Imitation Learning of Whole-Body Grasps

Kaijen Hsiao
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: kjhsiao@mit.edu

Tomas Lozano-Perez
Computer Science and Artificial Intelligence Lab
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: tlp@mit.edu

***Abstract*— 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, modeled as combinations of up to three aligned primitives—boxes, cylinders, and spheres. When presented with a target object, the system compares it against the objects in a stored database to pick a demonstrated grasp used on a similar object. By considering the target object to be a transformed version of the demonstration object, contact points are mapped from one object to the other. The most promising grasp candidate is chosen with the aid of a grasp quality metric. To test the success of the chosen grasp, a collision-free grasp trajectory is found and an attempt is made to execute it in simulation. The implemented system successfully picks up 92 out of 100 randomly generated test objects in simulation.**

I. Introduction

One of the challenges in humanoid robotics is enabling robots to grasp and manipulate objects as flexibly as humans do. 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. Such grasps include wrapping a hand around the handle of a hammer, lifting a vase with both hands, sandwiching a racket under one arm, or slinging a club over a shoulder. We are interested in enabling robots to do both fingertip and whole-body grasps; examples of the types of grasps we do are shown in Fig. 1.

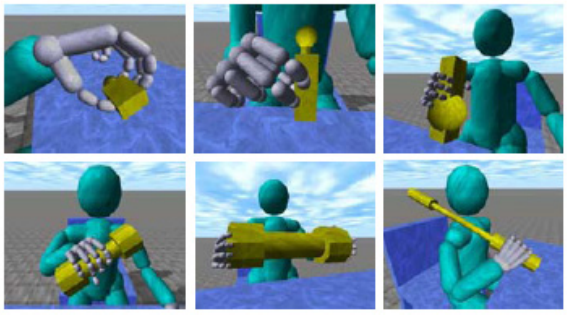

Fig. 1. Example Fingertip and Whole-Body Grasps

Using non-fingertip surfaces often makes grasps more powerful or stable. However, analyzing and synthesizing whole-body grasps is difficult because they may involve a large number of contacts, because there may be multiple steps in the grasp sequence, and because there are many constraints arising from the kinematics of the robot.

Our goal is to plan complex grasp sequences with a 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 of the racket 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 worst-case time exponential in the number of contacts [18]. Furthermore, such optimal grasps may be kinematically infeasible and so additional search may be required.

Humans appear to grasp most objects by finding good pre-grasp locations and then wrapping their hands around the object. Taxonomy-based, heuristic methods of grasping, such as [20], 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 them to new tasks.

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

Our goal, 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 demonstrated grasp sequences and performs them

on target objects with different geometries/positions than the training objects.

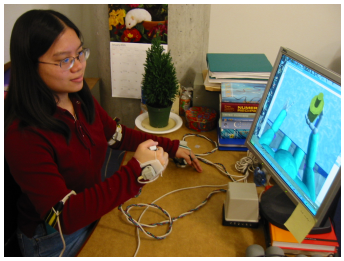

Fig. 2. Demonstrating Grasps

II. Approach

Currently, we model objects as a set of at most three primitives (boxes, cylinders, or spheres). The centers of the primitives lie on a line and their axes of symmetry must be either perpendicular or parallel to each other. Arbitrary orthogonal axes are used for spheres and for the curved part of cylinders.

Our general approach is as follows:

1) A human demonstrates a database of grasp sequences by teleoperating the simulated robot, as shown in Fig. 2. 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.
2) Given a target object, an appropriate demonstration grasp sequence is chosen from the database.
3) Keyframes from the demonstration sequence are adapted to the target object. This is done by assuming that the target object is a transformed version of the demonstration object and by mapping contacts appropriately, as in Fig. 5.
4) Adapted grasp sequences are filtered for kinematic feasibility, and a grasp quality measure is used to pick the best one.
5) A collision-free trajectory to carry out the grasp sequence is found.
6) The new grasp sequence is tested in simulation, using low-level controllers to wrap hands stably around the object.

