Manipulation Planning with Caging Grasps

Rosen Diankov* Siddhartha S. Srinivasa† Dave Ferguson† James Kuffner*

*The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
{ridankov, kuffner}@cs.cmu.edu

†Intel Research Pittsburgh
4720 Forbes Ave.
Pittsburgh, PA 15213, USA
{siddhartha.srinivasa, dave.ferguson}@intel.com

***Abstract*— We present a novel motion planning algorithm for performing constrained tasks such as opening doors and drawers by robots such as humanoid robots or mobile manipulators. Previous work on constrained manipulation transfers rigid constraints imposed by the target object motion directly into the robot configuration space. This often unnecessarily restricts the allowable robot motion, which can prevent the robot from performing even simple tasks, particularly if the robot has limited reachability or low number of joints. Our method computes "caging grasps" specific to the object and uses efficient search algorithms to produce motion plans that satisfy the task constraints. The major advantages of our technique significantly increase the range of possible motions of the robot by not having to enforce rigid constraints between the end-effector and the target object. We illustrate our approach with experimental results and examples running on two robot platforms.**

I. Introduction

In this paper, we address the problem of a robot manipulating an object whose motion in the world is constrained. Examples of this interaction include turning a crank or handle, and opening or closing a door, drawer, or cabinet. Previous work on constrained manipulation typically utilize some form of compliance control (e.g. [1]), in which constraints on the object are transformed into *task constraints* on the end-effector of the robot. Constraints are then enforced by maintaining a fixed relative position and orientation (i.e. a rigid grasp) between the object and end-effector. Unfortunately, for many robots, such rigid grasp constraints can be overly restrictive. For example, the opening of a door using a doorknob with one rotational degree of freedom (DOF), imposes five constraints on the robot end-effector if only rigid grasps between the doorknob and end-effector are considered.

The key insight obtained through the experiments presented in this paper, is that for a large class of constrained tasks, the robot end-effector does not have to be rigidly attached to the object throughout the entire motion (Fig.1). In fact, as long as the object is *caged* [2], [3] by the end-effector, moving the end-effector can produce a corresponding motion of the object. In other words, there exists a *grasp space* the end-effector can reside in such that the target object desired motion can be achieved, while providing the robot a greatly expanded range of allowable motion.

Relaxing task constraints through caging grasps has enabled real-world implementations of constrained task execution using low DOF robots. We present experimental results on the six-DOF Manus Assistive Robot Service Manipulator [4] and the seven-DOF Barrett WAM [5], involving the tasks of autonomously pulling doors and cabinets open at arbitrary placements of the robot base. We compare our caged grasp planning approach to a traditional planner that enforces rigid task constraints. The results indicate that relaxing task constraints through caging grasps provide a much a greater motion envelope for the robot as well as versatility in base placement. This expanded range of allowable motions of the robot directly results in: 1) improvements in the efficiency and the success rate of planning for a variety of constrained tasks; 2) greater success in executing the desired motion and achieving the final object goal state.

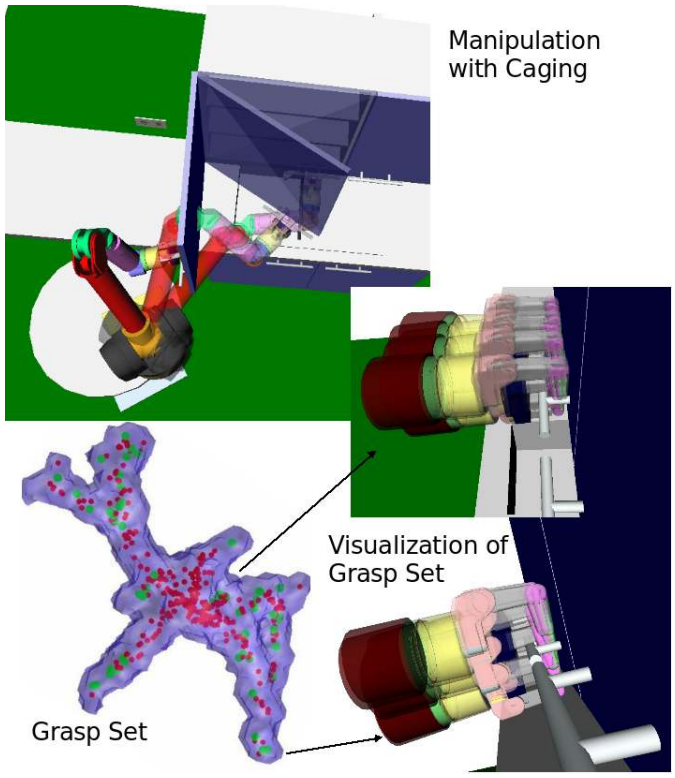

Fig. 1. A robot hand opening a cupboard by caging the handle. Space of all possible caging grasps (blue) is sampled (red) along with the contact grasps (green).

The expanded range of motion comes at the cost of algorithmic complexity. In the absence of a rigid grasp, care must

be taken to ensure that the object does not slip out of the robot end-effector. Of greater concern is the fact that there no longer exists a one-to-one mapping from robot motion to object motion: since the object is loosely caged, there can exist end-effector motions that produce no object motion, and object motion without explicit end-effector motion. The planner proposed in this paper uses a remarkably simple yet effective technique to narrow the grasp set choices as it moves towards the goal state. The planner works produces arm motions that have a high probability of accomplishing the task regardless of the uncertainty in the object motion.

