around the target object, Robonaut must navigate to a goal pose directly in front of the object. A simple turn-drive-turn control policy is used that ignores the presence of obstacles. Given a goal pose, this control policy turns in the direction of the goal, moves in an approximate straight line toward the goal, and after reaching the goal position, turns into the goal orientation. Note that this approach can only be used with robots capable of point-turns.

The "turn-drive-turn" strategy is a policy implemented over the state variables,

$$\begin{aligned} r &= \| \mathbf{x}_{ref} - \mathbf{x} \|, \\ \beta &= atan\left(\frac{y_{ref} - y}{x_{ref} - x}\right), \\ \alpha &= \theta_{ref} - \theta, \end{aligned} \tag{5}$$

where $(\mathbf{x}, \theta)$ is the current RMP pose and $(\mathbf{x}_{ref}, \theta_{ref})$ is a reference pose. The state variables are as follows: r is the distance between the current position and the reference position, β is the heading of the object from Robonaut, and α is the difference between the Robonaut orientation and the object orientation.

Turn-drive-turn is the control policy illustrated in Table I. It is defined over three discrete states and executes one of two PD controllers as a function of state: $\Phi_{rot}(\theta_{ref})$ and $\Phi_{for}(d_{ref})$. $\Phi_{rot}(\theta_{ref})$ rotates Robonaut to a reference orientation, θ_{ref}. $\Phi_{lin}(d_{ref})$ moves the robot forward by a distance, d_{ref}. If Robonaut is in state 1 (it is more than R distance from the object and is not pointing toward the reference position), then turn-drive-turn executes a turn toward the reference using $\Phi_{rot}(\beta)$. If Robonaut is in state 2 (it is pointing toward the reference, but more than R away), then it drives to to the reference position using $\Phi_{for}(r)$. Finally, when Robonaut is in state 3 (it is in the reference position, but not the reference orientation), it executes a final turn, $\Phi_{rot}(\alpha)$, toward the reference orientation.

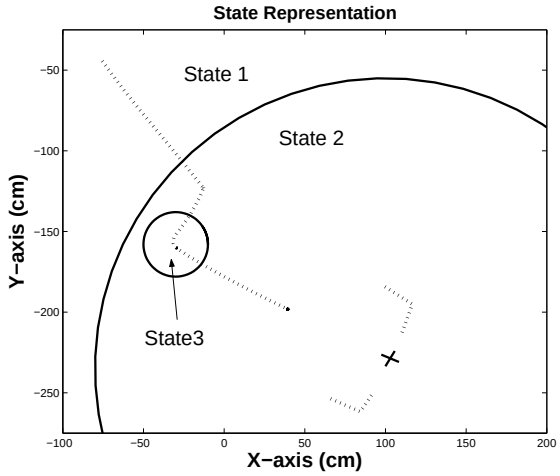


Fig. 2. States used by the approach control policy.

2) The Approach Control Policy: Instead of executing a single turn-drive-turn controller that moves directly to the target object from a point 2.5m away, the approach control policy, $\Phi_{approach}$, is implemented that traverses this distance in three distinct turn-drive-turns. A policy is defined over a discrete state space that essentially navigates the robot through a sequence of pose via-points. The discrete state space is a partition of the space of real-valued robot-object poses. The fixed policy associates each discrete state with an action that is implemented by a control process.

This implementation uses a fixed policy defined over the three position-based states identified in Figure 2. The x- and y-axes represent positions in centimeters. The cross near the lower right corner represents the position and orientation of the target object. The solid circle and the arc represent the boundaries between the three states. The dotted lines represent a sample trajectory taken by the RMP base and Robonaut's two hands using this implementation. State 1 corresponds to the set of positions at least 1.8m from the target object. State 2 corresponds to the set of positions less than 1.8m. State 3 corresponds to a small radius around the set of poses (position and orientation) 1.5m directly in front of the object.

The approach object policy is given in Table II. When Robonaut is more than 1.8m away from the target object (state 1), then it drives directly toward the object to a point 1.5m

State	Controller	Position reference
State 1	Φ_{drive}	(the point between robot and object, 1.5m away from object)
State 2	Φ_{drive}	(the point 1.5m directly in front of object)
State 3	Φ_{drive}	(the point 0.6m directly in front of object) the object)

TABLE II

APPROACH CONTROL POLICY.

