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 characterizes this type of solution. 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. In a series of tests, it is shown that a refining sequence of controllers decreases variance in robot configuration relative to the sample box until a successful grasp has been achieved.**

I. Introduction

One of the most common requirements of future mobile humanoid robots will be to locate, pick up, and retrieve objects. Indeed, NASA foresees this as one important way that space humanoids will be able to assist astronauts on future lunar and planetary missions. This paper focuses on a class of mobile manipulation problems called "move-to-grasp" problems where a mobile humanoid robot must navigate to and pick up a target object.

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. A mobile manipulator frequently consists of one or two arms mounted on a mobile base. Important previous work includes work in Khatib's lab regarding the augmented object model and virtual linkage model for controlling object dynamics in operational space and modeling internal forces, respectively [2]. 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 [3]. 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 [4]. 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 [5]. 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 [6].

Instead of addressing mobile manipulation in general, this paper specifically focuses on move-to-grasp problems where a mobile manipulator must locate, approach, and lift a desired object. The general notion of "controller funneling" applies to this class of tasks [7]. 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 [7]. Controller funneling has also been used in grasp synthesis where two grasp controllers execute sequentially to generate an enveloping grasp [8]. Huber and Grupen showed that it is possible to autonomously learn a sequence of controllers that funnel the state of a robot system toward specific goal configurations [9].

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. However, controller refinement also requires the domain of attraction for each subsequent controller in the sequence to be contained within the domain of all previous controllers. This structure implies that later controllers in the refining sequence will not cause the robot to leave the domain of attraction of earlier controllers. In addition, the state of the system will be iteratively confined to smaller and smaller regions of configuration space. While not all discrete control problems admit refining solutions, this paper proposes that move-to-grasp problems are naturally solved this way.

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. This paper presents results that show that the controller refinement approach leads to monotonically decreasing variance in position error relative to the object. It is shown that localization accuracy correspondingly goes up. Sections II and III propose navigation and hybrid position-force controllers that can be used to solve a move-to-grasp task. In Section IV, refining sequences of controllers are defined and characterized. Finally, Section V describes how these ideas apply to the Robonaut-SCOUT field test and present the results from experimental trials.

II. Navigation Controllers

This section describes the navigation controllers that are used in the solution to the move-to-grasp problem.

A. Approach Region Controller

The APPROACH REGION controller navigates the robot over uneven terrain while avoiding obstacles to within 2.5m of the object to be picked up. APPROACH REGION is a nested hierarchical controller, as illustrated in Figure 1. A high level controller iteratively computes obstacle-free paths to the goal at approximately 10Hz (the “plan curve” box in Figure 1). The low level controller follows this path by referencing PD controllers to via points along the last computed path (the “select via point” box in Figure 1). It is assumed that the goal region can be identified by looking for a large object known to be in the vicinity of the target object. In the Robonaut-SCOUT field test, the sample box is assumed to be located on SCOUT. Before moving, Robonaut visually localizes SCOUT, identifies local obstacles using a laser range finder, and plans an obstacle-free path to SCOUT. Robonaut’s motion is controlled by appropriately parameterizing PD controllers that servo to positions and angles along the path. *En route* to SCOUT, APPROACH REGION updates the positions of local obstacles using the laser range finder and re-evaluates a new obstacle-free path at approximately 10Hz.

Robonaut uses a SICK laser scanner and a three-axis inclinometer to detect obstacles. Since Robonaut may be moving through uneven terrain, the inclinometer is needed to project the range data into a uniform reference frame. Instead of developing a three-dimensional obstacle map, all obstacles are projected onto a planar occupancy grid approximately parallel to the ground. Each time the occupancy grid is populated, all prior knowledge of obstacles is discarded and the new occupancy grid is referenced to Robonaut’s most recent pose. This occupancy grid covers a fixed area around Robonaut; all space outside of the occupancy grid is assumed to be clear.