II. Related work

Our work builds upon related work in two areas of manipulation: caging, and task constrained manipulation.

Early work on caging [6] considered the problem of designing algorithms for capturing a polytope using a given number of points. Since then, there have been several applications to cooperative manipulation as well as grasping. Pereira, Kumar and Campos [7] proposed decentralized algorithms for planar manipulation via caging using multiple robots pushing the object. Rimon and Blake [2], [3] viewed caging as an intermediate step to immobilizing an object and computed caging sets that would lead to a pre-specified immobilization grasp. Sudsang, Ponce, and Srinivasa [8] introduced a more relaxed notion of capture regions, placing fingers where the object could be prevented from escaping to infinity. A state-of-the-art cage synthesis algorithm and survey of recent results in caging may be found in Vahedi and Van der Stappen [9].

One of the first formulations of task-constrained manipulation was provided by Mason [1] who observed that motion along task constraints which produced configuration-space surfaces or *C-surfaces* required the combination of position control to move along the C-surface, and force control to guarantee contact with the surface, which he termed compliant motion. He proposed a formalism that combined the natural constraints presented by the task and the desired goal trajectory to produce control policies in terms of artificial constraints. There has been a vast amount of literature following this work [10], [11], [12], [13].

Usually solving a task constrained problem is tightly coupled with simultaneously solving the compliant control and visual servoing problems. [14] implement a behavioral module that scripts the general task of opening a door while being compliant to unknown variables at run-time like direction to open the door and turn the handle. [15], [16] propose a framework to simultaneously solve the task by controlling forces and velocities through a visual servo loop.

Perhaps the work that is closest in spirit to ours is that of Prats *et. al.* [17] who allow the end-effector to move freely along certain directions during manipulations. One limiting factor is that these directions are carefully chosen by hand to ensure that they do not affect the overall task. In fact, end-effectors do not always have to cage the object; as long as the target object moves in the desired direction, just considering pushing can also increase the free space of the robot [18]. We believe our work is a generalization of the above ideas: instead of specifically parameterizing the relationship between the end-effector and the object using simple rules, we automatically generate a data-driven representation of this relationship: the caging manifold.

The main contribution of our work is a relaxation of the task constraint framework using caging and two algorithms for planning under this framework. In this paper, we define a cage as the condition where a robot hand constrains the configuration space of an object to a finite volume. In our case, we are interested in keeping the size of this volume small so the object can be controlled with the hand. The configuration space of the object itself can be one degree of freedom for a door hinge, a pose in the 2D plane, or a pose in 3D.

III. Relaxed Problem Formulation

We formulate the problem using the configuration space of the robot $q \in \mathcal{Q}$, the configuration space of the end-effector $g \in \mathcal{G}$, and the configuration of the constrained target object $\rho \in \mathcal{R}$. Each of these spaces is endowed with its corresponding distance metric $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$.

We represent g, henceforth termed a *grasp*, as the 6D pose of the end-effector in $SE(3)$. Although our proposed method is general enough to include joints in the *grasp* configuration, we assume that the hand joints do not move while planning. The assumption is mandated by our physical setup which does not provide accurate synchronization of arm and end-effector motion.

In the relaxed task constraint formulation, each target object is endowed with a task frame which is rigidly attached to it, and a set of grasps G represented in that task frame. The set G is carefully chosen to ensure that any grasp is guaranteed to *cage* the object. If we define $\mathcal{R}_g$ to be the set of target configurations reachable under a grasp g, then the target at configuration ρ is caged by the robot if $\rho \in \mathcal{R}_g \subset \mathcal{R}$ and every point on the the boundary of $\mathcal{R}_g$ is in collision with the end-effector at pose g. Although this is a conservative definition of a cage, it is necessary because end-effector is the only physical body known with certainty and caging should be environment independent. Note that a limiting case of a cage is a *form closure* grasp where $\mathcal{R}_g = \{\rho\}$.

In congruence with the traditional task constraint formulation, we describe the pose of grasps in G with respect to a coordinate frame that is rigidly attached to the object, termed the *task frame*. A transform T_ρ relates the task frame at an object configuration ρ to the world reference frame. The utility of this representation arises from the observation that, under a rigid grasp, the pose of the end-effector is invariant in the task frame. This allows us to compute and cache G offline, thereby improving the efficiency of the online search. At any configuration ρ, we denote the grasp set in the world frame by

$$T_\rho G = \{\, T_\rho g \mid g \in G \,\}. \tag{1}$$

Given a grasp g we define the set of (inverse kinematics)

robot configurations q that achieve g as

$$IK(g) = \{\, q \mid g = FK(q) \,\} \tag{2}$$

where $FK(q)$ is the forward kinematics transforming robot configuration to a grasp. One of the relaxed planning assumptions is that the end-effector of any configuration of the robot always lies within the grasp set G with respect to the task frame. Because this couples the motion of both the object and the robot during manipulation, their configurations need to be considered simultaneously. Therefore, we define the relaxed configuration space $\mathcal{C}$ as

$$\mathcal{C} = \{(\rho, q) \;\mid\; \rho \in \mathcal{R}, q \in \mathcal{Q}, FK(q) \in T_\rho G\} \tag{3}$$