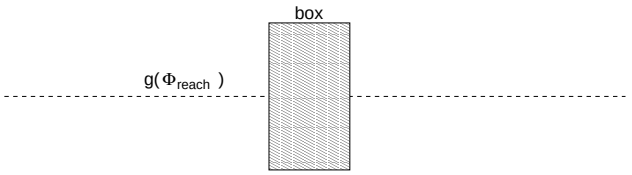


Fig. 3. The dashed line represents the goal region of the reach controller, Φ_{reach}.

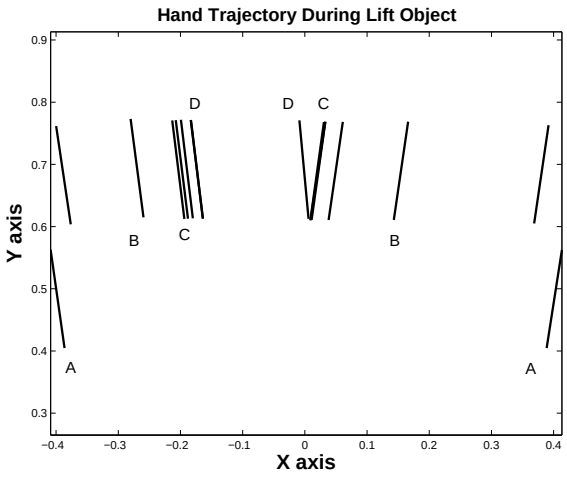


Fig. 4. An example of the trajectory taken by Robonaut's two palms as they grasp the sample box. After starting at configuration "A," the two palms reach toward the object toward configuration "B." Next, the robot executes guarded move and a compliance controllers that move the palms to configurations "C" and "D," respectively.

away. This should cause a transition to state 2. Once Robonaut is less than 1.8m away, it drives to a point 1.5m directly in front of the object, which likely causes a transition to state 3. Finally, when Robonaut is in state 3, it it drives to a point directly in front of the object.

3) Hybrid Force-Position Controllers: This paper takes a control-based approach to grasping whereby grasps are synthesized by executing closed-loop controllers that use position and force feedback. The grasping task is decomposed into a set of three hybrid force-position control objectives. The first objective is to move Robonaut's hands into a neighborhood of the desired contact positions based on the visually determined box pose. Next, a guarded move is executed that puts both palms in contact with the sides of the box. Finally, a moment controller executes that complies the two palms flat against the sides of the box.

The first controller, Φ_{reach}, moves Robonaut's two hands onto a line perpendicular to the sides of the box that intersects the box center as illustrated in Figure 3. This controller moves control points located near the middle of Robonaut's palms onto this line.

The second controller, Φ_{gm}, executes a guarded move that places both palms in contact with the object. The guarded move is implemented by two constituent controllers that execute concurrently using the subject-to operator of Section II-B. The first constituent controller, Φ_f, is a force controller that applies with Robonaut's palms reference forces directly inward towards the box. This constituent controller is concurrently combined with Φ_{reach} to yield

$$\Phi_{gm} = \Phi_f \triangleleft \Phi_{reach}. \tag{6}$$

In this expression, Φ_{reach} constrains the palms to move along the dashed line in Figure 3. In the null space of this objective, Φ_f applies the reference force.

The third controller, Φ_{comply}, complies the palm flat against the sides of the sample box. This controller executes a moment controller concurrently with the force controller and the reach controller described above. The moment controller, Φ_m, operates with respect to control points located in the middle of Robonaut's palms. Φ_m moves the hands so as to minimize the net moment about these control points. The resulting composite controller is

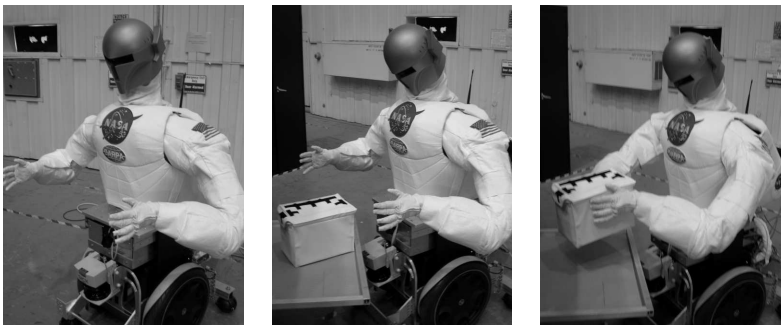
$$\Phi_{comply} = \Phi_m \triangleleft \Phi_f \triangleleft \Phi_{reach}. \tag{7}$$