The occupancy grid is used to plan an obstacle-free path from Robonaut’s current position to a region around SCOUT. In order to accommodate the non-holonomic constraints of the RMP base, a smooth curved path is calculated using a non-uniform rational b-spline (NURB). The NURB is parameterized by a set of control points that are used to “pull” the path away from obstacles. In addition, the NURB can be parameterized with a fixed minimum radius that ensures that minimum turn rate constraints are met.

Instead of planning a path for a volume of Robonaut’s actual dimensions, the planning problem is simplified by “growing” the obstacles by half the width of Robonaut and planning the path of a “point robot.” The set of control points that “pull” the NURB path away from obstacles is calculated iteratively. As a first step, a NURB path connecting current to goal configurations is calculated without using any control points. This path is checked for collisions with the “grown” obstacles in the occupancy grid. If a collision is detected, a control point is inserted that causes the NURB to avoid the obstacle. This process continues until no more collisions are detected. Although this method of greedily placing control points for a NURB is not guaranteed to find a solution, this was found to be an efficient method of calculating an obstacle-free path when relatively few obstacles were encountered. This method is well matched to the “approach region” problem where it is unnecessary to reach a precise goal configuration.

After planning a path, the APPROACH REGION controller follows the path by updating references for position and orientation PD controllers. Robonaut’s position is projected onto the closest point on the NURB. PD controllers are referenced to a via point at some offset from the current position along the curve.

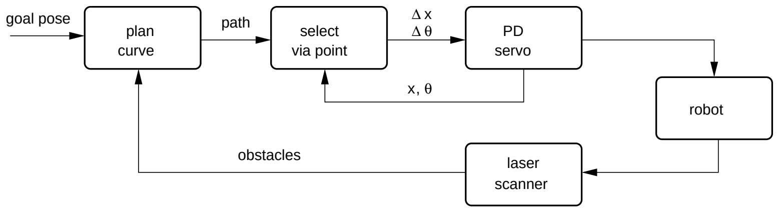

Fig. 1. APPROACH REGION controller

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

TABLE I
TURN-DRIVE-TURN CONTROL POLICY.

B. 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,

$$\begin{aligned} r &= \| \mathbf{x}_{ref} - \mathbf{x} \|, \\ \beta &= atan\left(\frac{y_{ref} - y}{x_{ref} - x}\right), \\ \alpha &= \theta_{ref} - \theta, \end{aligned} \tag{1}$$

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: $\pi_{rot}(\theta_{ref})$ and $\pi_{for}(d_{ref})$. $\pi_{rot}(\theta_{ref})$ rotates Robonaut to a reference orientation, θ_{ref}. $\pi_{lin}(d_{ref})$ 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 $\pi_{rot}(\beta)$. 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 $\pi_{for}(r)$. Finally, when Robonaut is in state 3 (it is in the reference position, but not the reference orientation), it executes a final turn, $\pi_{rot}(\alpha)$, toward the reference orientation.

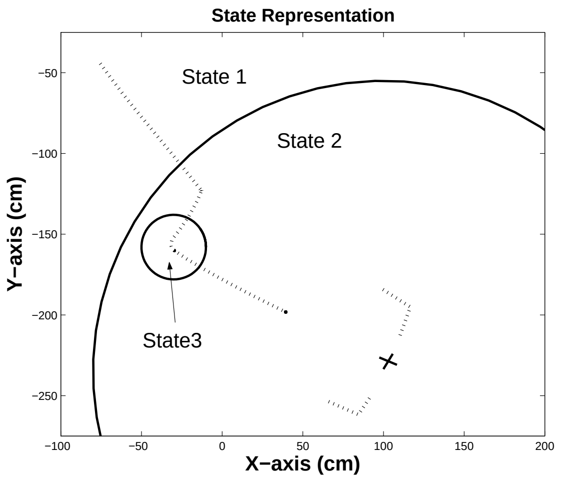

Fig. 2. States used by the APPROACH OBJECT control policy.

C. The Approach Object 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 OBJECT control policy 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 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

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

TABLE II
APPROACH OBJECT CONTROL POLICY.

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.