We define the free configuration space $\mathcal{C}_{\text{free}} \subseteq \mathcal{C}$ as all states not in collision with the environment, the robot, or the object. Given these definitions, the relaxed task constraint problem becomes:

Given start and goal configurations ρ_{start} and ρ_{goal} of the object, compute a continuous path $\{\rho(s), q(s)\}$, $s \in [0,1]$ such that

$$\rho(0) = \rho_{start} \tag{4}$$

$$\rho(1) = \rho_{goal} \tag{5}$$

$$\{\rho(s), q(s)\} \in \mathcal{C}_{\text{free}} \tag{6}$$

$$FK(q(1)) \in T_{\rho(1)} G_{contact} \tag{7}$$

Eqn.4 and Eqn.5 ensure that the target object's path starts and ends at the desired configurations. Eqn.6 forms the crux of the relaxed task constraint planning problem. Because the caging criteria dictates that each grasp be in the grasp set G, Eqn.6 ensures that any robot configuration $q(s)$ produces a grasp $FK(q(s))$ that lies in the world transformed grasp set $T_{\rho(s)}G$. Eqn.7 constrains the final grasp to be within a contact grasp set $G_{contact} \subseteq G$. This set is formally defined in Section IV. Informally, any grasp in this set is in contact with the object and guarantees that the object will not move away from the goal.

While the above equations describe the geometry of the problem, we make the following assumptions about the physics of the problem. These assumptions constrain the automatically generated grasps we use for planning as well as the motion of the robot and object during manipulation.

Our analysis is purely quasi-static. The robot moves slow enough that its dynamics are negligible. Furthermore, we assume that the object's motion is quasi-static as well. This can be achieved in practice by adding a dash pot to the hinges, damping their motion, or by a sufficient amount of friction in the case of an object being dragged across a surface. We also assume that we have access to a compliant controller on the robot. Under this assumption, we are guaranteed that the robot will not jam or exert very large forces on the object being manipulated. For our robot experiments, we implemented a compliance controller on the WAM, greatly facilitated by the cable-driven transparent dynamics of the robot. The Manus arm has built-in compliance since it was originally designed for wheel-chair users and close human interaction.

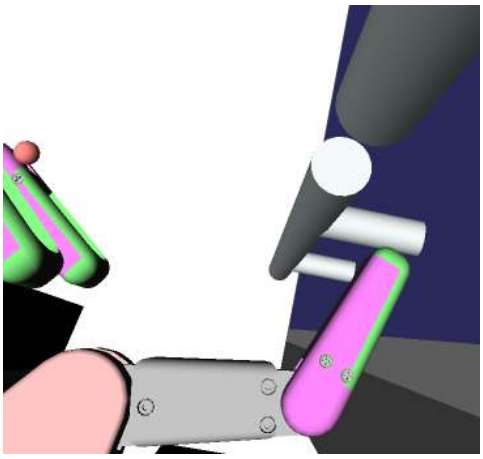

Fig. 2. A dainty grasp that was rejected by the random perturbation in the grasp exploration stage, even though it mathematically cages the handle.

IV. Grasp Sets

We generate the grasp set G by exploring the space around an initial seed caging grasp g, producing a collection of candidate grasps. Each candidate grasp that cages the object is added to G. By using caging grasps rather than grasps that fix the target object's configuration through contact force, we are able to provide the manipulator with significantly more flexibility in accomplishing the task while still guaranteeing the object cannot escape from the end effector's control.

Additionally, to guarantee that the object is eventually held and maintained at its desired final configuration, we compute the set $G_{contact}$ comprised of grasps in G that are both in contact with the object and do not allow any object motion.

The only human input to the entire algorithm is the initial seed grasp g. All subsequent steps are completely automated.

A. Generating the Grasp Set

Given a seed grasp g, we use Rapidly-exploring Random Trees (RRTs) [19] to explore the configuration space of the end-effector. In our experiments, we parametrize this space by freezing the joints of the end-effector and by parameterizing the end-effector pose in $SE(3)$ with three dimensions for translation and four dimensions for rotations represented as quaternions. The choice of exploration strategy is not central to our algorithm. In practice, we found that the RRT explored the constrained space quickly and efficiently.

Specifically, the RRT is run with a particular target configuration and only considers collisions between the object and the end-effector. From an initial grasp g, the RRT produces collision-free candidate grasps. A candidate grasp is added to G only if it passes two tests. First, we test for caging by moving choosing a random direction in the object's configuration space and moving small discrete steps along it to test if it can escape[1]. Second, we check for robustness by randomly jittering the grasp and re-testing for caging (Fig.2). Because

[1]Although this test is conservative, it produces a lot of caging grasps. A real caging test would have to run a sophisticated motion planner.