As before, Φ_{reach} constrains the palms to move along the lines illustrated in Figure 3. Φ_f presses the palms against the sides of the box. Finally, Φ_m rotates the palm so as to minimize moments about the control point - essentially rotating the palm so that all surfaces come into contact. Φ_f literally "pushes" the palms flat against the sides of the box.

Figure 4 shows an example of the trajectory taken by Robonaut's two palms when it executes these controllers in sequence. The lines in the figure illustrate Robonaut's two palms from an overhead perspective as paddles. Each hand is represented by a line drawn between the heel of the palm and the fingertips. The example starts when the palms are located at the two positions labeled (A) in Figure 4. Executing Φ_{reach} moves the palms onto the line approximately perpendicular to the sides of the box (positions (B) in the figure.) Note that due to visual localization error, the line is not exactly perpendicular. Next, Robonaut executes a guarded move, $\Phi_f \triangleleft \Phi_{reach}$, toward the box. This moves the palms to the positions labeled (C) in Figure 4. At this point in the example, the heels of the two palms are touching the sides of the box. Next, executing the comply controller, $\Phi_m \triangleleft \Phi_f \triangleleft \Phi_{reach}$, moves the palms to positions (D) in the figure. Now, each palm is pressed flat against the box.

B. Experimental Setup and Results

Experiments were conducted that characterize sequential and parallelized versions of a refining sequence of controllers.



(a) (b) (c)

Fig. 5. Illustration of Robonaut completing the move-to-grasp task in the Robonaut-SCOUT field study.

State	Condition	Controller
1	$\Phi_{approach}$	approach object
2	Φ_{reach}	reach toward object
3	Φ_{gm}	guarded move
4	Φ_{comply}	comply to object

TABLE III

THE REFINING CONTROL POLICY USED IN THE ROBONAUT-SCOUT FIELD STUDY.

1) Sequential Execution of a Refining Sequence: In the first experiment, a series of eight trials were conducted where Robonaut navigated to and picked up a geological sample box measuring 7in $\times$ 8in $\times$ 11in. Robonaut started each trial in a different location and orientation approximately 2.25m away from the box and executed the refining sequence illustrated in Table III. Starting no more than 2.5m away from the sample box, this sequence executes the approach controller, $\Phi_{approach}$, that moves the RMP to within a radius of 0.7m of the box (within reaching distance.) Next, it executes the reach controller, Φ_{reach}, that moves the two hands around the box. Next, a guarded move executes that makes contact with the sides of the box. After making contact, the refining sequence executes Φ_{comply} to comply to the sides of the box. The experimental scenario is illustrated in Figure 5. In Figure 5(a), Robonaut is 2.25m away from the box. In Figure 5(b), Robonaut has navigated to a point just in front of the box. In Figure 5(c), Robonaut is lifting the box.

Figure 6 illustrates the trajectories followed by the robot during these eight trials. In this figure, the sample box is at the origin with its major axis oriented horizontally. The lines on the left side of the plot illustrate the path of the center of the Robonaut RMP base. The two clusters of "L"-shaped lines on the right illustrate the paths of the left and right palms. The "lightning bolt" shape of the RMP trajectories is the result of the approach control policy, $\Phi_{approach}$. Since Robonaut is more than 1.5m away from the sample box, $\Phi_{approach}$ moves directly toward the box. When it gets to a point within 1.5m, a transition to state 2 occurs (in Table II) and Robonaut moves to a point along the axis of the box. When Robonaut

