Grasp Planning in Complex Scenes

Dmitry Berenson* Rosen Diankov* Koichi Nishiwaki† Satoshi Kagami† James Kuffner*†

*The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA, 15213, USA
{dberenso, rdiankov, kuffner}@cs.cmu.edu

†Digital Human Research Center
National Institute of Advanced Industrial Science and Technology
2-41-6 Aomi, Koto-ku, Tokyo, Japan 135-0064
{k.nishiwaki, s.kagami}@dh.aist.go.jp

***Abstract*— This paper combines grasp analysis and manipulation planning techniques to perform fast grasp planning in complex scenes. In much previous work on grasping, the object being grasped is assumed to be the only object in the environment. Hence the grasp quality metrics and grasping strategies developed do not perform well when the object is close to obstacles and many good grasps are infeasible. We introduce a framework for finding valid grasps in cluttered environments that combines a grasp quality metric for the object with information about the local environment around the object and information about the robot's kinematics. We encode these factors in a *grasp-scoring function* which we use to rank a precomputed set of grasps in terms of their appropriateness for a given scene. We show that this ranking is essential for efficient grasp selection and present experiments in simulation and on the HRP2 robot.**

I. Introduction

In recent years, many researchers in Humanoid Robotics have focused on topics in autonomous manipulation. Research in manipulation has been done on humanoid platforms such as the HRP2 [16], ARMAR [17], the NASA Robonaut [14], Justin [15], Dexter [13], and Domo [18]. However, much humanoid grasping still relies on tele-operation or hand-scripted grasps.

Many researchers have approached the problem of grasping from a machine learning perspective where the goal is to find grasps of novel objects using information about grasps of already-known objects [11] [12]. While these approaches are promising, it is difficult to find a general parameterization of objects that preserves sufficient information for grasping. Also, because robotic manipulators usually have very different kinematics and sensing capabilities from those of human hands, it is difficult to successfully apply imitation-learning algorithms that extract information from humans [10] to robotic grasping.

Another main area of grasping research focuses on finding a placement of contact points on an object's surface to maximize a certain grasp metric [6] [7]; however, it is difficult to match those contact points to a feasible configuration of a humanoid's manipulator that is collision-free in a given environment and reachable by the robot.

Regardless of the method used for grasp selection, much previous research has focused on finding grasps for the object when it is alone in the environment. Furthermore, the manipulator is often assumed to be disembodied when approaching the object [1] so the kinematics of the robot are not taken into account. While the above assumptions may be valid in certain situations, they are certainly not true for cluttered and constrained environments. Thus we approach grasp selection and manipulation planning from a holistic point of view. The goal is not only to select a grasp that is stable for a given object, but also to ensure it is feasible.

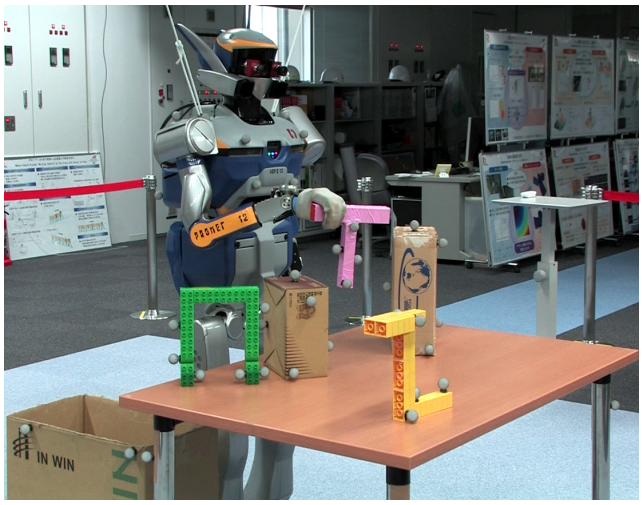

Fig. 1. The HRP2 lifting an object after executing one of the top-ranked grasps evaluated by the *grasp-scoring function*.

The remainder of the paper is outlined as follows: In section II, we give an outline of our overall framework. In section III, we describe the steps needed to compute our grasp-scoring function. In section IV, we show that our method greatly outperforms the naive approach to grasp selection and show results in simulation and on the HRP2.

II. Grasp Planning Framework

Our framework is similar to many papers in that we use force-closure [5] to evaluate good grasps when the object is alone in the environment. Like [17], we sample our grasp-parameter space to find a set of successful grasps for each object of interest. However, once we have computed a set of valid grasps for the given object, we are presented with a problem: Out of the potentially thousands of precomputed

grasps, *which one do we choose for a given environment?* Many grasps can be infeasible because of collisions with environment obstacles, still more can be unreachable because of the kinematics of the robot arm. The naive approach is to keep trying grasps in an arbitrary order and take the first one that is collision-free and reachable. Because of the potentially huge number of grasps in our grasp set, this approach is extremely time-consuming. Of course it is possible to prune the grasp set; however, there is always the risk that pruning can eliminate the only feasible grasp in the environment. It is also possible to apply manipulation planning techniques from [2] [3] [4] to move obstacles out of the way. However, to pick which obstacles to move, we must decide if those obstacles are graspable and the problem of grasp selection re-emerges.