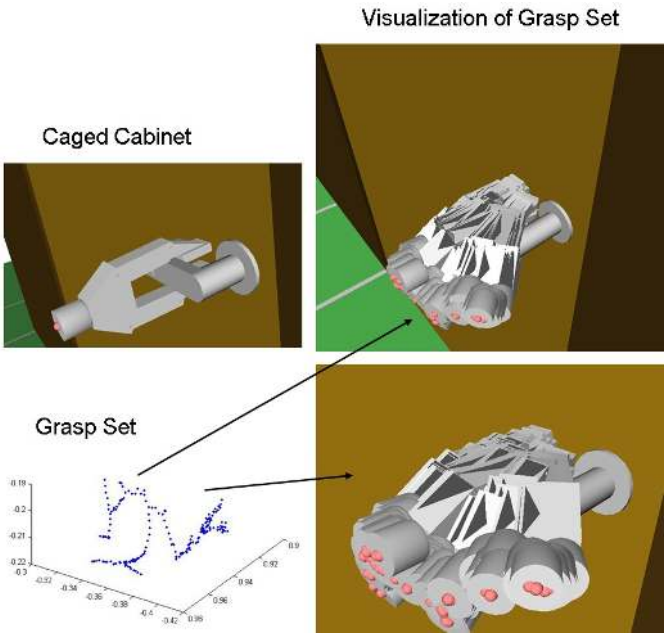

Fig. 3. Grasp Set generated for the Manus Hand.

the collision free caging grasps can be dependent on the configuration of the target, the grasp generation process runs multiple RRTs at multiple target configurations. The union of all the computed grasps is taken and a subset spanning the grasp space is extracted.

B. Generating a Contact Grasp Set

In order to guarantee that an end-effector pose is able to move the target object to its final destination, the end-effector has to exert forces through contact to keep the target object close to that configuration. We call this set $G_{contact}$.

A valid contact grasp is one where we are unable to move the object a small step ϵ without colliding with the grasp. Thus, the contact grasp approaches form-closure. This may be formalized as

$$G_{contact} = \{g \in G | \forall \rho \in \mathcal{R},\ d(\mathcal{R}_{T_\rho g}) < \epsilon\} \tag{8}$$

where

$$d(\mathcal{R}) = \max_{\rho_1 \in \mathcal{R}, \rho_2 \in \mathcal{R}} d(\rho_1, \rho_2) \tag{9}$$

is the maximum distance between any two configurations in $\mathcal{R}$. Fig.1 shows the results of the RRT exploration, pruning, and finally the grasps picked for the contact set. Fig.3 show the grasp set computed for the Manus Hand.

V. Planning with Relaxed Constraints

We describe two planning algorithms to solve the relaxed constraint problem: a discretized version and a randomized version. The randomized algorithm is more flexible and makes less assumptions about the problem statement, however the discretized algorithm is simple to implement and useful for explaining the concepts behind relaxed planning (as well as the motivation for a randomized algorithm).

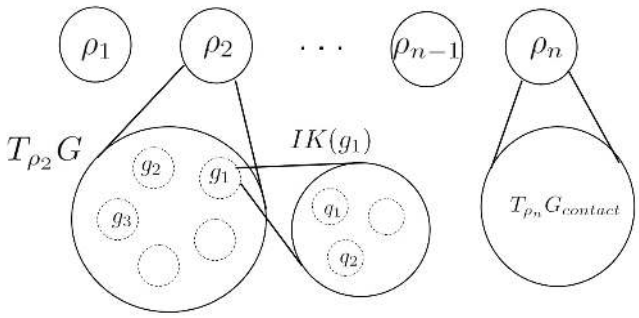

Fig. 4. The basic framework used for planning discrete paths $\{q_i\}|_1^n$ in robot configuration space to satisfy paths $\{\rho_i\}|_1^n$ in object configuration space.

A. Discrete Formulation

The underlying assumption of the discrete formulation is that a desired path of the target object is specified. Specifying the path in the object's configuration space as an input to the planner is trivial for highly constrained objects like doors, handles, cabinets, and levers. The configuration space of these objects is one dimensional, so specifying a path from a to b is easily done by disretizing that path into n points. In the more general case where an object's configuration space can be more complex, we denote its desired path as $\{\rho_i\}|_1^n$ where each of the configurations ρ_i have to be visited by the object in that order.

The discrete relaxed constrained problem is then stated as: given a discretized object configuration space path $\{\rho_i\}|_1^n$, find a corresponding robot configuration space path $\{q_i\}|_1^n$ such that

$$\forall_{1 \le i \le n}\ (\rho_i, q_i) \in \mathcal{C}_{free} \tag{10}$$

$$FK(q_n) \in T_{\rho_n} G_{contact} \tag{11}$$

$$\forall_{1 < i \le n}\ d(FK(q_{i-1}), FK(q_i)) < \epsilon_1 \tag{12}$$

$$\forall_{1 < i \le n}\ d(q_{i-1}, q_i) < \epsilon_2 \tag{13}$$

where Eqn.10 and Eqn.11 constrain the end-effector to lie in the current grasp set defined for the object and Eqn.11 guarantees the final grasp is in contact. To satisfy the continuity constraint on the robot configuration space path, Eqn.12 and Eqn.13 ensure that adjacent robot and grasp configurations are close to each other.