III. 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 sequence of hybrid force-position control objectives. First, Robonaut reaches both hands to visually determined reference configurations around the box. Next, a guarded move is executed that puts both palms in contact with the sides of the box. Next, a compliance controller executes that presses the two palms flat against the sides of the box. Finally, Robonaut lifts the box while maintaining a constant grasping force.

In order to create the right set of hybrid force and position controllers, a flexible framework for controller composition such as the *control basis* is used [10]. The control basis allows force and position control primitives to be parameterized by control points and references in a flexible way. It also allows force and position controllers to be concurrently combined, resulting in hybrid controllers. The control basis represents a position control primitive as follows, $\phi_p|_\tau^{\sigma_p(y,\mathbf{x}_{ref})}$. In this expression, ϕ_p is a position artificial potential function, $\sigma_p(y, \mathbf{x}_{ref})$ is the sensor transform that evaluates the position error between the set of y control points and a reference position $\mathbf{x}_{ref}$, and τ is the effector transform. For example, $\phi_p|_\tau^{\sigma_p(p_1,\mathbf{x}_{ref})}$ moves control point p_1 to reference position $\mathbf{x}_{ref}$. Similarly, the orientation control primitive, $\phi_r|_\tau^{\sigma_r(y,\theta_{ref})}$, moves the y control points to a reference orientation, θ_{ref}. The force control primitive is represented, $\phi_f|_\tau^{\sigma_f(y,\mathbf{f}_{ref})}$, where y is a set of control points (*i.e.* contacts) and $\mathbf{f}_{ref}$ is a reference force. For example, $\phi_f|_\tau^{\sigma_f(\{p_1,p_2\},\mathbf{f}_{ref})}$ applies the reference force, $\mathbf{f}_{ref}$ at control points, p_1 and p_2. In order to simplify the notation in this paper, the following abbreviations for these position and force controllers are defined. The notation, $\pi_p(y, \mathbf{x}_{ref})$, is used to represent the position control primitive, $\phi_p|_\tau^{\sigma_p(y,\mathbf{x}_{ref})}$. The notation, $\pi_r(y, \theta_{ref})$, is used to represent the orientation controller, $\phi_r|_\tau^{\sigma_r(y,\theta_{ref})}$. Finally, $\pi_f(y, \mathbf{f}_{ref})$, represents the force control primitive, $\phi_f|_\tau^{\sigma_f(y,\mathbf{f}_{ref})}$.

The control basis creates composite controllers by combining multiple control primitives. One controller, π_2, executes in the nullspace of the error function of another controller, π_1. In this case, both controllers execute concurrently, but the contribution of π_2 is constrained to not interfere with π_1. The control basis framework denotes this composite controller $\pi_2 \triangleleft \pi_1$. For example, consider executing the composite controller, $\pi_f(p_1, \mathbf{f}_{ref}) \triangleleft \pi_p(p_2, \mathbf{x}_{ref})$, where control point p_2 is already in position $\mathbf{x}_{ref}$. This controller attempts to apply a force of $\mathbf{f}_{ref}$ at control point p_1 while not moving control point p_2 from $\mathbf{x}_{ref}$.

Grasps are synthesized by executing a sequence of four controllers constructed using the control basis framework. The first controller,

$$\pi_{reach} = \pi_p(p_{palm}, \mathbf{x}_{object}) \triangleleft \pi_r(\theta_{palm}, \theta_{object}), \quad (2)$$

moves the center of the palm to a reference position and orientation. p_{palm} is a control point in the middle of the palm, $\mathbf{x}_{object}$ is a goal position near the target object, and θ_{object} is a goal orientation near the object. The two control primitives, $\pi_p(p_{palm}, \mathbf{x}_{object})$ and $\pi_r(\theta_{palm}, \theta_{object})$, move the center of the palm to a reference position and orientation, respectively. For a redundant arm with at least six degrees of freedom, both objectives can be simultaneously achieved. In this case, it is possible to execute the two control primitives in either order.