Instead we propose a more intelligent framework for grasp selection that combines the force-closure scores of our grasp set with the features of the object's local environment and features of the robot kinematics to produce an overall *grasp-scoring function*. We use this function to evaluate each grasp and compute a plan using Bidirectional RRTs [8] that takes the arm[1] to the desired position and closes the fingers appropriately.

Our method of grasp planning consists of two main steps: a precomputation step and an online computation step (see Figure 2). In the *precomputation* step, we use a geometric model of our manipulator and the object we wish to grasp to build a set of feasible grasps in terms of force-closure. In the *online computation* step, we compute the score of each grasp for the given environment using our *Grasp-scoring Function*. The grasps are ranked in order of the scores assigned by the *Grasp-scoring Function* and validated using collision checking and Inverse Kinematics(IK) algorithms. Once a feasible grasp is found, we plan a reaching motion for the arm of the robot to reach and grasp the object using Bidirectional RRTs. The process is detailed below:

Precomputation: First, sample a set of grasp parameters in the grasp-parameter space. Second, compute the final manipulator pose by executing a grasping policy on those parameters for the given manipulator and object. Third, Compute the force-closure scores of all grasps and store the ones that have force closure in the grasp set.

On-line Computation: Fourth, evaluate the grasp set using a *grasp-scoring function* that takes into account the force closure score, the object's local environment, and the kinematics of the robot. Use the scores assigned by this function to rank the set of grasps. Fifth, test the grasps in order of their rank for collisions in the environment and reachability. Sixth, plan an arm trajectory to achieve the first valid grasp, if the plan fails, try the next valid grasp and so on.

A. Defining a Grasp

Choosing a parameterization for grasping policies is key to grasp planning. Since one goal is to generalize beyond a given manipulator, the definition of parameters must be as broad as possible, yet the definition must also include manipulator-specific information to take advantage of the unique capabilities of a given manipulator. Thus we define the following general parameters:

- $\mathbf{P}_d$, the direction of approach of the hand (a 3D unit vector)
- $\mathbf{P}_t$, the 3D point on the surface of the object that the hand is approaching
- $\mathbf{P}_r$, the roll of the hand about the approach direction
- $\mathbf{P}_p$, parameter(s) defining the preshape[2] of the hand

In addition to the above, other hand-specific parameters can be included in $\mathbf{P}$. Also define O as the object to be grasped and E as the current environment.

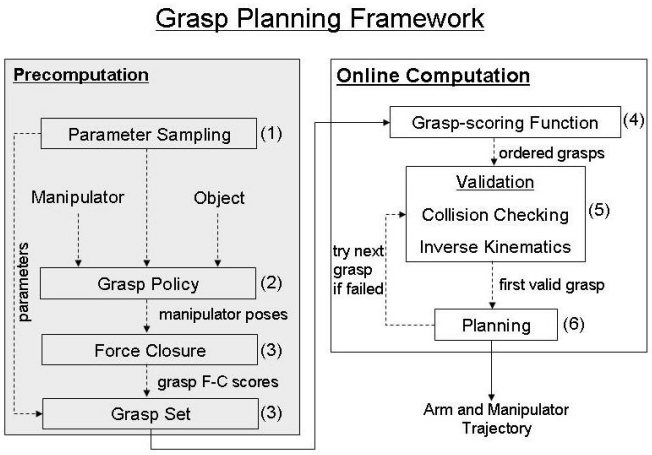

Fig. 2. The proposed grasp planning framework. The precomputation section need only be computed once for each object-manipulator pair. Once the grasp set is built by precomputation, the online part of the framework sorts grasps using the grasp-scoring function and plans a trajectory to get to the first valid grasp.

B. Grasping Policy

The grasping policy is a function that maps a set of parameters $\mathbf{P}$ to a final grasping pose of the manipulator. A manipulator pose is defined by the manipulator's translation, orientation, and joint angles. In the grasping policy of [1], the manipulator first starts with a specified preshape and approaches the object's center of mass from a given direction; the parameters are the approach direction, the roll around the direction, and the preshape. Once any part of the manipulator hits the object, manipulator joints close until each finger reaches collision.

We use a similar but slightly more general policy in this paper. From empirical experiments with several robotic hands, the most successful grasps usually have the manipulator's palm facing the object's surface normal at the palm contact points. By using the parameterization of [1], the manipulator's palm will rarely face the surface normal unless the object is a sphere, and many good grasps will be missed. Instead of approaching the center of the object, our grasping policy approaches a target point $\mathbf{P}_t$ (Figure 3) with the palm

[1]"Arm" is used to loosely define all the joints from the base of the robot to the manipulator that affect the manipulator's position. Joints that control the translation and rotation of the robot are usually ignored.