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 that is independent of the number of contacts in the grasps; 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

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, as in [6], [25], and [19].

However, 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 [17], [16], [12], [21], [23], and [2].

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 on the order of 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 [3] and [8]. In the works that most closely relate to ours, [18] 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, and [11] shows how to create humanlike enveloping grasps by finding portions of objects that fit the curvature of a specific demonstrated hand shape.

A number of more taxonomy-based appraoches to robot grasping have been proposed. In taxonomy-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 [7], [4], [20], [9], [14].

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. For a more complete survey of the field of grasping, see [1].

In the field of imitation learning there is work dealing with the imitation learning of pick-and-place operations, such as [24], [5], and [15].

IV. Representations

A. 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. 3.

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 relevant rotational alignments between objects.

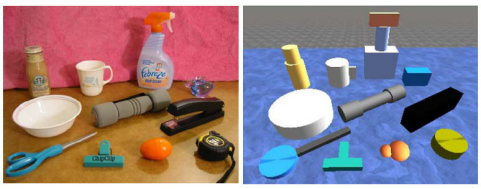

Fig. 3. Real Objects and Their Primitive Models

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 could 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. Fig. 3, however, shows that for the purposes of grasping, even fairly complex objects can be reasonably modeled with primitives.

In our implementation, we use only user-provided objects consisting of a maximum of three primitives (sphere, cylinder, and box) in a line, with axes of symmetry aligned. All the primitive models of objects in Fig. 3 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.

B. Template 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. The information recorded at each of these points 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. 4. 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.

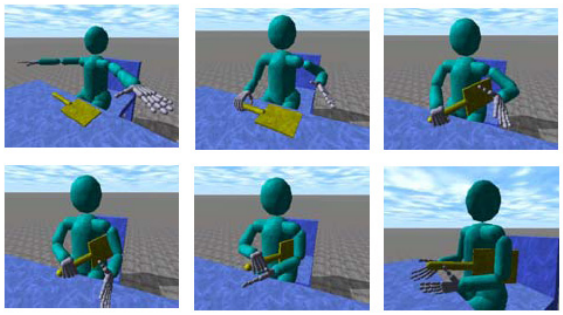

Fig. 4. Keyframes in Under-Arm Grasp Demonstration

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 target 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 [22].

C. 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 both hands to a small set of representative contacts. The geometry of the hand and the object 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 and orientation of the middle knuckle on the palm relative to the object, so that the transformed contact location only gives the general pre-grasp location on the target object. The representative contact(s) are used to locate the hands relative to the object. Once the position of a hand relative to the object is specified, the grasp is essentially specified, since we would in general prefer the fingers to simply wrap around the shape of the object. The actual wrapping of fingers is taken care of by a low-level hand controller during execution of the grasp.

V. Picking a Template Grasp Sequence

When presented with a target 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 z-axis of the object. The automatically-assigned

z-axis of the object is a feature that expresses the alignment of primitives in the object. Distance between two objects is based on a weighted sum of differences between feature values for the two objects being compared at a particular relative rotation. Distances are found for each of the three relative rotations that we will consider while transforming contacts, discussed in section VI; the lowest of the three is considered the final distance between the two. Additionally, template grasps can be flipped from right to left, so that a right-hand grasp turns into a left-hand grasp; this effectively doubles the number of possible template grasps. In this implementation, the weights are hand-tuned.

While this set of features does not take into account fine object features that may be useful for selecting an appropriate grasp, bulk object characteristics appeared to be both more important as well as 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.

VI. Transformations

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 grasps both sets of '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. By equating parts of the template object with parts of the target object and grasping them in the same way throughout the entire grasp sequence, grasp continuity in the adapted grasp sequence is automatically assured.

Fig. 5 shows a box being transformed into a hammer using the above-listed transformations, via three sequences of transformations that will morph one object into another. 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.