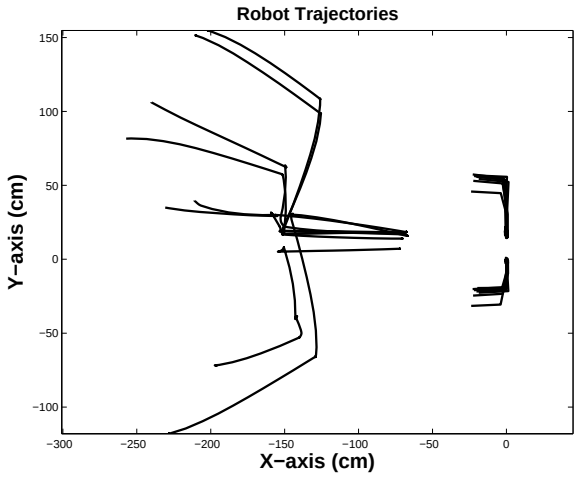
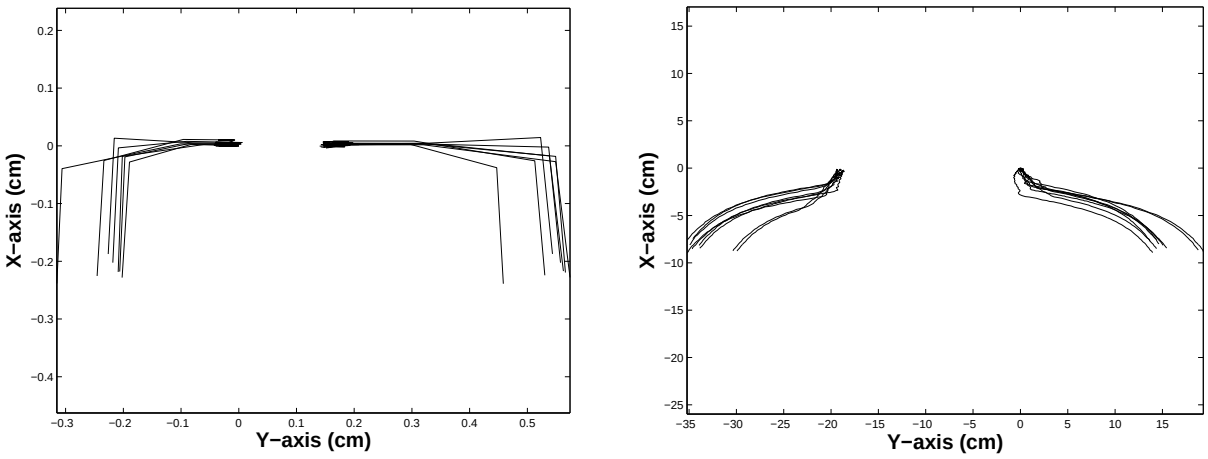


Fig. 6. The trajectories taken by Robonaut during the eight experimental trials. The "lightning-bolt" trajectories on the left side are the trajectories taken by the mobile base. The "L"-shaped trajectories on the right are the paths taken by Robonaut's two palms.

reaches a point 1.5m directly in front of the box, the system transitions to state 3 in the approach control policy and drives toward the box. After arriving in front of the box, the approach control policy terminates and the refining sequence of Table III takes over again and reaches the two palms toward the box. Following the reach, the palms make contact with the sides of box, comply with the box, and pick it up.

The eight trajectories shown in Figure 6 illustrate how Robonaut is confined to a smaller and smaller region of configuration space as it approaches the goal. Robonaut starts the experiment in a large range of positions, approximately 2.25m away from the object. However, the variance in Robonaut's position decreases significantly when it reaches a position directly in front of the sample box. Finally, after Robonaut makes contact and complies with the box, this variance virtually disappears.

Robonaut's progression through the refining sequence of controllers is mirrored by a continual decrease in the variance of the estimated pose of the sample box. This is illustrated



(a) (b)

Fig. 8. The trajectory taken by Robonaut's two palms when executing the sequential control sequence, (a), compared with the parallelized control sequence, (b).

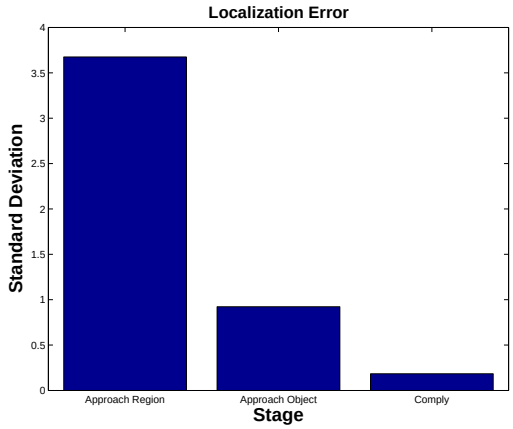


Fig. 7. Standard deviation in the estimated box position decreases as the refining control policy of Table III executes. The first bar, "approach region" gives standard deviation when Robonaut is approximately 2.25m away from the sample box. The second bar shows standard deviation after approaching the sample box. The third bar shows standard deviation after making contact and complying to the sides of the box.