A straightforward discrete planning approach to solve this problem is provided in Algorithm 1. We begin by first running a feasibility test through the entire object trajectory. This step is also used to initialize the grasp and kinematics structures used for caching. We assume an inverse kinematics solver is present for every arm. Furthermore, if the arm is redundant the solver will return all solutions within a discretization level. We compute the set of contact grasps that will keep the object in form-closure at its desired final destination ρ_n (line 11). For each grasp in this set we compute IK solutions for the complete configuration of the robot, and for each IK solution we attempt to plan a path through configuration space that

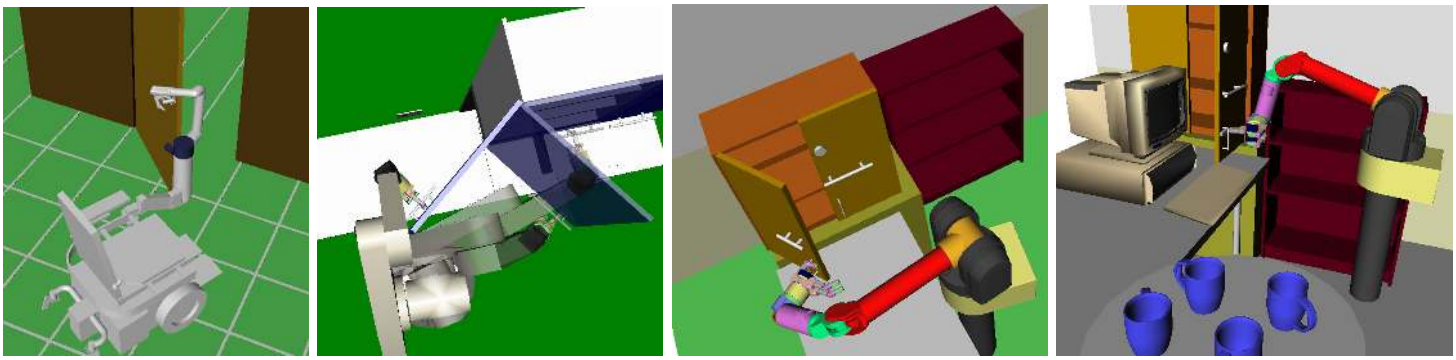

Fig. 5. The scenes used to test the algorithm: 6DOF Manus Arm, 6DOF Puma arm, and 7DOF WAM arm in an 'easy' and a 'harder' scene

Algorithm 1: $Q \leftarrow$ DISCRETESEARCH()

```
1  for i = 1 to n − 1 do
2      G_i ← T_{ρ_i} G
3      for g ∈ G_i do
4          if (IK_{i,g} ← IK(g)) ≠ ∅ then
5              break
6          G_i.remove(g)
7      end
8      if G_i = ∅ then
9          return ∅
10 end
11 for g ∈ T_{ρ_n} G_contact do
12     for q ∈ IK(g) do
13         Q_next ← DISCRETEDEPTHFIRST(q, n − 1)
14         if Q_next ≠ ∅ then
15             return { Q_next, q }
16     end
17 end
18 return ∅
```

tracks the object path $\{\rho_i\}$ using depth first search (line 13)[2].

Fig.4 provides a diagram of the discrete search framework. Given an object path $\{\rho_i\}|_1^n$ we search for a robot path $\{q_i\}|_1^n$ that consists of a sequence of robot configurations $q_i, 1 \leq i < n$ such that $FK(q_i) \in T_{\rho_i}G$ and $FK(q_n) \in T_{\rho_n}G_{contact}$. Each of these configurations q_i is generated as an IK solution from one of the grasps in the grasp set $T_{\rho_i}G$. The depth first search process takes a robot configuration at a time step j and calculates all the robot configurations that correspond to valid grasps at time $j-1$ (i.e. are members of set $T_{\rho_{j-1}}G$), then recursively processes each of these configurations until a solution is found.

B. Randomized Formulation

There are several disadvantages to the discretized algorithm. First, it is highly dependent on the discretization level of the grasp set and IK solver. For robots with six degrees of freedom

[2]This depth first search expands states in the same order as A* would using a heuristic function based on (an underestimate of) the target object distance to goal.