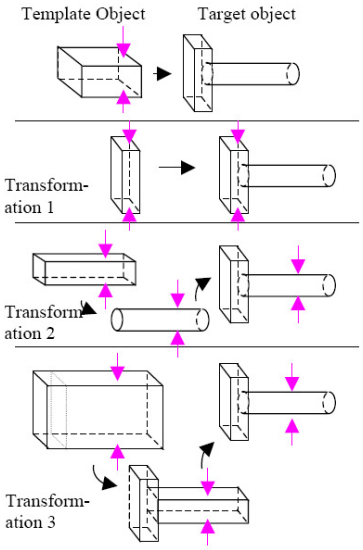

Fig. 5. Transformation sequences from box to hammer

We want to consider more than one possible relative orientation, however. Because our objects consist of primitives in a line with symmetry axes aligned, the most useful rotations are those that cause the axes of both the template and target objects to be aligned (meaning all axes are either parallel or orthogonal to each other). In our current implementation, we consider only three possible relative rotations between the two objects. The first is the alignment that requires the least amount of relative rotation to bring the two objects into alignment from their initial starting positions, and the other two are rotated by ± 90 degrees around the vertical axis. Because our objects are always sitting on a table rather than floating in space, rotating about any other axis (which would result in being able to grab tipped-over objects after seeing a grasp of them upright) often results in collisions between fingers and the table, so we do not use them.

This is not as restrictive as it sounds, however, since the grasps we might desire of tipped-over objects can almost always be obtained through transformations of other template grasps. For example, an object previously grasped with a side grasp that has been tipped over can now be grasped using a demonstrated top grasp.

To transform objects through a sequence of transformations, each transformation is performed in turn. As a shorter way to think about how objects can be transformed, we can consider the perspective that a sequence of transformations is equivalent to matching chunks of an object to chunks of the target object, with each chunk in the old object morphing into the equivalent chunk in the target 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 target object. Thus, merely considering all the different ways of matching chunks between objects covers all the relevant sequences of transformations.

The number of possible transformations scales exponentially with the number of primitives in both template and target object. For only three primitives that are in a line and axis-aligned, we simply compare all 125 possible transformations. For the more general case, with more primitives or with arbitrary orientations/alignments of primitives, we would like to compare the structure of the template object to the structure of the target object to search more likely geometric matches first. If both objects are expressed as primitives in a part hierarchy, then we can use principles from 3-D object recognition to compute a partial distance between two matches, as in [13]. If we wish to limit the number of transformations considered, we can then do so by either setting a cutoff on the number of transformations searched, or by setting a cutoff on the distance between the two matches.

VII. Mapping Contacts Through a Transformation

For each sequence of transformations, we need a method of mapping 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 target object.

One simple method, used in this implemention, 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, or squashes to the closest point on a sphere as if the box's corners were squashed inward. Thus, the positions of contacts are scaled with the boundary of the object, maintaining their relative position with respect to edges and corners. An example of this sort of contact transformation is shown in Fig. 6; the corner contact on the square maps to the closest point on the circle, or to the corner of the stretched-out square.

More formally, if the coordinates of a contact with respect to the center of a chunk of the demonstration object are (p_x, p_y, p_z), and the dimensions of the bounding box of that chunk are (b_x, b_y, b_z), we can find the new coordinates on a target object chunk of dimensions (b'_x, b'_y, b'_z) by shrinking the coordinates by the old dimensions and then scaling them up by the new dimensions. Thus, the new coordinates, $(p'_x, p'_y, p'_z) = (\frac{p_x b'_x}{b_x}, \frac{p_y b'_y}{b_y}, \frac{p_z b'_z}{b_z})$. Since these coordinates may not be on the surface of the target object, we take the closest point on the target object to these coordinates.