[2]The preshape is the initial joint values of the manipulator.

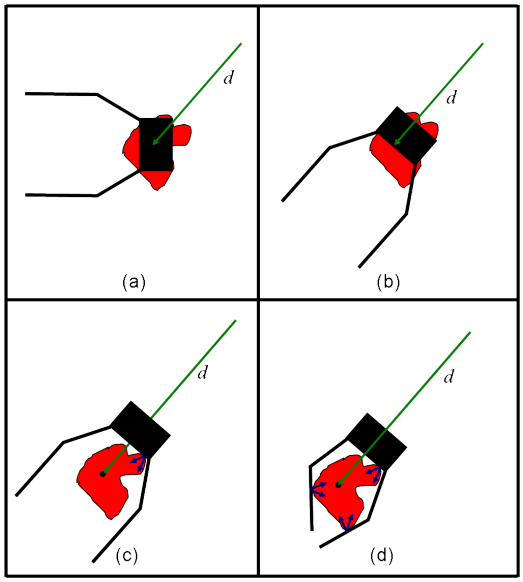

Fig. 3. The grasping policy that is used to evaluate grasp quality. (a) The manipulator is placed inside the object at its target grasp point (specified by $\mathbf{P}_t$). (b) The manipulator is rotated so that its palm normal faces along $\mathbf{P}_d$. (c) The manipulator is moved back along $-\mathbf{P}_d$ until it is just out of collision. (d) The fingers are wrapped around the object until every finger is in collision or has hit its joint limits. The contact wrenches (blue arrows) are used to compute force closure.

normal oriented towards $\mathbf{P}_d$ and a roll angle $\mathbf{P}_r$ around the approach direction. Although the dimension of the grasp policy seems to have increased significantly from that of [1], we constrain $\mathbf{P}_t$ to always lie on the surface of the object O and $\mathbf{P}_d$ so that it always points towards the normal of the object's surface at $\mathbf{P}_t$. Compared to using [1], the percentage of total force-closure grasps usually increases by four times for the objects we use in our experiments while the dimension of the grasp parameters only increases by one.

C. Precomputing a Valid Grasp Set

Given the above grasping policy, we can generate $\mathbf{P}$ and determine its force-closure quality for a given O. Though it is desirable to generalize across objects, our approach currently does not possess this capability, thus a set of grasps is stored for every O of interest. Because each grasp set can be stored in a small file on disk, having such a set for every object of interest is reasonable considering the multi-terrabyte hard-drives available today. Also, storing grasps along with their grasp-quality saves computation time because we never need to re-evaluate force-closure online.

However, we must be careful to generate $\mathbf{P}$ values that are likely to yield force-closure for the object, otherwise we will waste time computing grasps that have no chance of achieving force-closure. Thus we sample our grasp-parameter space as follows:

We start by sampling the surface of O using six fine-resolution grids, one on each side of the object (Figure 4). A ray is cast from every point in the grids and the location where it first intersects O becomes a sample of the object's surface. Let the number of rays hitting the surface be N. Each sample is treated as the target point $\mathbf{P}_t$. For every target

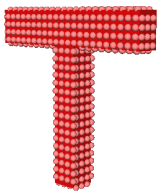

Fig. 4. An object (red) with sampled surface points (pink). Each surface point can be transformed to a set of grasps by using the point's position as $\mathbf{P}_t$ and the surface normal at that point as $\mathbf{P}_d$.

point $\mathbf{P}_t$, define the approach directions $\mathbf{P}_d$ as the negative of O's surface normal at $\mathbf{P}_t$. The parameters left to define are the roll $\mathbf{P}_r$ and the preshapes $\mathbf{P}_p$. Roll is discretized into R bins in $[0, 2\pi]$. The preshapes of the manipulator $\mathbf{P}_p$ are specific to the given manipulator. Let the number of possible preshapes be M. Then the total number of samples used to build the force closure table is NRM. We run the grasping policy for each $\mathbf{P}$ and record the force-closure values in a table. Because this procedure is done offline, the computation time is not an issue.

D. Grasp-scoring Function

Much previous research on grasping has focused on selecting grasps in terms of force-closure, but force-closure alone is not sufficient when choosing a grasp for an object in a cluttered environment, especially when the manipulator is attached to an arm with its own unique kinematic structure. Taking into account the force closure value of a grasp guarantees successful manipulation only if both the object and manipulator are freely floating in space, but this is never the case. If there is a cup on a table and the environment is not taken into account, then testing grasps that approach from under the table is simply a waste of processing time. Such grasps would never be tested if the environment is considered. Thus, taking the environment into account can filter out those grasps which are likely to collide with environment obstacles, and therefore save time spent on needless testing. Furthermore, considering the position of the robot can filter out grasps that are unreachable and again save time in validation and planning. Thus the goal of the grasp-scoring function is to take into account all important information to score grasps in terms of likelihood of success. Section III describes the computation of the scores assigned by this function in detail.