Another controller,

$$\pi_{gm} = \pi_p(p_{palm}, \mathbf{x}_{object}) \triangleleft \pi_f(p_{palm}, 0), \quad (3)$$

executes a guarded move that places both palms in contact with the object. This controller executes $\pi_p(p_{palm}, \mathbf{x}_{object})$ in the nullspace of $\pi_f(p_{palm}, 0)$. As before, p_{palm} is a control point in the middle of the palm. $\mathbf{x}_{object}$ is a goal position on (or inside) the target object. $\pi_p(p_{palm}, \mathbf{x}_{object})$ moves the center of the palm to a point on the object. $\pi_f(p_{palm}, 0)$ is a force control primitive that moves the palm away from applied forces. When no forces are applied to the palm, this controller moves the palms toward the object. However, the controller will not push into the object because the higher-priority force control primitive will prevent the manipulator from applying large forces to the object.

The next controller,

$$\pi_{comply} = \pi_f(\{f_1, f_2\}, 0) \triangleleft \pi_f(p_{palm}, \mathbf{f}_{int}) \triangleleft \pi_p(p_{palm}, \mathbf{x}_{line}), \quad (4)$$

complies the palm flat against the sides of the sample box. This controller executes $\pi_f(\{f_1, f_2\}, 0)$, $\pi_f(p_{palm}, \mathbf{f}_{int})$, and $\pi_p(p_{palm}, \mathbf{x}_{line})$ concurrently. The highest priority controller, $\pi_p(p_{palm}, \mathbf{x}_{line})$, is a position controller that keeps the palm in approximately the same position on the object surface. A point at the center of the palm is constrained only to move normal to the object surface, along $\mathbf{x}_{line}$. Without violating this position constraint, the second highest priority controller, $\pi_f(p_{palm}, \mathbf{f}_{int})$, applies a force of $\mathbf{f}_{int})$ at the palm toward the object. This controller causes the palm to press on the object. The lowest priority controller, $\pi_f(\{f_1, f_2\}, 0)$, allows the two control points, f_1 and f_2 at the fingertips and the heel of the palm to comply to the object surface. These two control points are on either side of the point in the middle of the palm. Since this controller is subordinate to the other two controllers, it cannot push the palm away from the surface.

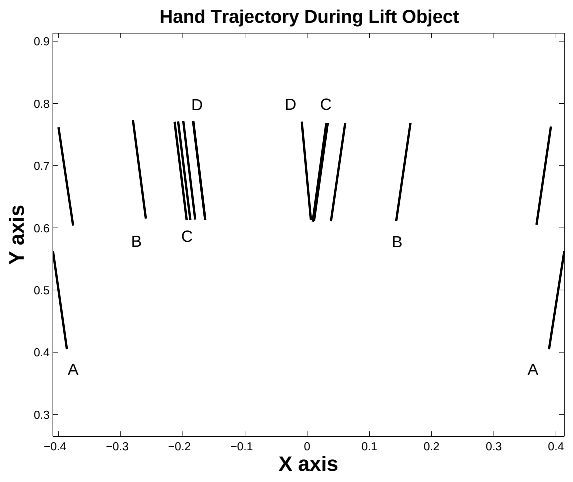

Fig. 3. An example of the trajectory taken by Robonaut's two palms as they grasp the sample box. After starting at configuration "A," the two palms reach toward the object toward configuration "B." Next, the robot executes guarded move and a compliance controllers that move the palms to configurations "C" and "D," respectively.

The point in the middle of the palm applies a force toward the object and "pushes" the fingers or the heel of the palm so that they comply to the object surface.

Last,

$$\pi_{lift} = \pi_p(\{p_l, p_r\}, \mathbf{x}_{goal}) \triangleleft \pi_f(\{p_l, p_r\}, \mathbf{f}_{int}), \qquad (5)$$

moves the two palms to a reference position while applying an inward holding force. The highest priority control primitive, $\pi_f(\{p_l, p_r\}, \{\mathbf{f}_l, \mathbf{f}_r\})$, applies an internal force between the two palms, p_1 and p_2, where $\mathbf{f}_l$ and $\mathbf{f}_r$ are inwardly directed reference forces. The subordinate control primitive, $\pi_p(\{p_l, p_r\}, \mathbf{x}_{goal})$, moves the two palms to $\mathbf{x}_{goal}$ while maintaining the internal force.

