Imitation Learning of Whole-Body Grasps

Kaijen Hsiao and Tomás Lozano-Perez
Computer Science and Artificial Intelligence Lab, MIT

***Abstract*—Humans often learn to manipulate objects by observing other people. In much the same way, robots can use imitation learning to pick up useful skills. A system is detailed here for using imitation learning to teach a robot to grasp objects using both hand and whole-body grasps, which use the arms and torso as well as hands. Demonstration grasp trajectories are created by teleoperating a simulated robot to pick up simulated objects. When presented with a new object, the system compares it against the objects in a stored database to pick a demonstrated grasp used on a similar object. Both objects are modeled as a combination of primitives—boxes, cylinders, and spheres—and by considering the new object to be a transformed version of the demonstration object, contact points are mapped from one object to the other. The best kinematically feasible grasp candidate is chosen with the aid of a grasp quality metric. To test the success of the chosen grasp, a full, collision-free grasp trajectory is found and an attempt is made to execute in the simulation. The implemented system successfully picks up 92 out of 100 randomly generated test objects in simulation.**

***Index Terms*—imitation learning, grasping, example-based grasping, whole-body grasping**

I. Introduction

One of the basic issues in making useful humanoid robots is enabling them to grasp and manipulate objects. In addition to well-understood fingertip grasps, humans often use what are termed "whole-body grasps"—grasps that can use non-fingertip surfaces such as an entire finger, palm, arm, or even torso. Examples of such grasps include wrapping a hand around the handle of a hammer, lifting a vase with two hands (as in the left side of Fig. 1), sandwiching a tennis racket under one arm, or slinging a club over a shoulder (as in the right side of Fig. 1).

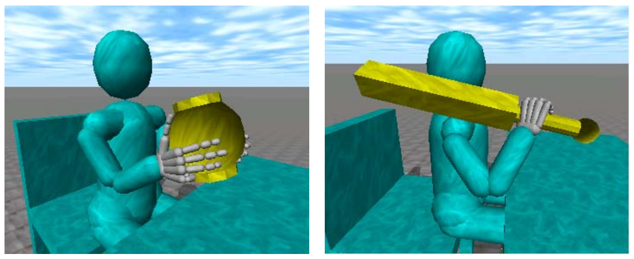
Fig. 1. Whole-Body Grasps

Using non-fingertip surfaces often makes grasps more powerful or stable. However, analyzing and synthesizing whole-body grasps is difficult because of the large number of point contacts required to approximate them, because there is the possibility of multiple steps in the grasp sequence, and because the kinematics of the robot puts a large number of constraints on the relative locations of non-fingertip contacts.

We are interested in finding a way to allow robots to perform complex grasp sequences with a potentially large number of contacts, as well as multiple steps in the grasp sequence. For instance, tucking a racket under an arm requires a grasp sequence with several steps: first the handle is grasped with a one-hand grasp, then the racket must be accurately placed beneath the arm, then the arm must sandwich the racket stably, and finally the hand must be removed. Figuring out how to accomplish this grasp sequence requires finding "grasps" (which we will also call "keyframes") with several different combinations of body parts—just the hand, the hand and the arm, and just the arm—and continuity must be maintained between keyframes, so that the hand grasps the handle in the same place in the first two combinations and the arm sandwiches the head in the same place in the latter two. The same issues arise in other complex grasp sequences such as regrasping operations, and while we are currently working on whole-body grasps, we would like our method to be applicable to both types of manipulation tasks.

Synthesizing grasps by constructing or globally optimizing individual contact locations requires time exponential in the number of contacts [21] since any contact can be placed anywhere on the object surface. Even when contacts can be found, the kinematics of the robot are typically not taken into account, and the resulting contact locations are often kinematically infeasible.

Humans can usually pick objects up simply by finding good pre-grasp locations and then wrapping hands around the object. Behavior-based, heuristic methods of grasping, such as [23] typically define hard-coded rules of grasping objects that identify possible pre-grasp locations for the hand to close around. These methods work well for simple situations, but it is difficult to generalize these heuristics to more complex, new tasks.

Instead of constructing new grasps from scratch or requiring that rules be hard-coded to deal with every situation, our method can create a database of successful grasp strategies through human demonstration. As long there is a grasp sequence in the database that can be applied to a new object, the task becomes one of picking the correct grasp sequence and then adapting it to the new object.

The goal of our project, therefore, is to enable a simulated robot to learn whole-body grasps through imitation: a human demonstrates picking up several simulated objects, and the robot chooses appropriate

Manuscript received November 10, 2005. This work was supported in part by an NSF Graduate Research Fellowship.

demonstrated grasp sequences and performs them on new objects with different geometries/positions than the training objects.

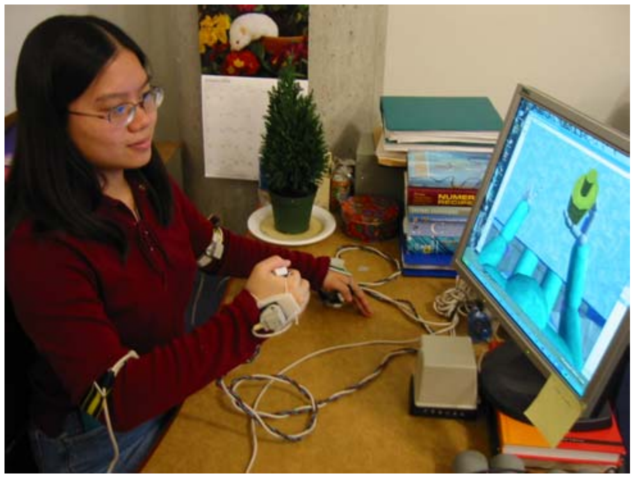
Fig. 2. Demonstrating Grasp

II. Approach

Our approach can be construed as a mix of behavior-based and individual contact methods. We use low-level controllers to grasp at good general locations on the object, as behavior-based methods often do, but we find those locations by adapting and examining sets of individual contacts.