It is essential that the grasp-scoring function be computed quickly because it must be evaluated online for the current environment. Since fully simulating a grasp is time consuming, the grasp-scoring function must rely on approximations and therefore grasps must be validated in simulation even if they recieve a high score. As shown in section IV, using the grasp-scoring function greatly reduces the amount of validation necessary before a successful grasp is found.

E. Validation and Planning

Since the *grasp-scoring function* cannot gaurantee that a grasp will be successful, grasps must be validated one-by-one

in the current environment by running the grasping policy. If there is no collision with environment obstacles during the execution of the grasping policy, we evaluate the IK equations of the arm to see if the grasp is reachable and check if the IK solution[3] is in collision. If the final pose of the arm is collision-free, we plan a path using Bidirectional RRTs to reach the desired manipulator position and close the fingers. The planner's configuration space consists of the arm and manipulator joints of the robot. The planner uses the IK solution values for the arm and final manipulator joint values from the grasping policy to set the configuration space goal. If any of the above steps fail, we try the next grasp in the set, as ranked by the grasp-scoring function.

III. Computing the Grasp-scoring Function

The goal of the *Grasp-scoring function* is to predict the result of a given grasp of a given object in a given environment. Let $G(O, E, \mathbf{P})$ be the score of the grasp defined by the grasp policy parameters $\mathbf{P}$ performed on object O in environment E. O can change depending on which object is being considered for grasping and E changes whenever something in the environment is moved. G is composed of three parts:

- $G_q(\mathbf{P})$, the force-closure score of $\mathbf{P}$
- $G_b(E,\mathbf{P})$, the robot-relative position score
- $G_e(O, E,\mathbf{P})$, the environment clearance score

Once the three scores are computed, they are combined into a final score as shown in the equation below.

$$G(O, E, \mathbf{P}) = e^{(c_1 G_q)} e^{(c_2 G_b)} e^{(c_3 G_e)} \quad (1)$$

where c_1, c_2, and c_3 are coefficients that determine the relative importance of the three criteria in computing the overall score.

We set G_q by looking it up in our precomputed grasp set (Section II-C).

$G_b(E,\mathbf{P})$ takes into consideration the position of the robot in the given environment. Since the robot's base is fixed, we use standard IK algorithms to check if a given grasp is feasible. However, testing IK feasibility for all grasps in the grasp set is time-consuming. As a simple approximation, we set G_b to be the cosine of the angle between the manipulator approach direction and the vector from the robot to the target object. This ensures that grasps where the palm faces away from the robot are preferred over grasps where the palm faces the robot.

$G_e(O, E,\mathbf{P})$ allows us to consider the environment around the object when computing the grasp score. We calculate the clearance along the approach direction $\mathbf{P}_d$ at the point $\mathbf{P}_t$. If that clearance is small, the manipulator is likely to collide when executing the grasp specified by $\mathbf{P}$. When the clearance is high, there is a greater chance that the grasp will be collision-free. We detail the computation of G_e below.

Figure 7 shows how the various scores are combined into a total score for a 'T' shaped object in a certain scene.

[3]If IK equations exist for the arm of the robot, this step is trivial. If not, there exist many numerical methods for IK.

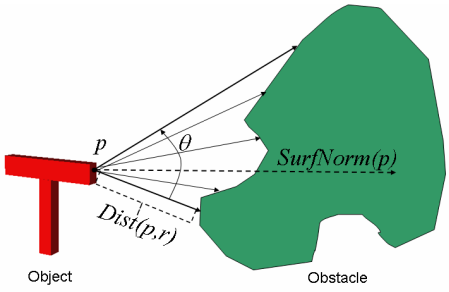

Fig. 5. Computing the minimum distance value for one point in the distance map. The red object is the object we wish to grasp, the green one is an obstacle in the environment. θ is the width of the cone whose tip is at $\mathbf{P}$ and whose alignment is the surface normal at $\mathbf{P}$. $Dist(p, r)$ is the minimum distance as evaluate over all rays, r, in the cone.

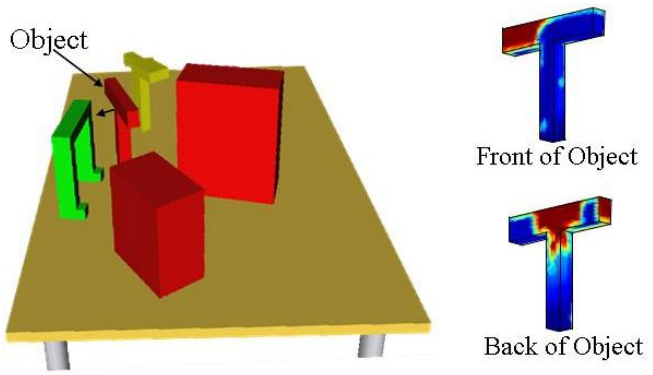

Fig. 6. Distance map for the 'T'-shaped object in the scene shown. Distances range from dark red (high clearance) to blue (low clearance). The arrow denotes the front of the object.