Figure 3 shows an example of the trajectory taken by Robonaut's two palms when it executes these controllers in sequence. The figure shows Robonaut's two hands from an overhead perspective. 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 3. Executing $\pi_{reach} = \pi_p(p_{palm}, \mathbf{x}_{object}) \triangleleft \pi_r(\theta_{palm}, \theta_{object})$ moves the palms to positions approximately 10cm away from the sides of the box (positions (B) in the figure.) Next, Robonaut executes a guarded move, $\pi_p(p_{palm}, \mathbf{x}_{object}) \triangleleft \pi_f(p_{palm}, 0)$, toward the box. This moves the palms to the positions labeled (C) in Figure 3. At this point in the example, the heels of the two palms are touching the sides of the box. Next, executing the comply controller, $\pi_f(\{f_1, f_2\}, 0) \triangleleft \pi_f(p_{palm}, \mathbf{f}_{int}) \triangleleft \pi_p(p_{palm}, \mathbf{x}_{line})$, moves the palms to positions (D) in the figure. Now, both the fingers and the heel on each palm are pressing flat against the box. Finally, Robonaut executes $\pi_p(\{p_l, p_r\}, \mathbf{x}_{goal}) \triangleleft \pi_f(\{p_l, p_r\}, \mathbf{f}_{int})$ and lifts the box.

IV. Controller Refinement

In a *refining* sequence of controllers, the attractive domain of each controller in the sequence is a subset of the domain of all previous controllers. In addition, a refining sequence must satisfy the conditions required for controller funneling: each controller in the sequence must deliver the robot to a configuration within the attractive domain of the next controller. Refining controller sequences constitute an important class because of their discrete transition characteristics. While not all discrete control problems admit refining solutions, some problems, such as move-to-grasp, are naturally solved this way. This section reviews controller funneling and describes controller refinement.

In controller funneling, pairs of controllers that execute sequentially must satisfy the *prepares* condition. π_1 is said to *prepare* π_2 when the goal region of π_1 is inside the domain of π_2: $g(\pi_1) \subseteq \mathcal{D}(\pi_2)$. This condition guarantees that the robot always remains within the domain of attraction of the currently executing controller. A discrete control system that obeys this constraint is guaranteed to maintain control of the robot [7], [9]. One way to build such a 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. Breath-first-search may be used to search this graph for a sequence of controllers that leads to the goal configuration.

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 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 [9], [11]. 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 [9], [11]. When all "unsafe" actions are eliminated from the MDP, trial-and-error learning algorithms such as RL are able to explore the space safely.

Controller refinement defines an additional constraint beyond the *prepares* condition. If π_2 *refines* π_1, then the following conditions must be satisfied: First, π_1 must *prepare* π_2, *i.e.* $g(\pi_1) \subseteq \mathcal{D}(\pi_2)$. Second, the domain of attraction of π_2 must be a subset of the domain of π_1: $\mathcal{D}(\pi_2) \subseteq \mathcal{D}(\pi_1)$.

Not all discrete control problems admit refining solutions. However, when a refining solution is possible, it can be characterized in two special ways. First, if it is assumed that

State	Condition	Controller	Description
1	$r_{rmp} \geq 2.5m$	π_{ar}	approach region
2	$r_{rmp} < 2.5m$	π_{ao}	approach object
3	$r_{rmp} < 0.7m$	π_{reach}	reach toward object
4	$r_{palms} < 0.2m$	π_{gm}	guarded move
5	c_{palms}	π_{comply}	comply to object
6	$c_{heel} \wedge c_{tips}$	π_{lift}	lift object

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

controllers are only active within their domains of attraction, then no subsequent controller can cause the system to leave the domain of attraction of an earlier controller. In the context of a state-based representation, no controller can cause the system to transition to a previously visited state. Second, as a refining sequence of controllers executes, the robot is confined to a smaller and smaller region of configuration space. In the context of a state-based representation, the state of the robot can be defined in terms of the smallest domain of attraction that contains the current robot configuration. These characteristics are illustrated in the solution to the move-to-grasp problem that is described in the next subsection.