Our general approach is as follows:

- A human demonstrates a database of grasp sequences by teleoperating the simulated robot, as shown in Fig. . Each demonstration is recorded as a sequence of keyframes in which contacts with the object are added or removed, and in which hand grasps with many contacts are represented by a reduced set of representative contacts.
- Given a new object, an appropriate demonstration grasp sequence is chosen from the database.
- The objects are modeled as combinations of basic primitives such as boxes, cylinders, and spheres for ease of grasp adaptation.
- Keyframes from the demonstration sequence are adapted to the new object. This is done by assuming that the new object is a transformed version of the demonstration object and by moving contacts appropriately, as in Fig. 3.
- Adapted grasp sequences are filtered for kinematic feasibility, and a grasp quality measure is used to pick the best one.
- A non-colliding trajectory to carry out the grasp sequence is found.
- The new grasp sequence is carried out, using low-level controllers to wrap hands stably around the object.

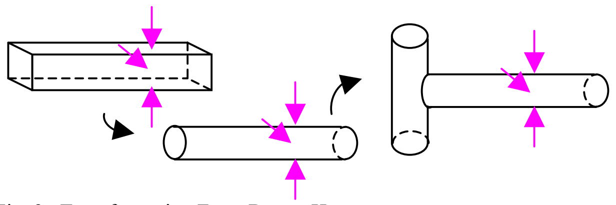
Fig. 3. Transformation From Box to Hammer

Although our approach offers no guarantees that it will find a good grasp (for instance, if no appropriate grasp is contained in the database then no adaptation of an existing grasp will be successful), it is possible to adapt the grasp in time polynomial in the number of grasp contacts; the adaptation process takes into account the kinematics of the robot; and our overall approach is potentially useful for other manipulation tasks such as regrasping.

III. Related Work

Aspects of our approach relate to diverse previous works in both grasping and imitation learning.

A. *Grasping*

In the realm of grasping, most of the approaches deal with finding or analyzing sets of contacts at precise locations on the surface of an object. The goal is to find a set of contacts that guarantee force-closure, and perhaps additionally that make up a high quality grasp according to some quality measure [17, 8, 15, 29, 22].

However, all of these approaches ignore the kinematics of the robot, assuming that the robot can not only reach any set of contacts on the surface of the object, but that arbitrary forces can be exerted at those contact points. One of the main ideas behind our approach is that the actual set of contacts that can be made by a hand is severely limited by the geometry of the hand, and thus finding sets of independent contacts that cannot be reached is wasteful. A few approaches take kinematics into account, such as [3, 4, 20, 25, 27].

Since we are dealing with enveloping, two-hand, and more complex whole-body grasps in addition to fingertip grasps, we need to handle issues that are less important for a fingertip grasp planner. A two-hand grasp can have as many as 32 contacts, and all of them are defective (meaning that there are not enough degrees of freedom to create arbitrary forces at each contact). Rather than trying to construct or search for good grasps from scratch, we can use previous experience to figure out how to grasp a new object that may be similar to one we have seen before. There are several approaches that deal with learning to grasp from experience, such as [5] and [11]; in the work that most closely relates to ours, [21] shows how to adapt a demonstrated grasp to a new object by finding a family of grasps that are guaranteed to have a quality value that is some percentage of the original grasp. This method is excellent for many situations and can deal well with more arbitrary-shaped objects than our method can. However, it is somewhat limited in that it only finds grasps within a particular family of grasps that are guaranteed to have a quality value that is at least a fraction of the original quality. This does not include many potentially desirable grasps, and requires searching the entire surface of the object in every possible degree of rotation, which can be difficult for 3-D objects.

On the other side of the field of robotic grasping are the heuristic approaches. In behavior-based or heuristic grasping, grasp taxonomies are used to grasp objects by classifying objects into categories that should be grasped by each canonical grasp, and then pre-shaping the hand into the appropriate grasp shape and using low-level controllers to execute the grasp. Works related to this sort of approach include [9, 6, 23, 12, 16].

In general, in order to extend any of the heuristic methods to dealing with under-arm, over-shoulder, or other grasps, one would have to hand-code heuristics for each new grasp type. While this is possible, it precludes extension to more complex, yet-unseen manipulation tasks.

Areas of grasping that are beyond the scope of this paper include dextrous manipulation (how to move the object within the hand, once it is already grasped) and second-order effects such as grasp stability (characterizing the stability of a grasp with respect to perturbations). We also ignore the effects of dynamics, assuming that the robot's movements are slow enough that such effects are minimal. For a more complete survey of the field of grasping, see [2].

B. *Imitation Learning*

In the field of imitation learning, there are works that deal with learning the dynamics of tasks such as dance movements [10], air hockey [1], and balancing a pole [24]. Of closer relevance are works dealing with the imitation learning of pick-and-place operations, such as [14, 28, 7, 17].

IV. Modeling Objects with Primitives

To make adapting grasps between objects easier, we require that models be provided for all objects that consist only of shape primitives such as spheres, boxes, and cylinders. Examples of such primitive models are shown in Fig. 4.

Modeling objects with primitives allows us to simplify the problem drastically by providing a sensible method of 'chunking' each object; also, the symmetries inherent in the primitives provides a reduced number of rotational alignments between objects.