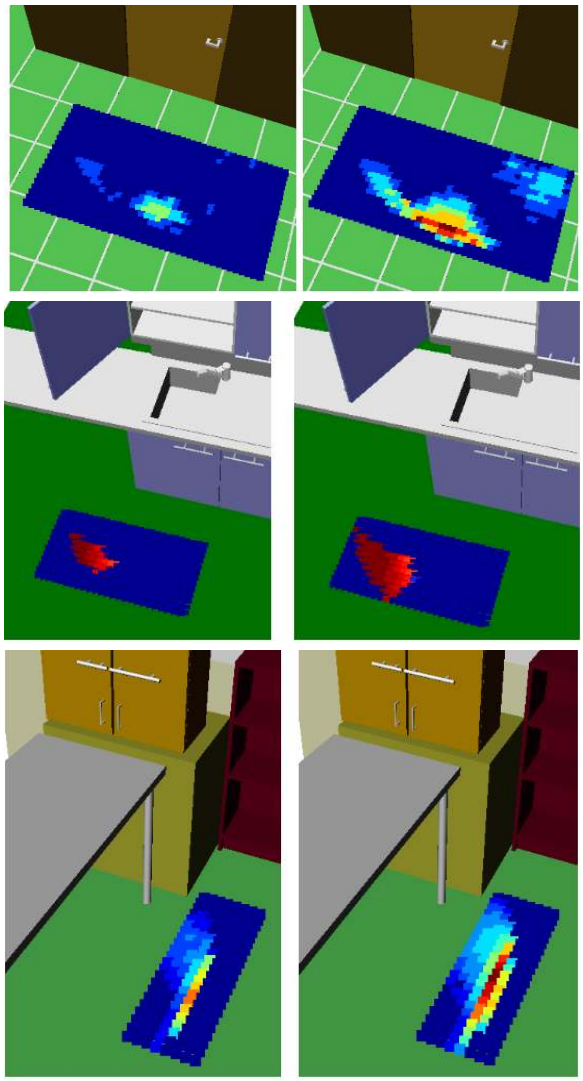

Fig. 6. Comparison of fixed feasibility regions (left) and relaxed feasibility regions (right) for each scene.

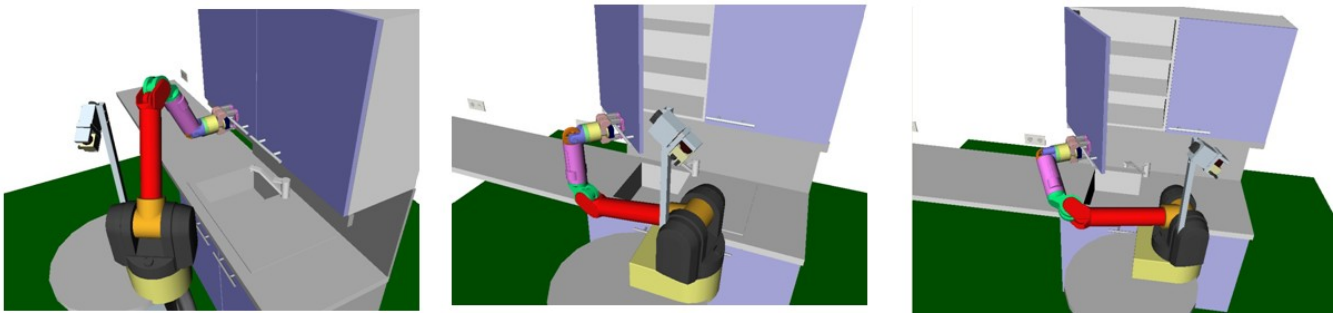

Fig. 7. WAM arm used to open a kitchen cupboard.

or less, enumerating all IK solutions isn't a problem. However, as soon as the joints increase or a mobile base is considered, the discretization required for $IK(g)$ to reasonably fill the null space grows exponentially. Second, the desired object trajectory is fixed, which eliminates the possibility of moving the door in one direction and then another to accomplish the task (see [12] for an example where this is required).

To overcome these limitations, we also applied a randomized planner to the problem. We chose the Randomized A* algorithm [20], which operates in a similar fashion to A* except that it generates a random set of actions from each state visited instead of using a fixed set. Randomized A* is well suited to our current problem because it can use the target object distance to goal as a heuristic to focus its search, it can guarantee each state is visited at most once, it does not need to generate all the IK solutions for a given grasp, and it can return failure when no solution is possible. The key difference between Randomized A* and regular A* is the sampling function used to generate neighbors during the search. For our relaxed constraints problem the task of this sampling function is to select a random configuration (ρ_{new}, q_{new}) and a random grasp $g_{new} \in T_{\rho_{new}} G$ such that $q_{new} \in IK(g_{new})$. Ideally, this should be done efficiently without wasting time considering samples previously rejected for the same configuration. The A* criteria will ensure that the same configuration isn't re-visited and that there is progress made towards the goal, so the sampling function needs only return a random configuration in $\mathcal{C}_{\text{free}}$ around the current configuration (ρ, q) as fast as possible.

Algorithm 2 provides our implementation of the sample function. It first samples a target object configuration ρ_{new} close to the current configuration ρ (line 3), then searches for feasible grasps from the new grasp set $T_{\rho_{new}} G'$ (line 6), and then samples a collision-free IK solution close to q (line 8). In order to guarantee we sample the entire space, RANDOMCLOSECONFIG should discretize the sampling space of the target configuration so that the number of distinct ρ_{new} that are produced is small. This is necessary to ensure that sampling without replacement is efficient. Each time a sample is chosen (line 6), it is removed from $\mathcal{G}_{\rho_{new}}$ so it is never considered again, an operation that takes constant time. If the target is close to its goal then G' is the contact grasp set $G_{contact}$, otherwise G' is the regular grasp set G. Once a grasp is found, SAMPLEIK samples the nullspace of the IK solver around q until a collision-free solution is found. If not, the entire process repeats again. If all grasps are exhausted for a particular target configuration, the sampler checks for termination conditions and returns false (line 9).