V. CONTROLLER REFINEMENT IN THE ROBONAUT-SCOUT FIELD STUDY

Controller refinement was explored in the context of the Robonaut-SCOUT field study. The Robonaut-SCOUT field study involves a mobile humanoid robot, the NASA/JSC Robonaut Unit B mounted on an RMP mobile base, and a semi-autonomous rover, SCOUT. Starting far away from SCOUT, Robonaut must avoid obstacles while navigating to a platform mounted on the rear of SCOUT. After reaching the platform, Robonaut must pick up a geological sample box placed there. The move-to-grasp problem is solved by executing the controllers described in Section II and III in a refining sequence. A deterministic policy is defined over states that roughly correspond to the domains of attraction of the navigation and hybrid force-position controllers. The state set and associated control policy is shown in Table III. The states are defined in terms of: 1) distance from the RMP to the target object, d_{rmp}; 2) distance of the two palms to the object, d_{palms}; 3) boolean contact state for the two palms (whether the palms are in contact with the object or not), c_{palms}; 4) boolean contact state for the heel of the palms, c_{heel}; 5) boolean contact state for the tips of the fingers, c_{tips}. Column two ("Condition") of Table III defines a sequence of states by adding constraints to earlier states. For example, the condition for state 4 denotes that $r_{palms} < 0.2m$ *and* $r_{rmp} < 0.7m$. This reflects the characteristic of refining control sequences described in Section IV, that states later in the sequence describe subsets of the configuration space represented by earlier states.

The policy in Table III does the following. When the RMP base is more than 2.5m away from the target object, the policy

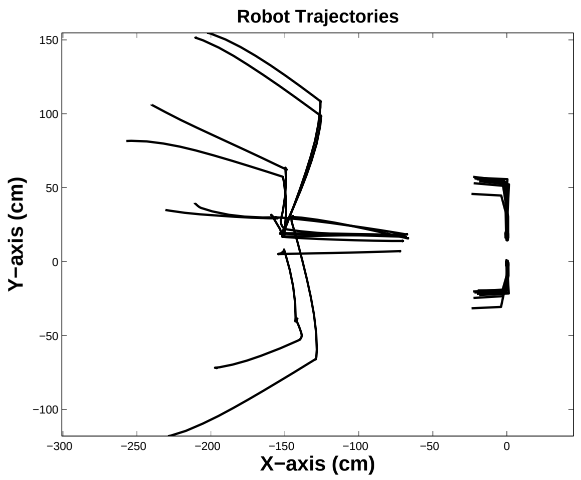

Fig. 5. The trajectories taken by Robonaut during the eight experimental trials. The "lightning-bolt" trajectories on the left side are the trajectories taken by the mobile base. The "L"-shaped trajectories on the right are the paths taken by Robonaut's two palms.

executes the APPROACH REGION controller, π_{ar}, that uses the visual location of the SCOUT vehicle to move the RMP to a point within 2.5m of the sample box. At this point, the policy executes the APPROACH OBJECT control policy that moves the RMP directly in front of the object. When the RMP is less than 0.7m from the box (state 3), the policy executes a reach controller that moves the hands around the box. Next, the policy executes a guarded move that makes contact with the sides of the box. After making contact, the control policy executes π_{comply} to comply to the sides of the box and π_{lift} to lift the box.

In order to characterize this solution to the move-to-grasp task, a series of eight trials were conducted where Robonaut navigated to and picked up a geological sample box measuring 7in$\times$8in$\times$11in. This experiment did not test the first controller in Table III, APPROACH REGION. Instead, Robonaut started each trial in a different location and orientation approximately 2.25m away from the box and executed the refining control policy illustrated in Table III. The experimental scenario is illustrated in Figure 4. In Figure 4(a), Robonaut is 2.25m away from the box. In Figure 4(b), Robonaut has navigated to a point just in front of the box. In Figure 4(c), Robonaut is lifting the box.