VIII. Grasp Candidate Quality Metrics

The process of finding all the possible transformations between template and target object and adapting the contacts using the chosen contact mapping 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

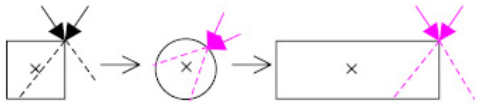

Fig. 6. Mapping Contacts Through a Transformation

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 goals that are kinematically impossible to reach. 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 target objects when overlaid, the distances that the contact points move when mapped, reachability of the goal 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. Alternatively, we could use a more standard grasp quality measure, such as the convex hull L_1metric from [6]. This would require an estimate of the actual contacts that would be made from the chosen pre-grasp locations, which can be made by using collision-checking to close fingers around the object.

Finding the best grasp candidate, with its estimated kinematically feasible contact locations and grasp quality, is as far or further than many grasp planners go. Our new grasp planning approach is polynomial in the number of contacts (due to finding the grasp quality of at least one grasp) and exponential in the number of primitives (due to having to match all possible chunks of one object with all possible chunks of the other). It also requires searching for kinematically feasible arm positions through an optimization process that in reality takes not much more time than basic numerical inverse kinematics.

IX. Finding Collision-Free Keyframes and Trajectories

In order to test our proposed grasps, we must carry out the proposed grasp sequences in simulation. This process involves 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 contacts. 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 shift.

Given a sequence of adjusted keyframes, we can use a probabilistic roadmap to find a collision-free trajectory to traverse the keyframes, as described in [10].

X. Simulating Trajectories

Once a proposed collision-free grasp trajectory is found, we can execute it in simulation to test whether the grasp has any hope of working in the real world.

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. Since the controllers have no special accommodation for sliding along surfaces, grasps that require sliding are usually broken. For example, in the keyframes for the demonstrated under-arm grasp in Fig. 4, the object shifts position considerably while being sandwiched by the arm. This resulted in the failure of all adapted under-arm grasps, since attempts to slide the object result in dropping it.

XI. Results

A. Template Grasps

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. 7.

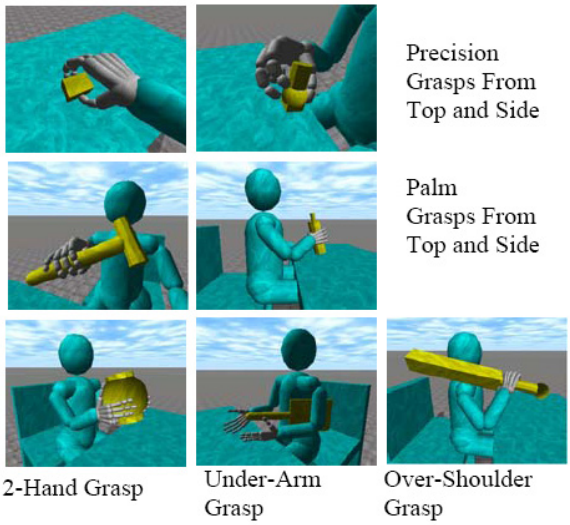

Fig. 7. Template Grasps

B. Successful 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 of these randomly generated objects are shown in Fig. 8.

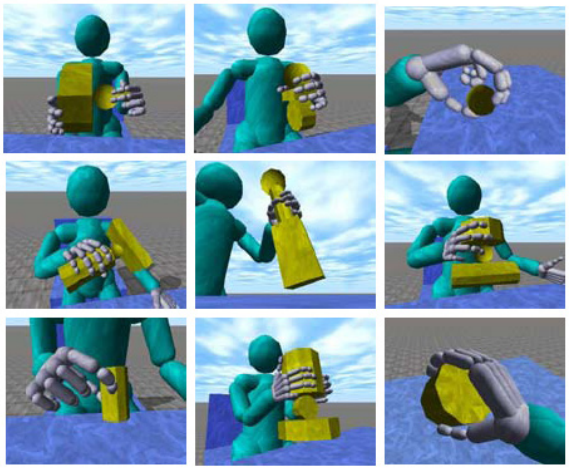

Fig. 8. Successful Grasps of Random Objects

Examples of successful grasps of hand-generated test objects made to look more like real objects are shown in Fig. 1.