in Figure 7. When Robonaut is 2.25m away from the box, the variance in the visually estimated position is large (the "approach region" bar in Figure 7). However, after approaching the box, Robonaut is able to localize the box much more precisely (the "approach object" bar). Finally, after contacting and complying with the object, Robonaut augments is visual sense with tactile information that estimate the object pose very precisely ("comply" bar).

2) Comparison to the Parallelized Controller: The second experiment explored a parallelized version of the refining sequence. This parallelized version omitted the approach controller, $\Phi_{approach}$, and just executed the hybrid force-position controllers, Φ_{reach}, Φ_{gm}, and Φ_{comply}. Instead of executing Φ_{reach}, Φ_{gm}, and Φ_{comply} sequentially as in Table III, the corresponding parallelized controller can be written following Equation 3,

$$\begin{aligned}
\Phi_{parallelized} &= \Phi_{comply} \triangleleft \Phi_{gm} \triangleleft \Phi_{reach} && (8)\\
&= (\Phi_m \triangleleft \Phi_f \triangleleft \Phi_{reach}) && (9)\\
&\quad \triangleleft (\Phi_f \triangleleft \Phi_{reach}) \triangleleft \Phi_{reach}\\
&= \Phi_m \triangleleft \Phi_f \triangleleft \Phi_{reach} && (10)\\
&\quad \triangleleft \Phi_f \triangleleft \Phi_{reach} \triangleleft \Phi_{reach}\\
&= \Phi_m \triangleleft \Phi_f \triangleleft \Phi_{reach} && (11)
\end{aligned}$$

In this equation, $\Phi_{comply} \triangleleft \Phi_{gm} \triangleleft \Phi_{reach}$ is reduced to $\Phi_m \triangleleft \Phi_f \triangleleft \Phi_{reach}$ by substituting into Equation 8 using Equations 6 and 7. In Equation 10, the parentheses have been removed using an associativity property. Finally, in Equation 11, redundant controllers have been removed from the expression (only the right-most constituent controller is kept).

The results of executing this parallelized controller was compared to the results of executing the controllers sequentially. Figure 8(b) shows the results of executing the parallelized controller nine times toward the visually-located box. The s-shaped lines on the left and the right represent the trajectories of the left and right palms as they move toward the box in the middle. For comparison, Figure 8(a) repeats the palm trajectories shown in Figure 6 - those generated by executing Φ_{comply}, Φ_{gm}, and Φ_{reach} sequentially, as shown in Figure 6. In addition to yielding a smoother and more natural motion, in this case, the parallelized controller executes faster than the sequential motion because both Φ_{gm} is able to make progress toward its goal without violating the constraints of Φ_{reach} (*i.e.* without moving the palms away from the line of reach objectives.) Note that, instead of using the null space projection, the advantages of concurrent executions could also be realized simply by adding the outputs of Φ_{comply}, Φ_{gm}, and Φ_{reach}. However, this approach would not guarantee that Φ_{reach} would ever succeed. Note that in Figure 8(b), the palms are displaced a short distance approximately tangent to the

box surface after making contact. This displacement is caused by the Φ_{reach} objectives continuing to be asserted even after making contact. The reach controller forces the contacts to slide a small distance along the box surface to their final positions.

3) Discussion: One advantage of funneling control sequences (whether they are refining or not) is that they provide an easy way to utilize different kinds of information at different stages in a task. This is highlighted in Figure 7. The decrease in variance shown in the figure during execution of the refining sequence suggests that information sufficiently accurate to solve this task in a single step is simply not available at the beginning of the task. Before executing $\Phi_{approach}$, the sample box cannot be localized accurately enough to grasp it. This only becomes possible when 1) visual accuracy improves as Robonaut approaches the box, and 2) Robonaut is able to touch the box, thereby augmenting visual information with tactile information. Funneling sequences of controllers can take advantage of improvements in information accuracy as the task progresses because controllers that execute at different stages in the sequence can use different kinds of information. This was advantageous in the Robonaut-SCOUT field test implementation because, in the later stages of the task, tactile information could be used to move the contacts into a precise grasping configuration.