Figure 5 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 OBJECT control policy. Since, in each of these trials, Robonaut started less than 2.5m from the sample box, Robonaut is in state 2 in Table III and executes the APPROACH OBJECT control policy first. Since Robonaut is more than 1.5m away from the sample box, APPROACH OBJECT moves

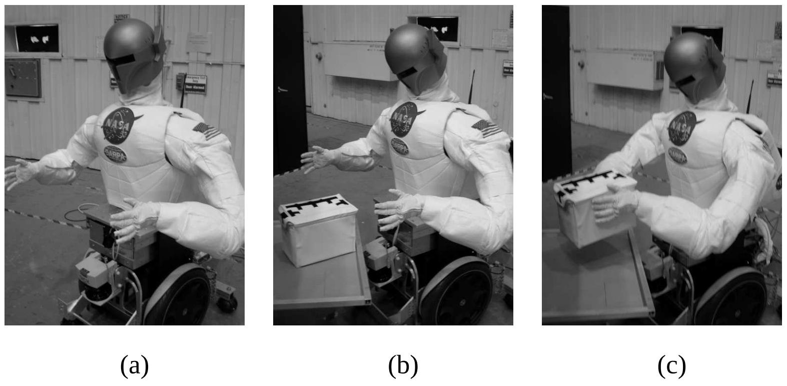

Fig. 4. Illustration of Robonaut completing the move-to-grasp task in the Robonaut-SCOUT field study.

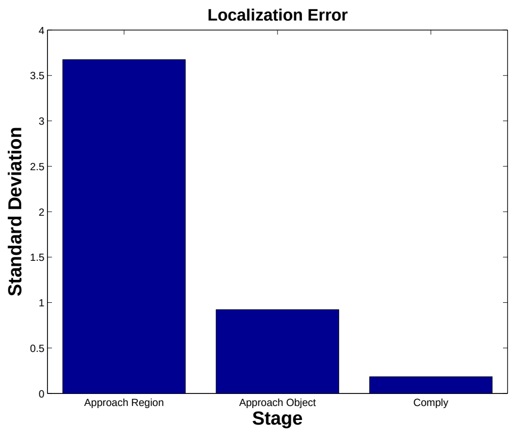

Fig. 6. Standard deviation in the estimated box position decreases as the refining control policy of Table III executes. The first bar, "approach region" gives standard deviation when Robonaut is approximately 2.25m away from the sample box. The second bar shows standard deviation after approaching the sample box. The third bar shows standard deviation after making contact and complying to the sides of the box.

directly toward the box. When it gets to a point within 1.5m, a transition to state 2 occurs (in the APPROACH OBJECT policy, shown in Table II) and Robonaut moves to a point along the axis of the box. When Robonaut reaches a point 1.5m directly in front of the box, the system transitions to state 3 in the APPROACH OBJECT policy and drives toward the box. After arriving in front of the box, APPROACH OBJECT terminates and the refining policy 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 5 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 in Figure 6. When Robonaut is 2.25m away from the box, the variance in the visually estimated position is large (the "approach region" bar in Figure 6). 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).

This improvement in localization accuracy is one reason why the sequence of executed controllers have a refining effect. The variance in estimated box position when Robonaut is 2.25m away suggests that information sufficiently accurate to solve this task in a single step is simply not available at the beginning of the task. Any solution must approach the object in stages, acquiring better information at each step. As long as the constituent controllers are themselves robust, a refining sequence of controllers will robustly move the robot closer and closer to the goal configuration. In addition, since this can be a refining sequence of *different* controllers, controllers at different stages of the task can use different kinds of information to solve the problem. This was particularly 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.

VI. CONCLUSION

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. This approach is explored in a move-to-grasp task where Robonaut must navigate to and pick 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. It is proposed that a refining sequence of controllers is a good way to take advantage of new and different sensory information accumulated by the robot during the progress of the task.

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] 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.

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

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

[6] 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.

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

[8] 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.

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

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

[11] 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.