Note that when O is rotated or translated from it's identity transform, all grasps and scoring function computations are transformed accordingly.

A. Computing the Environment Clearance Score

When computing the environment clearance score, we assume that grasps that approach the object along directions with high clearance to obstacles are more likely to succeed. Although such directions are not always guaranteed to be feasible for the manipulator, it is more likely that a successful grasp approaches from high-clearance directions than from directions cluttered with obstacles. The *distance map*, $\mathcal{D}_{(O,E)}(p, \theta)$, gives the distance to the nearest obstacle within a cone of angle θ whose tip is at the surface point p and whose alignment is the surface normal of p (see Figure 5). A cone is used so that each direction summarizes a bigger region of the environment; if $\mathcal{D}_{(O,E)}(p, \theta)$ has some value, this implies that the entire cone is free of obstacles up to that value in meters. The wider the cone angle, the more conservative the distance map becomes in estimating free regions. Let $Dist(p, d)$ be the distance to the nearest point the ray starting at position $\mathbf{P}$ with direction d hits. The distance map is defined as:

$$\mathcal{D}_{(O,E)}(p, \theta) = \min_{r : r \cdot SurfNorm(p) \leq \cos(\theta)} Dist(p, r). \quad (2)$$

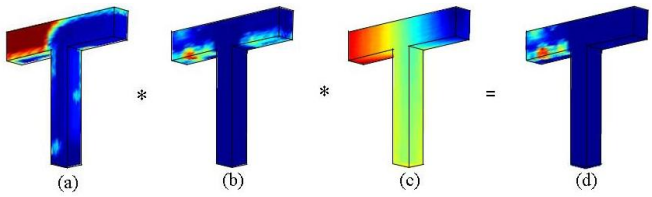

Fig. 7. (a) Environment Clearance Score ($G_e(O, E, P)$), (b) Grasp Quality ($G_q(P)$) of grasps with $\mathbf{P}_r = \frac{3\pi}{2}$ and $\mathbf{P}_p = 0$ for the HRP2 manipulator. (c) Robot-relative position score ($G_b(E, P)$) (d) Total combined score ($G(O, E, \mathbf{P})$).

We use a θ of 15 degrees in our experiments.

To generate the surface points needed for the distance map, we sample the surface of O as described in Section II-C but at a finer resolution. Because ray collision checking is very fast, we are able to generate the distance map for scenes like that in Figure 6 in about one second on a 3Ghz, dual-core PC.

Once we have computed this high-resolution distance map, we look up the point $\mathbf{P}_t$ in the distance map using a k-nearest-neighbor algorithm and take the weighted-average of the distances of the neighbors of $\mathbf{P}_t$ to determine $G_e(O, E, P)$.

Figure 6 shows a scene with the target object and the computed distance map for its position.

IV. Results

To gauge the usefullness of the grasp-scoring function, we perfomed a benchmark experiment on two robots in simulation. In this experiment, we compare the number of tests necessary before finding a valid grasp when sorting the grasp set by the grasp-scoring function to the number of tests necessary when randomly ordering the grasp set. We also describe an implemenation of the framework on an HRP2 robot at the Digital Human Research Center and show it performing several complex tasks.

A. Benchmark Experiment

In order to benchmark our algorithm's effectiveness, we compare it to a more naive approach to grasp selection. In the naive approach, a table of valid grasps is created in exactly the same manner as in our approach. However, instead of sorting the grasp set, the grasps are tested on the object in the given scene in a random order. The question we seek to answer is: how many grasps must we test before finding a successful one when using the *grasp-scoring function* vs. the naive method?

We compare the statistics for 50 randomly generated scenes. When using the naive approach, we re-run the validation 10 times for each scene, each time re-ordering the set randomly. We record the indices of the first successful grasp of each run and compute their mean. This mean is compared to the index of the first successful grasp when using the *grasp-scoring function* to sort the grasp set. No re-running is necessary when using the grasp-scoring function because our algorithm does not use any random variables.

Generating the random scenes is done as follows: The position of the robot is always fixed. The object to be grasped is placed in a region that is known to be reachable by the robot[4]. Obstacles are then placed randomly around the object to be grasped in a circle with 20cm radius and rotated randomly about the vertical axis. No collisions between obstacles are allowed. See Figure 8 for several random scenes.