This primitive modeling is only used for transforming contacts from one object to the other. Although our current implementation only works with the primitive models, prior to and after transforming contacts, more complex models can be used. Contact points on the complex model would merely be transferred to the nearest points on the simplified model, and vice versa. Differences between the actual geometries and their primitive models can be treated essentially as errors. This means that objects that are poorly modeled by a small number of such primitives may be grasped incorrectly by our system. As you can see in Fig. 4, however, for the purposes of grasping, even fairly complex objects can be reasonably modeled with primitives.

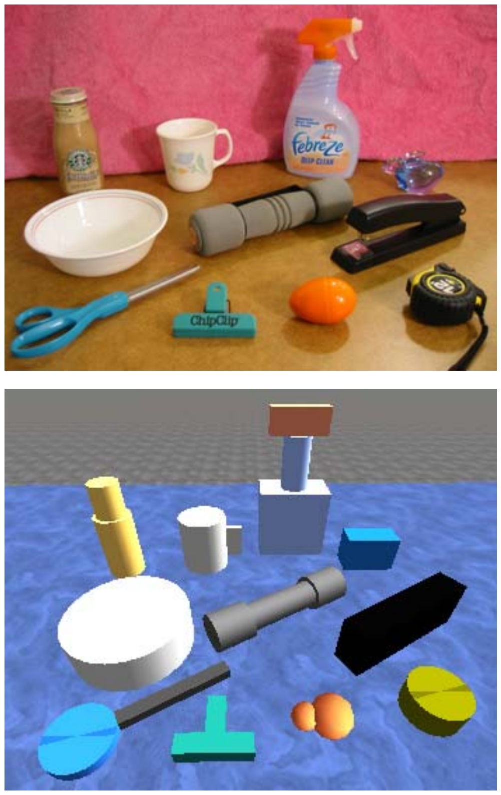
Fig. 4. Real objects and their primitive models

While it is possible to implement a system that automatically finds these primitive models, such is beyond the scope of this project; the primitive models that we use are generated by hand. In our implementation, we use only objects consisting of a maximum of three primitives (sphere, cylinder, and box) in a line, with axes of symmetry aligned, as explained earlier in the definition of primitive models. If you look at the objects in Fig. 4, you may note that all the primitive models are of this description. With a moderate increase in the complexity of the grasp adaptation process, it would be possible to use more complicated primitive models, with other types of primitives such as handles or cones, more primitives, or differently arranged primitives.

V. Demonstrating Grasps

The grasps in the template grasp database are created by having a human teleoperate the simulated robot to pick up simulated objects. Our implementation uses the Nest of Birds$^{(TM)}$, a set of four magnetic sensors that determine the position and orientation of both wrists and elbows. Additionally, switches held in each hand allow the user to choose one of three pre-grasp configurations (C-shaped hand, L-shaped hand, or flat palm) and tell each hand when to wrap around the object and when to let go.

Data is recorded at the start and end of the simulation, as well as every time contact between the object and a new body part is made or lost. Each of these recorded points, along with the information recorded, is called a keyframe; a demonstrated grasp trajectory is thus a sequence of keyframes. For instance, our demonstration of tucking a sign under one arm has seven keyframes: start position, hand grasping handle, sign touching torso, sign touching upper arm, sign touching lower arm, hand being removed, and end of simulation. Six of the seven keyframes (all but the hand being removed, since it is nearly identical to the one before it) are shown in Fig. 5. The parameters recorded for each keyframe are: global object position, arm joint angles, and locations of contact points on both the object and the body/table.

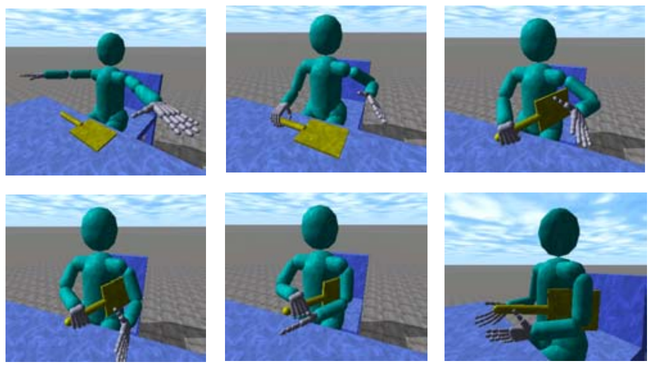
Fig. 5. Keyframes in Under-Arm Grasp Demonstration

Sliding of contacts is highly discouraged, since controllers that can successfully execute contact sliding are difficult to create and the resulting adapted grasps are likely to be ruined; however, it is difficult to eliminate all sliding while demonstrating complex grasps such as under-arm grasps. A picture of a user demonstrating a grasp using the Nest of Birds$^{(TM)}$ is shown in Fig. 2.

The simulated world you see in Fig. 2, which is the same world used to execute adapted grasps of new objects, is created using Open Dynamics Engine (ODE), an open-source physics simulator that provides a fair approximation of real-world physics and collision detection [26].

VI. Representative Contacts

To further simplify choosing good pre-grasp locations, instead of transforming all contacts made in the demonstration grasp, we reduce the potentially large number of contacts made by one hand to a small set of representative contacts. The geometry of the hand imposes severe constraints on the relative locations of the hand contacts. Finding 16 separate locations for points on a single hand is terribly wasteful, since for a given hand position, the range of possible contacts is severely limited.
For our current implementation, we track only the position of the middle knuckle on the palm relative to the object, so that the transformed contact only gives the general pre-grasp location on the new object. This works well enough in most cases. However, without tracking the positions of the fingertips, it is possible (albeit rare) to generate a grasp in which the fingertips miss the object entirely, particularly for objects with very small components. In our newest implementation, we plan to track up to three points for each hand grasp made: the contact points closest to the tip of the middle finger, the tip of the thumb, and the middle knuckle on the palm. An example of the three representative contacts chosen is shown in Fig. . This is akin to simplifications made using the idea of virtual fingers: the concept of virtual fingers says that a number of fingers (or other body parts) often work in concert to exert a force, and thus can be viewed as a single "virtual finger". In our case, all four fingers are regarded as a single virtual finger, with the thumb providing the second and the palm (when in contact) providing a third. For instance, the power grasp in Fig. 6 can be represented adequately by an inward force exerted at each of the three representative contacts.