C. Failed Grasps

The eight failed grasps from of the set of 100 randomly generated objects are shown in Fig. 9.

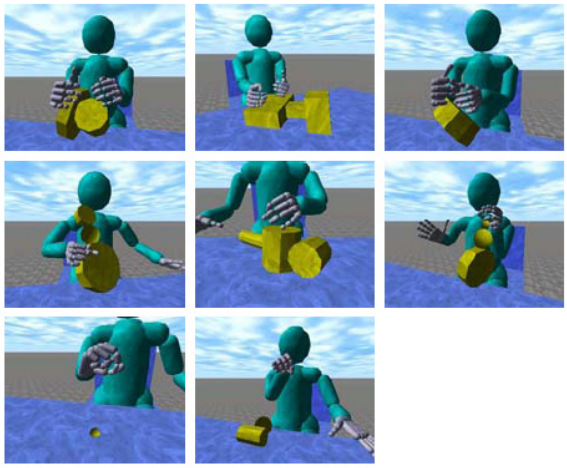

Fig. 9. Failed Grasps of Random Objects

Most of the grasps shown failed because of problems with the grasp controllers, not because of problems with the chosen pre-grasp location. All three grasps in the top row failed because of incorrect finger forces applied in the two-hand controller, as discussed in section X. The first two grasps in the middle row failed because of problems with the thumb controller that caused the thumb joint to go unstable; these problems were largely an artifact of the simulation and have since been resolved. The rightmost grasp in the middle row failed because the grasp selected was an under-arm grasp, which as we mentioned earlier, all resulted in failure. The grasp of the tiny ball in the bottom left failed because it was too tiny for the two-finger grasp from the demonstration. The bottom middle object was dropped because the grasp chosen was barely unreachable by the arm kinematics, causing the thumb to land on a bad location. Our current quality metric was unable to detect such a problem, but a convex hull quality metric would have revealed the problem instantly. With the exception of sliding in under-arm grasps, all of the problems encountered with these objects are easily fixable.

REFERENCES

[1] Bicchi, Antonio. "Hands for Dexterous Manipulation and Robust Grasping: A Difficult Road Toward Simplicity," *IEEE Trans. on R & A*, 16(6), 2000.

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

[3] 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 Intl Conf on Humanoid Robots*, 2000.

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

[5] 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.

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

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

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

[9] Kang S.B. and K. Ikeuchi, "Toward automatic robot instruction for perception - recognizing a grasp from observation," *IEEE Trans. on R & A*, 9, 432-443, 1993.

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

[11] Li, Ying, and Nancy Pollard. "A Shape Matching Algorithm for Synthesizing Humanlike Enveloping Grasps," IEEE-RAS Intl Conf on Humanoid Robots, 2005.

[12] Lozano-Prez, Toms, Joseph L. Jones, Emmanuel Mazer, Patrick A. O'Donnell. *HANDEY: A Robot Task Planner*. MIT Press, 1992.

[13] Medioni, Gerard G. and Alexandre R.J. Francois. "3-D Structures for Generic Object Recognition," *ICPR*, 2000.

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

[15] Ogata, H. and T. Takahashi, "Robotic assembly operation teaching in a virtual environment," *IEEE Trans. on R & A*, 10(3), 391-399, 1994.

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

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

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

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

[20] Rijpkema, H., Girard, M., "Computer Animation of Knowledge-Based Human Grasping," ACM SIGGRAPH, 1991.

[21] Simeon, Theirry, Laumond, J.P., Cortez, J, Sahbani, A. "Manipulation Planning with Probabilistic Roadmaps," *IJRR*, 23(7-8), 729-746, 2004.

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

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

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

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