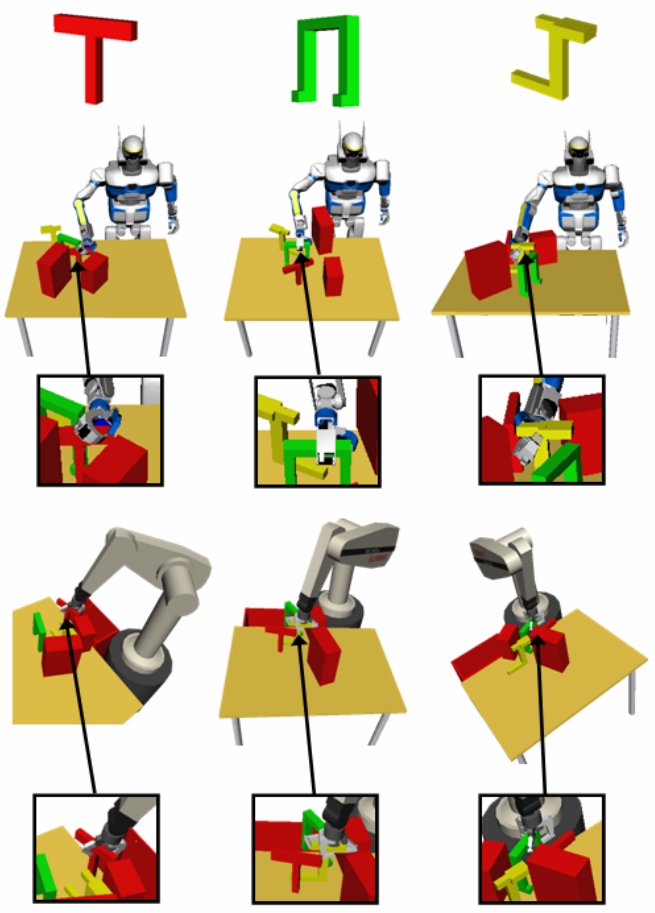

Fig. 8. Objects on which we tested our algorithm (above) and the HRP2 and Puma grasping those objects in random scenes. The objects are referred to as 1, 2, and 3, numbered from left to right. All grasps shown were ranked in the top 50 grasps by our grasp-scoring function for their respective scenes.

This experiment is run for three different objects and two different robots in simulation, see Figure 8. The robots used where an HRP2 robot, with the manipulator shown in Figure 9 and a Puma 560 robot with a Barrett hand. The two robots have very different kinematics and reachability ranges. The manipulators are also very different, the HRP2's manipulator is designed to wrap-around objects, while the Barrett hand is meant for more precision and dexterous grasping. We run experiments on these two robots to show that our framework, and especially the grasp-scoring function, is general enough to be applied to any reasonable robot-manipulator combination.

For both robots and all objects, the grasp-scoring function clearly outperforms the naive approach to grasp selection,

[4]Doing otherwise would result in many IK failures that would tell us nothing about the efficacity of sorting by the $G(O, E, \mathbf{P})$ scores