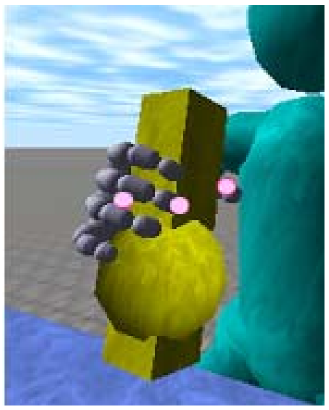
Fig. 6. Representative Contacts

To find a good hand pre-grasp location using a set of representative hand contacts after grasp adaptation, we can set the fingers in the desired pre-grasp configuration and then use a quick optimization over the arm joint angles to figure out where to place the hand such that the contacts can be made. Assuming that the contacts are in a sensible location for grasping on the object (which we will check using a grasp quality measure on estimates of the resulting contact locations), the low-level grasp controllers will take care of wrapping around the object to create a stable force-closure grasp.

VII. Picking a Template Object

When presented with a new object, a template grasp must be chosen from the database of demonstrated grasps. This is done by choosing the grasp that was used on the most similar object, as determined using a nearest-neighbor classification system. The parameters we used to compare objects were object dimensions, object mass, inertia in each of three directions, and the automatically-assigned z-axis of the object, which is based on the alignment of primitives in the object. While this does not take into account fine features that may be useful for selecting an appropriate grasp, bulk object characteristics appeared to be both more important and sufficient for applying the template grasps we chose. While two objects with a similar, hand-sized protrusion could both be grasped by wrapping a hand around the protrusion, if one of the two objects is large and heavy, a human would be more likely to grasp it with a two-hand grasp that provides greater support.

The nearest neighbor classification system can be used to rank the template grasps, and more than one of the best grasps can be examined for suitability, particularly if the size of the database is large.

VIII. Adapting Contacts

Because each object is made up only of a small number of known primitives, we can imagine morphing one object into the other through a small set of geometric transformations such as expanding/shrinking primitives, morphing one primitive to another, adding/removing primitives, or splitting/combining primitives. Now imagine grasping the demonstration object and then morphing the object within the grasp according to these transformations. If the two objects are reasonably similar, it is likely that the grasp will still succeed. This is the intuition behind our method of contact transformation, which essentially equates 'chunks' of one object consisting of subsets of that object's primitives with 'chunks' of another object, and grabs both 'chunks' in the same manner.

This is particularly useful for grasp sequences or other manipulation tasks that have multiple steps, such as picking up a racket and sandwiching it under an arm, or object regrasping operations, since each grasp in the sequence is connected to the grasp before and after. If you grasp the racket initially by the handle, that hand grasp should remain the same while the arm starts to sandwich the head of the racket. By equating part of the new object with the handle of the template racket, and grasping that part in the same manner as the handle throughout the entire grasp sequence, grasp continuity is automatically assured. This is not true of grasp generation from scratch, or even of grasp adaptation that searches for good contacts separately for each grasp in a sequence. In such a situation, one would have to make continuity a separate requirement. If the first keyframe has g possible grasps, and each of those grasps is consistent with approximately g possible grasps for the next keyframe, finding good grasps for the first two keyframes requires searching on the order of g^2 possible combinations. If there are k keyframes in the entire sequence, this quickly blows up into searching through a tree whose size is on the order of g^k grasps.

As shown in Fig. 7, there are many sequences of transformations that will morph one object into another—in this case, a box into a hammer. In the first transformation, the box shrinks into the head of the hammer, and a cylindrical handle is added on. In the second, the box shrinks to the size of the handle, morphs into a cylinder, and has a box head tacked on. In the third, the box splits into two primitives, one of which becomes the head of the hammer, and the other of which shrinks to the size of the handle and then morphs into a cylinder. While there are more roundabout ways of morphing from the box to the hammer with this particular relative orientation, those are the shortest routes, and the only ones of interest to us.

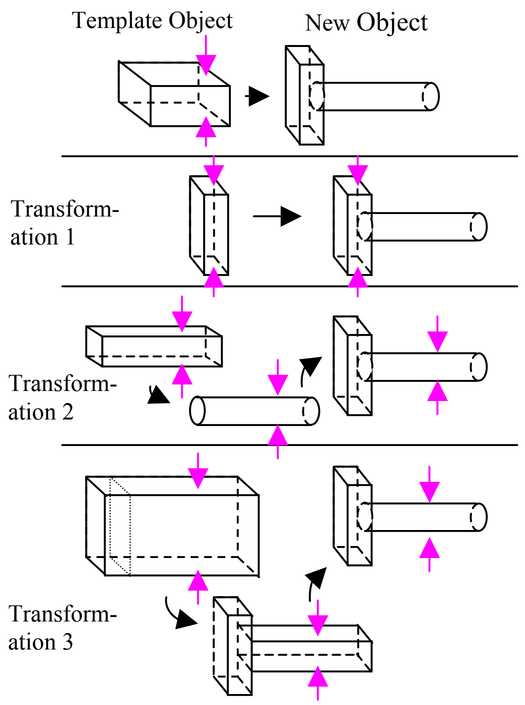

Fig. 7. Transformation sequences from box to hammer

A. *Mapping Contacts Through a Transformation*

For each sequence of transformations, we need a method of transforming the contacts from an object to its transformed equivalent. In doing so, we would like the contacts to roughly maintain their relative positions, while staying on the appropriate chunks of the new object.