Algorithm 2: $\{\rho_{new}, q_{new}\} \leftarrow$ SAMPLENN(ρ, q)

```
1  G ← ∅, q_new ← ∅
2  while q_new = ∅ do
3      ρ_new ← RANDOMCLOSECONFIG(ρ)
4      if not EXIST(G_ρnew) then
5          G_ρnew ← T_ρnew G'
6      g_new ← SAMPLEWITHOUTREPLACEMENT(G_ρnew)
7      if g_new ≠ ∅ then
8          q_new ← SAMPLEIK(g_new, q)
9      else if CHECKTERMINATION() then
10         return {∅, ∅}
11 end
12 return {ρ_new, q_new}
```

	Discrete	Randomized
6DOF Manus Arm	441%	503%
6DOF Puma Arm	130%	126%
7DOF Barrett WAM	13%	24%
7DOF Barrett WAM (Harder)	163%	162%

TABLE II

INCREASE IN FEASIBILITY SPACE WHEN USING RELAXED PLANNING COMPARED TO FIXED-GRASP PLANNING.

VI. EXPERIMENTS

The robotic simulation environment we used to perform all planning, testing, and real robot control is OpenRAVE [21], the Open-Source Cross-Platform Robotics Virtual Environment. To test the performance of the algorithm, the planning times and feasibility regions are calculated for three different robots in various scenes (Fig.5, Fig.7). All the handles of the target objects are measured carefully from their real-world counterparts. The tasks are as follows:

- The Manus Arm is required to open the door 90 degrees.

	# Trials	Grasp Set Size	Discrete (Successes)	Discrete (Failures)	Randomized (Successes)	Randomized (Failures)
6DOF Manus Arm	7784	550	0.235 s	0.234 s	0.143 s	0.23 s
6DOF Puma Arm	6755	300	1.49 s	0.043 s	1.83 s	0.028 s
7DOF Barrett WAM	2734	276	10.5	8.43	10.5	37.4
7DOF Barrett WAM (Harder)	2422	123	0.116 s	0.021 s	0.209 s	0.029 s

TABLE I

STATISTICS FOR THE SCENES TESTED SHOWING AVERAGE PLANNING TIMES (IN SECONDS) AND SIZE OF THE GRASP SETS USED.

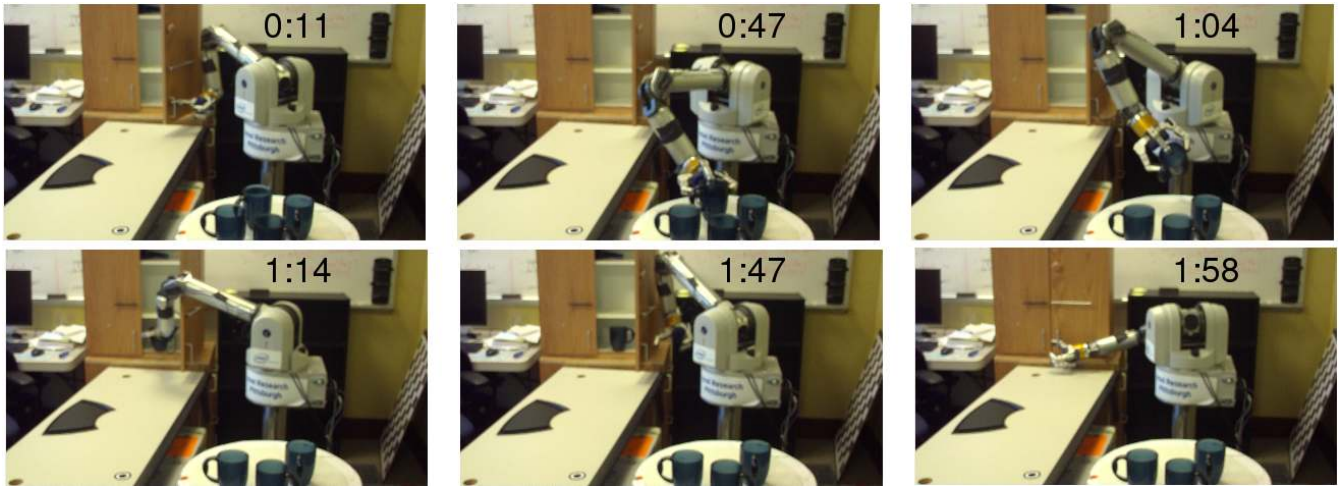

Fig. 8. WAM arm autonomously opening a cupboard, putting in a cup, and closing it. Wall clock times from start of planning are shown in each frame.

- The Puma Arm is required to open the cupboard 115 degrees.
- The WAM arm is required to open the closer cabinet 90 degrees and the farther cabinet 60 degrees.

In each scene, the robot is randomly positioned and oriented on the floor, and then the planners are executed. Thousands of random positions are tested in each scene to calculate average running times (Table I). The parameters for the randomized algorithm stayed the same across all robots. Note that the planning times for the easier WAM scene are much higher than the rest of the scenes. This shows that the more open the space is, the longer it takes to search for all possible solutions, and especially longer to declare failure when a solution doesn't exist.