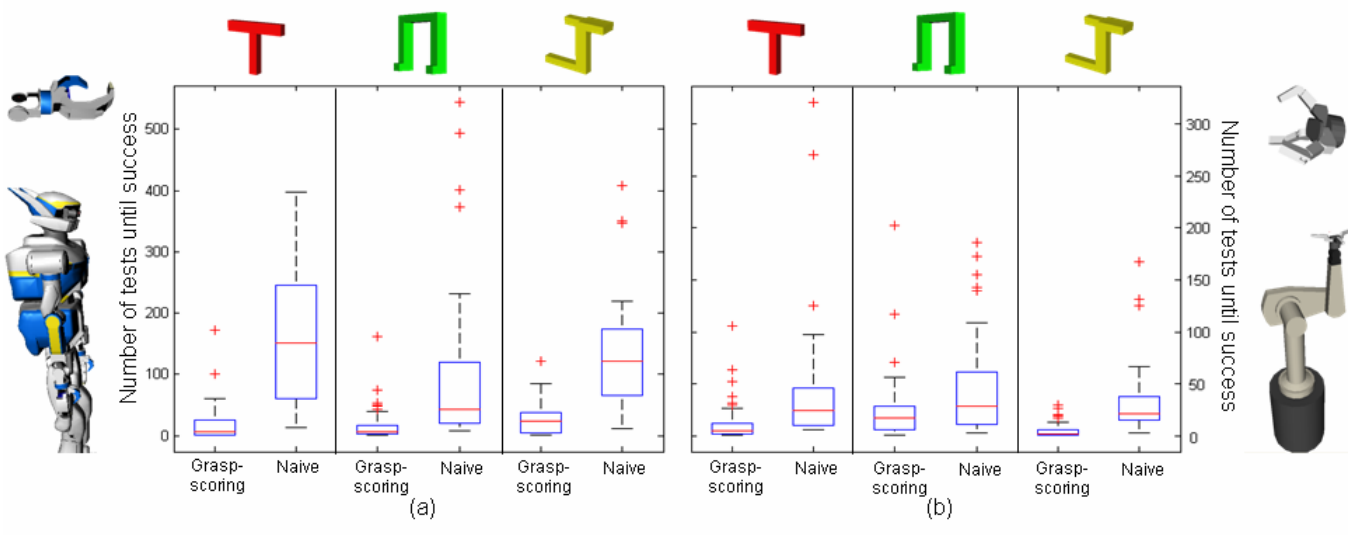

Fig. 9. Box plot of the statistics for the benchmark experiment on the three objects for the HRP2 and Puma robots. Each object shown at the top was placed in 50 random scenes. The vertical axis is the number of grasps tried before a successful one was found. The red line in each box plot represents the number of tests necessary averaged over the 50 scenes. Lower is better. (a) Statistics when using the HRP2. (b) Statistics when using the Puma+Barrett hand robot.

see Figure 9. For the HRP2, grasp-scoring outperformed the naive method in 98%, 100%, and 98% of scenes for objects 1, 2, and 3, respectively. For each scene, grasp scoring performed 57.06, 11.51, and 10.59 times better on average than the naive approach for objects 1, 2 and 3, respectively.

For the Puma robot, grasp-scoring outperformed the naive method in 96%, 72%, and 98% of scenes. Grasp scoring performed 6.861, 3.101, and 12.54 times better on average than the naive approach. The results for the Puma robot are less impressive because the Puma has a far larger reachability, thus many more grasps have an IK solution and there is a greater chance that a random grasp is reachable.

B. Experiments with the real HRP2

To evaluate our method in a real environment with a real robot, we ran several experiments on the HRP2. We used a motion capture system to get the position of the robot and the objects in our scene. The robot's base was fixed in these experiments.

The first task was for the HRP2 to reach and grasp objects 1 and 2 and place them into a "trash can." We required all trajectories to be collision-free and required each object to stay within the robot's grasp during the trajectory to reach the trash can. The experiment was meant to simulate a clean-up task, where some objects in the environment must be thrown away, which is one of the target applications for many humanoid robots. Several frames from a video showing the robot performing this task for object 1 can be seen in Figure 10. Since we compute the grasp-scoring function online, the robot is able to compensate for changing scenes and the addition/removal of obstacles and objects to grasp. All grasps and trajectories required for this proceedure were automatically generated online using our grasp-planning framework. Videos of the robot performing the clean-up task in multiple scenes can be seen in the video corresponding to this paper.

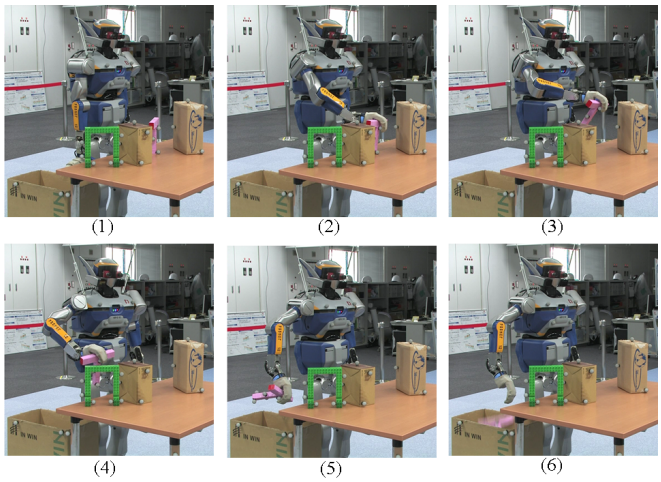

Fig. 10. The HRP2 reaching, grasping, and dropping object 1 into the trash can. (1) The initial state of the environment. (2) Grasping the object. (3) Lifting the object. (4)-(5) Moving the object to the trash can. (6) Dropping the object into the trash can.

The second task was for the HRP2 to pick up object 2 and place it upside-down on the other side of the table. Because of the reachability constraints of the robot, this task could not be accomplished using only one arm, so re-grasping the object was necessary. To accomplish this task, we specified that the robot must first grasp the object with its right hand, move it into an intermediate position, grasp the object with its left hand, release the object with the right hand, and move the object into the destination position with its left hand. See Figure 11 for snapshots from the execution of this proceedure. Again, all grasps and trajectories required for this proceedure were automatically generated online using our grasp-planning framework. A video showing the re-grasping sequence can be seen in the video corresponding to this paper.