One simple method that might come to mind is to move the contacts as if they were on the surface of a sponge, so that a contact on the corner of a box remains on the corner while the box stretches or shrinks, and squashes to the closest point on a sphere as if the box's corners were squashed inward. Thus, contacts grow or shrink with the boundary of the object, maintaining their relative position with respect to edges and corners. We will call this method of mapping contacts "dimensionally normalized coordinates", because it preserves the relative positions of the contacts regardless of the dimensions of the old and new chunks. An example of this sort of contact transformation is shown in Fig. 8; the corner contact on the square maps to the "corner" of the circle, or to the corner of the stretched-out square.

This method of mapping contacts is used in our current implementation. It has the advantage that relative contact positions are maintained fairly well, and the new object chunks are grasped in a similar manner as the demonstration object chunks. This works well in many cases, particularly when the objects have a high coefficient of friction. However, this type of contact mapping sometimes yields poor transformed grasps, particularly when mapping from boxes to spheres. If the square in Fig. 8 were grasped by the top and bottom faces near the top right corner and the bottom right corner, the mapped contacts on the circle would land at the equivalent "corners" of the circle, which would only stably grasp the circle if the coefficient of friction were extremely high. Thus, for our next iteration, we will map contacts through a transformation in a method that tries to preserve the quality of the grasp in addition to just the relative positions of the contacts. We will call this mapping "quality-preserving contact transformations." Before we can explain the new mapping, however, we must explain the concept of grasp quality.

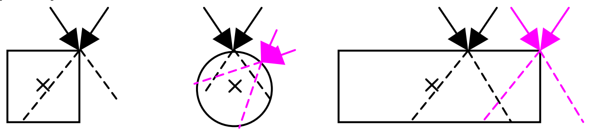
Fig. 8. Quality-Preserving Contact Transformations (dark arrows) vs. Dimensionally Normalized Coordinates (light arrows)

B. *Grasp Quality Measure*

To judge whether a set of mapped contacts would result in a good grasp, we need a quality measure that will give a numerical value of goodness for a set of grasp contacts. There are many different grasp quality measures that can be used to evaluate grasps based on their contact points, such as those found in [29] and [15]. One of the simplest and most commonly used is the L_1 quality metric found in [8], which we will explain briefly and use here. The purpose of this quality measure is to determine how well a grasp, defined as a set of contact points on an object, is able to resist arbitrary disturbance wrenches. These can include the force of gravity when the object is at an arbitrary orientation, or external contact forces hitting the object. More specifically, this particular quality metric is a measure of the magnitude of the largest disturbance wrench that can be resisted, when that wrench is exerted in the worst direction possible for breaking the grasp and the total of the normal forces exerted at all the contacts sums to 1. This quality value is found by taking the convex hull of all the wrenches possible at all grasp contacts, and finding the radius of the largest sphere centered at the origin that fits within that convex hull. The quality value is that radius. For details about why this is, see [8].

C. *Quality-Preserving Contact Transformations*

Given this quality measure, an important question to ask when adapting a grasp from one object to another is: what happens to the quality value when contact wrenches move in the wrench space? If you look at the 2-D example in Fig. 9, if you start with a convex hull with quality q, all the points on the convex hull must be at least a distance q away from the origin. If you move vertices of the convex hull, points on the face of the convex hull formed by those vertices (in this case, the line between two vertices) move by an amount that can be found by interpolating the distance moved by the boundary vertices. Thus, assuming the vertices of the convex hull remain the same, no point on the surface of the convex hull can move any more than the maximum amount moved by any one wrench. Since all points on the convex hull start out at least a distance q, and since the distance can drop no more than $\max(d_1 \ldots d_n)$, where $d_1 \ldots d_n$ are the distances moved by the vertices of the convex hull, all the points on the convex hull must then be at least a distance $q - \max(d_1 \ldots d_n)$. If wrenches previously on the inside of the convex hull move to become one of the vertices on the surface of the convex hull, that can only increase the distance of surface points from the origin, since the convex hull with such points contains the entire volume of the convex hull without them. If wrenches previously on the outside of the convex hull move to the inside of the new convex hull, that only means that some points move a smaller amount than otherwise expected. Thus, our new quailty $q' \geq q - \max(d_1 \ldots d_n)$.

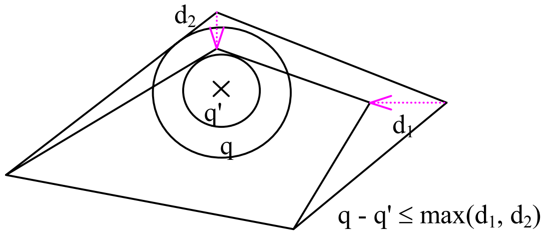

Fig. 9. 2-D Diagram of Shrinking Quality

Thus, when figuring out how to move contacts from one object to its transformed equivalent, our goal will be to move the wrenches exerted at that contact as little as possible, maintaining kinematic feasibility whenever possible. We will call the resulting contact transformations 'Quality-Preserving Contact Transformations.' As an example, when mapping the corner contact on the square shown in Fig. 8 to the circle, the torques exerted from any point on the circle are the same. However, the directions of the forces exerted at different contact points vary. Thus, rather than mapping to the "corner" of the circle, which would change the directions of the forces considerably, we map to the top of the circle, which results in keeping the force directions the same. When mapping from the square on the left to the stretched-out rectangle on the right, rather than mapping to the corner as before, we map to the location on the face that keeps both forces and torques the same as before.