The additional robustness that a refining sequence of controllers provides comes at the expense of potential difficulties implicit in designing controllers that can be sequenced so as to satisfy Equation 2. Rather than designing a set of controllers with goal regions that satisfy the requirements of the refining sequence, it can be easier to design a sequence of funneling controllers that lead the robot through a series of via points on a path toward the goal. The design of these controllers can be easier because their domain of attraction does not need to cover all configurations where the sequence will ultimately lead the robot. If the approximate configuration of the robot at the start of a particular controller's execution is known, then the controller's domain of attraction need only cover that configuration. This simplification comes at the expense of the robustness that a refining sequence allows (described in Section II-B). If a controller assumes a particular starting configuration, it cannot necessarily be used to return the robot to a stable configuration if a subsequent controller fails.

IV. Summary

This paper has addressed a class of mobile manipulation problems called "move-to-grasp" problems, where a mobile manipulator must navigate to and pick up an object. It is proposed that move-to-grasp problems are best solved by a *refining sequence* of controllers, where each controller in the sequence iteratively confines the robot to a smaller and smaller region of configuration space. Refining sequences are particularly robust because the robot is always within the domain of attraction of all previously executed controllers in the sequence. In addition, a procedure is given for converting a refining sequence of controllers into a single "parallelized" controller that realizes the same results as the sequence by executing all controllers simultaneously. This approach is explored in a move-to-grasp task where Robonaut navigates to and picks up a geological sample box off of a platform in the rear of SCOUT. Results are given that show that over a series of trials, Robonaut's configuration is confined to an iteratively smaller region around the sample box. This narrowing in configuration space is mirrored by improvements in the precision of Robonaut's estimated position of the box.

References

[1] M. Diftler, J. Mehling, P. Strawser, W. Doggett, and M. Spain, "A space construction humanoid," in *Proceedings of the IEEE Int'l Conf. on Humanoid Robotics*, 2005.

[2] K. Chang, R. Holmberg, and O. Khatib, "The augmented object model: Cooperative manipulation and parallel mechanism dynamics," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2000.

[3] O. Khatib, "The augmented object and reduced effective inertia in robot systems," in *Proc. American Control Conference, Atlanta*, vol. 3, 1988, pp. 2140–2147.

[4] D. Williams and O. Khatib, "The virtual linkage: A model for internal forces in multi-grasp manipulation," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 3, 1993, pp. 1025–1030.

[5] J. Tan, N. Xi, and Y. Wang, "Integrated task planning and control for mobile manipulators," *Int'l Journal of Robotics Research*, vol. 22, no. 5, 2003.

[6] D. MacKenzie and R. Arkin, "Behavior-based mobile manipulation for drum sampling," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1996.

[7] L. Petersson and H. Christensen, "A framework for mobile manipulation," in *7th International Symposium on Robotics Systems*, 1999.

[8] B. Pimentel, G. Pereira, and M. Campos, "On the development of cooperative behavior-based mobile manipulators," in *Proc. of the Int'l Conf. on Autonomous Agents*, 2002.

[9] R. Burridge, A. Rizzi, and D. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *International Journal of Robotics Research*, vol. 18, no. 6, 1999.

[10] R. Platt, A. Fagg, and R. Grupen, "Extending fingertip grasping to whole body grasping," in *IEEE Int'l Conference on Robotics and Automation, Taipei, Taiwan*, May 2003.

[11] M. Huber and R. Grupen, "Robust finger gaits from closed-loop controllers," in *IEEE Int'l Conf. Robotics Automation*, 2002.

[12] R. Platt, A. H. Fagg, and R. A. Grupen, "Manipulation gaits: Sequences of grasp control tasks," in *IEEE Int'l Conf. Robotics Automation, New Orleans, Louisiana*, April 2004.

[13] M. Huber, "A hybrid architecture for adaptive robot control," Ph.D. dissertation, U. Massachusetts, 2000.

[14] R. Platt, "Learning and generalizing control-based grasping and manipulation skills," Ph.D. dissertation, University of Massachusetts, September 2006.