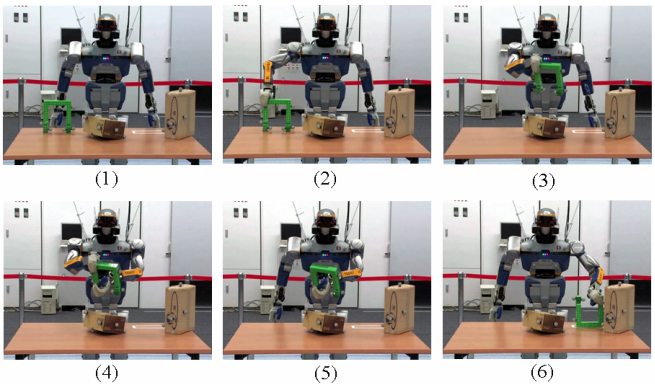

Fig. 11. The HRP2 picking up, regrasping, and putting down object 2. The task is to place the object upside down into the white rectangle. (1) The initial state of the environment. (2) Grasping the object with the right hand. (3) Moving the object to an intermediate position. (4) Grasping the object with the left hand. (5) Releasing the object with the right hand. (6) Placing the object into the goal position with the left hand.

V. Conclusion

This paper presents a general framework for grasping which takes into account the kinematics of the robot, the environment around the object being grasped, and the grasp's force-closure quality. By precomputing a set of grasps offline and efficiently computing a grasp-scoring function online to rank the grasps, we are able to quickly find stable, collision-free, reachable grasps in cluttered environments. We show that our framework greatly outperforms a more naive approach and that it can be applied to two different robots with very different manipulators. We also describe an implementation of the framework on the HRP2, where we are able to perform complex tasks in a dynamic environment.

A. Future Work

In future work, we plan to extend our framework to various manipulators and robots. We plan to improve our methods for measuring grasp quality to take into account task-specific constraints as well as improving our sampling strategy for generating grasp tables. We would also like to explore the integration of our framework with planning for a mobile base.

References

[1] Pelossof, R., Miller, A., Allen, P., Jebara, T., "An SVM Learning Approach to Robotic Grasping," *ICRA*, 2004.
[2] Stilman, M., Schamburek, J., Kuffner, J., and Asfour, T., "Manipulation Planning Among Movable Obstacles," *ICRA*, 2007.
[3] Okada, K., Haneda, A., Nakai, H., Inaba, M., and Inoue, H., "Environment Manipulation Planner for Humanoid Robots Using Task Graph That Generates Action Sequence," *IROS*, 2004.
[4] Cambon, S., Gravot, F., and Alami, R., "A robot task planner that merges symbolic and geometric reasonning," *ECAI*, 2004.
[5] Mason, M.T., "Mechanics of Robotic Manipulation," Cambridge, MA: MIT Press, August 2001.
[6] Li, Z., and Sastry, S., "Task oriented optimal grasping by multifingered robot hands," *IEEE Trans. on Robotics and Automation*, 1988.
[7] Zhu, X., Wang, J., "Synthesis of Force-Closure Grasps on 3-D Objects Based on the Q Distance," *IEEE Trans. on Robotics and Automation*, 2003.
[8] LaValle, S., and Kuffner, J., "Rapidly exploring random trees: Progress and prospects," *Workshop on the Algorithmic Foundations of Robotics*, 2000.
[9] Hirano, Y., Kitahama, K., and Yoshizawa, S., "Image-based Object Recognition and Dexterous Hand/Arm Motion Planning Using RRTs for Grasping in Cluttered Scene," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.
[10] Li Y., Fu J., and Pollard N., "Data driven grasp synthesis using shape matching and task-based pruning," *IEEE Transactions on Visualization and Computer Graphics*, 2007.
[11] Saxena, A., Driemeyer, J., Kearns, J., and Ng, A., "Robotic Grasping of Novel Objects," *NIPS*, 2007.
[12] Hsiao K., and Lozano-Perez, T., "Imitation Learning of Whole-Body Grasps," *Proc. RSS Workshop: Manipulation for Human Environments*, 2006.
[13] Platt, R., "Learning and Generalizing Control Based Grasping and Manipulation Skills," *PhD Dissertation, Department of Computer Science, University of Massachusetts Amherst*, 2006.
[14] Martin, T., Ambrose, R., Diftler, M., Platt, R., Jr., and Butzer, M., "Tactile gloves for autonomous grasping with the NASA/DARPA Robonaut" *ICRA*, 2004.
[15] Wimbock, T., Ott, C., Hirzinger, and Gerd., "Impedance Behaviors for Two-handed Manipulation: Design and Experiments" *ICRA*, 2007.
[16] Okada, K., Ogura, T., Haneda, A., Fujimoto, J., Gravot, F., and Inaba, M., "Humanoid motion generation system on HRP2-JSK for daily life environment," *IEEE International Conference Mechatronics and Automation*, 2005.
[17] Morales, A., Asfour, T., Azad, P., Knoop, S., and Dillmann, R., "Integrated Grasp Planning and Visual Object Localization For a Humanoid Robot with Five-Fingered Hands" *IROS* 2006.
[18] Edsinger-Gonzalez, A., and Webber, J., "Domo: a force sensing humanoid robot for manipulation research," *IEEE International Conference on Humanoid Robots*, 2004.