To transform contacts through a sequence of transformations, each quality-preserving contact transformation is performed in turn. As a shorter way to think about how contacts should be transformed, we can consider the perspective that a sequence of transformations is equivalent to matching "chunks" of an object to "chunks" of the new object, with each chunk in the old object morphing into the equivalent chunk in the new object. For each matching chunk-pair, we can expand all the primitives of the original object to just fit inside the closest bounding box/cylinder/sphere that fits around both chunks, combine them into one primitive, then split that bounding primitive into appropriate parts that shrink into the primitives in the new object. Thus, merely considering all the different ways of matching chunks between objects covers all the relevant sequences of transformations.

D. Choosing the Best Grasp Candidate

The process of finding all the possible transformations between template and new object and adapting the contacts using the quality-preserving contact transformations results in a large number of possible adapted grasp sequences, which we will refer to as grasp candidates. To pick the best one, there are two factors to consider: kinematic feasibility and grasp quality. To decide either, however, we must first do a quick optimization over the arm angles for each grasp candidate to find the approximate hand/arm locations that will best make the desired adapted contacts for each keyframe. Once that is done, we can quickly eliminate any grasp candidates that have either major collisions or awkward arm angles. Next, grasp candidates that are too similar to each other can be eliminated, so that for small objects, we do not consider a large number of nearly identical grasps.

After paring down the grasp candidates in this manner, we need to find a way to assign each remaining grasp candidate a numerical quality value. Our current implementation uses a quality value consisting of a weighted combination of an empirically chosen set of features whose weights are learned from a training set. These features include geometric overlap of the template and new objects when overlaid, the distances that the contact points move when mapped, awkwardness of the arm positions, and level of collisions found between body parts and the object. While this method works fairly well at picking good grasps, it is somewhat slow and does not directly reflect the ability of a new grasp candidate to hold the object stably.

Instead, we can use one of the quality measures discussed earlier, such as the one from [8]. To do this, the approximate kinematically feasible contact locations that can be made for each grasp candidate must be estimated. This is done by closing the fingers (without actual physics involved, just checking collisions for various points from fully open to fully curled) until they hit the surface of the object. Using those estimated contact locations, we can check the expected quality of the grasp by finding the convex hull of the possible wrenches at all the contact points. The best grasp is then the remaining grasp candidate with the highest grasp quality value.

IX. Keyframing

In order to test our proposed grasps, we must actually carry out the proposed grasp sequences in simulation. This process involves adding a few approach and depart keyframes to those adapted from the demonstration, adjusting the keyframes so that they are entirely non-colliding, finding non-colliding paths between those keyframes, and finally carrying out the proposed grasp trajectory to test for success.

For each keyframe, a collision-free arrangement of both arms and object must be found. This is done by optimizing over the arm angles, penalizing for collisions and encouraging positions that accurately make the desired estimated contacts that were used to calculate grasp quality. For keyframes in which the robot has control of the object, the object is assumed to move with the body parts in contact with the object; hand contacts are given precedence over arm contacts, which are more likely to need to shift. At this stage, it is possible to test the ability of the robot to statically support the object in the desired keyframe position. If any keyframe in the grasp sequence cannot adequately support the object without dropping it, the grasp is declared unsuccessful.

X. Finding Trajectories

Once a sequence of keyframes is found, we must still find collision-free trajectories to traverse the keyframes. To do this, we use a probabilistic roadmap, as described in [13]. Briefly, the idea is to connect the keyframes with a series of sub-goal positions, each of which is connected to the next by a direct, collision-free path. Each sub-goal position is a node in the graph representing the probabilistic roadmap, and a direct, collision-free path between two nodes is recorded as an edge in the graph. To populate the graph, sets of joint angles (nodes) are selected at random, and the ability of that node to connect to its closest neighbors through a direct, collision-free path is tested. If a path is found, an edge is created between the two nodes. When two keyframes are in the same connected component, there is a collision-free path from one to the other.

XI. Executing Trajectories

Once a proposed collision-free grasp trajectory is found, it must be executed in simulation to test whether the grasp is successful. The simulation is only an attempt to approximate the real world, so while the resulting trajectory can be executed with the same controllers and sensors in the real world, a successful grasp in simulation still has only some (hopefully reasonably high) probability of working in real life.

There are three main types of controllers that are needed when executing a planned trajectory. The first perform position control of the arms, to move them along the trajectory. The second add torque components to the arm joints (on top of the position control torques) that attempt to apply appropriate forces at contact points between arm surfaces and the object. To apply forces at contact points between the object and fixed body parts such as the torso, the joint torques are calculated by assuming that the object moves with the arm/hand parts in contact with the object. This is a loose approximation, and often results in imprecise but sufficient force generation.

The third type of controller wraps fingers around an object when a hand is in the appropriate position for grasping. The finger and thumb joints are made to bend along a preset trajectory that creates a natural closing motion. When a joint hits the object, all proximal joints on that digit are frozen in place while distal joints continue to curl. When the tip of the digit has hit the object, the finger freezes its shape except at the base of the digit, which is proportional-controlled to maintain a given level of force on the object.

Our current implementation uses a constant level of force for all grasps. This turns out to be problematic, since for instance, in a two-hand grasp, the fingers should only wrap loosely around the object while the palms exert most of the necessary force to hold up the object. On the other hand, for fingertip grasps of medium-weight objects, the fingers must exert a fair amount of force to maintain their grasp. If the larger amount of force is used for both grasps, the fingers tend to push too hard against the object in the two-hand grasp, potentially disrupting the grasp. Thus, for our next iteration, we plan to use different levels of force for different pre-grasp configurations. More generally, finger force is a parameter that could be either learned or specified by the user for each demonstration.

The other major issue encountered by this simplistic system of controllers is that of sliding. As mentioned earlier, sliding in the demonstration is discouraged but difficult to eliminate in certain complex grasps. Thus, since the controllers have no special accommodation for sliding along surfaces, grasps that require sliding are usually broken. For example, if you look carefully at the keyframes for the under-arm grasp in Fig. 5, you may note that the object shifts position considerably while being sandwiched by the arm. This is an artifact of the under-arm grasp demonstration, which contained a considerable amount of sliding. This resulted in the failure of all adapted under-arm grasps, since attempts to slide the object resulted in dropping it. Fortunately, there is little sliding needed in standard one-hand, two-hand, or over-shoulder grasps, and thus this was not an issue for those grasps.

XII. Results

The seven template grasps supplied in our database include two precision grasps, two palm grasps, one two-hand grasp, an over-shoulder grasp, and an under-arm grasp, as shown in Fig. 10.

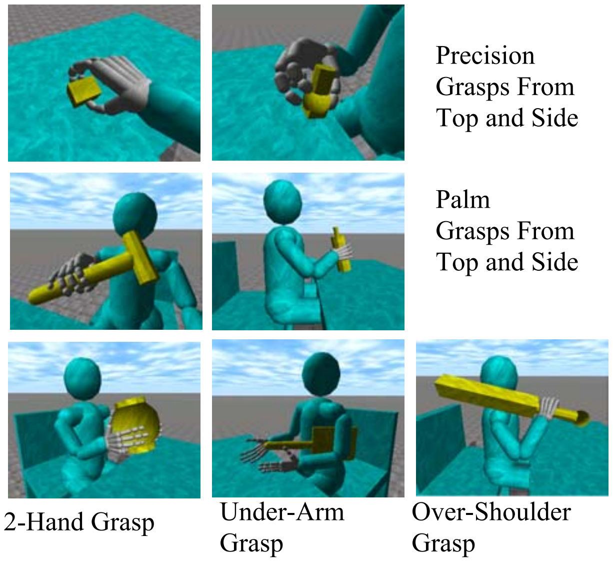
Precision Grasps From Top and Side
Palm Grasps From Top and Side
2-Hand Grasp
Under-Arm Grasp
Over-Shoulder Grasp

Fig. 10. Template Grasps

By adapting these seven template grasps, our current implemented system can already pick up 92 out of 100 randomly generated objects. Examples of successfully executed grasps of a few objects are shown in Fig. 11.

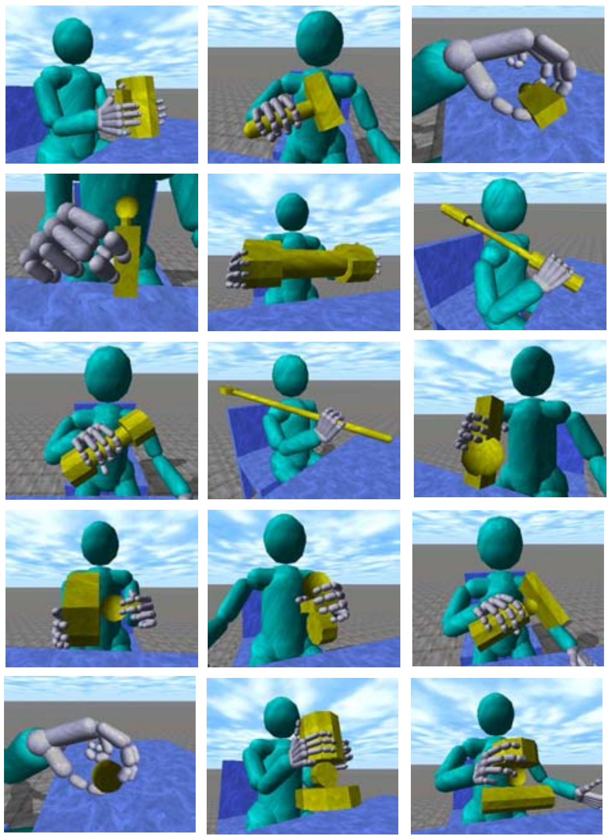

Fig. 11. Successful Grasp Examples

The problems that we discussed earlier with current implementations of various modules in our algorithm were responsible for a fair number of the grasps that failed. A few failed grasps are shown in Fig. 12.

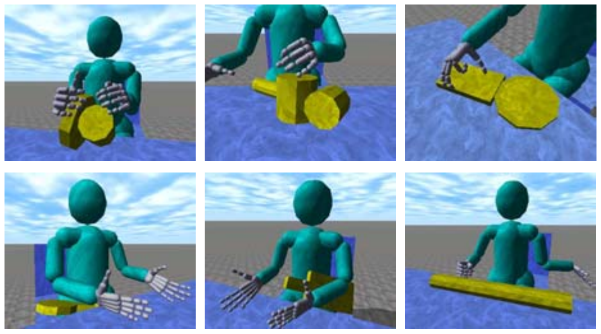

Fig. 12. Failed Grasps

As mentioned earlier, none of the under-arm grasps succeeded due to sliding in the demonstration. Two of the failed objects in Fig. 12 (bottom left and bottom middle) were attempts at performing an under-arm grasp. The top left and top middle objects failed only because of an inappropriate level of finger force being applied by the low-level grasp controllers; these would have succeeded with the force modifications discussed. The top right grasp failed because the hand was not actually large enough to fit around the object; this would have been detected by the convex hull grasp quality metric. The grasp of the log in the bottom right failed because the log was too heavy to be picked up in the same manner as the over-shoulder grasp. Had we used the convex hull grasp quality metric, we could have used the quality of the grasp along with the weight of the log to determine the infeasibility of the grasp before trying to execute it. While the under-arm grasps would not be fixed by implementing the rest of the techniques detailed in this paper, we believe that most of the other failures could have been avoided.

XIII. Future Work

In addition to implementing the components discussed that have not yet been finished, there are several other components that we plan to add. The first is to add in the object model and position errors; to this point, we have tested only on primitive models with exact position information. In the future, we intend to use more complex models, and to add in varying levels of error in both object model and position. Using data gained from testing with varying levels of error, we would like to learn the uncertainty in the quality of the resulting grasp based on features that are associated with instability and other grasp difficulties. We also plan to create a module that will automatically determine an appropriate primitive model given a more complex model, and add new primitives and a larger number of primitives to our primitive models. Finally, we plan to extend the ability of our system to perform manipulation tasks other than grasping, such as opening doors, stacking dishes, or screwing in screws.

References

[1] Bentivegna, Darrin C. and Christopher G. Atkeson, "Learning How to Behave from Observing Others," *Workshop on motor control in humans and robots (SAB 2002)*, Edinburgh University, August 10-11, 2002.

[2] Bicchi, Antonio. "Hands for Dexterous Manipulation and Robust Grasping: A Difficult Road Toward Simplicity," *IEEE Transactions on Robotics and Automation*, Vol. 16, No. 6, 2000.

[3] Bicchi, Antonio. "On the Problem of Decomposing Grasp and Manipulation Forces in Multiple Whole-Limb Manipulation," *Journal of Robotics and Autonomous Systems*, 1994.

[4] Borst, Ch., M. Fischer, and G. Hirzinger, "A Fast and Robust Grasp Planner for Arbitrary 3D Objects," *ICRA*, 1999.

[5] Coelho, J., J.H. Piater, and R.A. Grupen, "Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot," in *First IEEE-RAS International Conference on Humanoid Robots*, September 2000.

[6] Cutkosky, M.R. and R.D. Howe, "Human Grasp Choice and Robotic Grasp Analysis," in *Dexterous Robot Hands*. Springer-Verlag, 1990.

[7] Ehrenmann, M., Zoellner, R.D., Rogalla, O., Dillmann, R. "Programming Service Tasks in Household Environments by Human Demonstration," *IEEE Intl. Workshop on Robot and Human Interactive Communication*, 2002.

[8] Ferrari, Carlo, and John Canny. "Planning Optimal Grasps," *ICRA*, 1992.

[9] Iberall, T., and MacKenzie, C.L., "Opposition Space and Human Prehension," *Dextrous Robot Hands*, Springer-Verlag, 1990, p.32-54.

[10] Jenkins, O. C., Matari'c, M. J. & Weber, S., "Primitive-Based Movement Classification for Humanoid Imitation," in `Proceedings, First IEEE-RAS International Conference on Humanoid Robotics', Cambridge, MA, MIT, 2000.

[11] Kamon, Ishay, Tamar Flash, and Shimon Edelman, "Learning to grasp using visual information," *ICRA*, 1996.

[12] Kang S.B. and K. Ikeuchi, "Toward automatic robot instruction for perception - recognizing a grasp from observation," *IEEE Transactions on Robotics and Automation,* vol. 9, pp. 432-443, Aug. 1993.

[13] Kavraki, Lydia E., P. Svestka, J. Latombe, and M. Overmars. "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," In *IEEE Trans. on Robotics and Automation*, 12(4), 566-580, 1996.

[14] Kuniyoshi, Y., M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 6, pp. 799-822, December 1994.

[15] Li, Z., and Sastry, S., "Task-Oriented Optimal Grasping by Multifingered Robot Hands," *IEEE Journal of Robotics and Automation*, Vol. 4, 1988.

[16] Miller, A., S. Knoop, H. Christensen, and P. Allen, "Automatic Grasp Planning Using Shape Primitives," *ICRA*, 2003.

[17] Nguyen, Van-Duc. "Constructing Force-Closure Grasps," *The International Journal of Robotics Research*, Vol. 7, No. 3, June 1988.

[18] Ogata, H. and T. Takahashi, "Robotic assembly operation teaching in a virtual environment," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 391-399, June 1994.

[19] Platt, R., A. Fagg, and R. Grupen. "Extending Fingertip Grasping to Whole Body Grasping," ICRA, 2003.

[20] Platt, R., A. Fagg, and R. Grupen. "Nullspace Composition of Control Laws for Grasping," *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2002.

[21] Pollard, Nancy. "Synthesizing Grasps from Generalized Prototypes," *ICRA*, 1996.

[22] Ponce et al. "On computing four-finger equilibrium and force-closure grasps of polyhedral objects," *International Journal of Robotics Research*, Vol. 16, 1997.

[23] Rijpkema, H., Girard, M., "Computer Animation of Knowledge-Based Human Grasping," *ACM SIGGRAPH*, pp 339-348, 1991.

[24] Schaal, S., "Learning from demonstration," In M. Mozer, M. Jordan, and T. Petsche, editors, Advances in Neural Information Processing Systems 9, pages 1040-1046. MIT Press, Cambridge, 1997.

[25] Simeon, Theirry, Laumond, J.P., Cortez, J, Sahbani, A. “Manipulation Planning with Probabilistic Roadmaps,” Int’l Journal of Robotics Research, Vol. 23, No. 7-8, 729-746, 2004.

[26] Smith, R. Open Dynamics Engine, www.ode.org, 2005.

[27] Strandberg, Morten, “Robot Path Planning: An Object-Oriented Approach,” Ph.D. diss., Royal Institute of Technology, Stockholm, Sweden, 2004.

[28] Tung, C. and A. Kak. "Automatic Learning of Assembly Tasks using a Dataglove System," *IROS*, 1995.

[29] Zhu, X., H. Ding, and H. Li. "A Quantitative Measure For Multi-Fingered Grasps," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2001.