To show that relaxed grasp sets really do increase the regions the arm can achieve its task from, we compare the feasibility regions produced with the relaxed grasp set method and the fixed grasp method. The fixed grasp method uses a single task-frame grasp throughout the entire search process. To make things fair, we try every grasp in $G_{contact}$ before declaring that the fixed grasp method fails. Table II shows how many times the feasibility region increased for the relaxed methods compared to the fixed method. As expected, the lower dimensional manipulators benefit greatly from relaxed task constraints. Furthermore, the door can be opened much further using the relaxed approach than with the fixed grasp method. The randomized algorithm's improvement over the fixed method is occasionally less than that of the discrete algorithm because we are using early termination criteria; running the algorithm for longer produces feasibility regions that are greater than or equal to what the discrete algorithm produces. Fig.6 shows the feasibility regions in each scene between relaxed grasps and fixed grasps.

Real experiments were done on two robots: the Manus Arm on a wheelchair opening a door (Fig.9), and the Barrett WAM putting cups in a cupboard (Fig.8). It is impossible to open the door so wide with the Manus Arm without considering relaxed grasps because the reachability is so low. The experiment we performed with the WAM is to autonomously open a cabinet, put a cup inside it, and close it. The robot autonomously planned for collision-free and reachable grasps when picking up the cup using the grasp planning framework proposed by [22]. It should be noted that the final destination is very tight, but the planner was able to find a solution and the robot successfully completed execution of the entire task in a combined time of 1 minute and 58 seconds.

VII. CONCLUSIONS AND FUTURE WORK

We have proposed a motion planning method for constrained manipulation tasks that combines object "caging grasps" and efficient search algorithms to produce motion plans that satisfy the task constraints. The effectiveness of our approach has been illustrated by experimental results on two different real-world autonomous manipulation tasks. Relaxing the task constraints can give the arm more chances to finish the task without relying on synchronization with

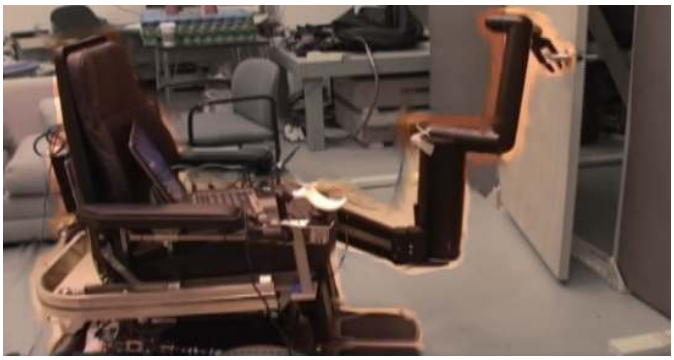

Fig. 9. Manus arm on a wheel chair opening a door.

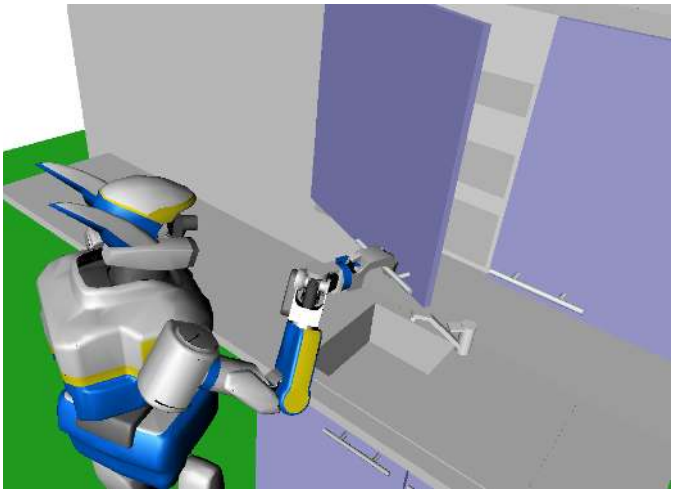

Fig. 10. The humanoid robot HRP-2 opening a cupboard.

the mobile base. This method is especially useful when the robot's own localization is not accurate because it allows for the robot to control how far away it is from collisions. Furthermore, these results can be generalized to arbitrary pushing tasks where the robot "cages" certain directions of the object configuration space. Our experimental results have shown that planning using caging grasps can be implemented efficiently, and can result in improved overall planning times and execution performance. The proposed algorithm generally applies to a large class of manipulators, and can easily be adapted for the dynamic actions of humanoid robots (Fig.10).

VIII. Acknowledgements

This project is partially supported by the Quality of Life Technology Center and the Personal Robotics project at Intel Research Pittsburgh. We are grateful to the HERL Pittsburgh Lab for the wheel chair hardware, Exact Dynamics BV for the Manus arm hardware. We also want to thank Mike Vande Weghe for the WAM hardware support and his invaluable input.

References

[1] M. Mason, "Compliance and force control for computer-controlled manipulators," vol. 11, no. 6, 1981, pp. 418–432.

[2] E. Rimon and A. Blake, "Caging 2d by one-parameter two-fingered gripping systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1986.

[3] E. Rimon, "Caging planar bodies by one-parameter two-fingered gripping systems," *The International Journal of Robotics Research*, vol. 18, pp. 299–318, 1999.

[4] "http://www.exactdynamics.nl."

[5] "http://www.barrett.com."

[6] W. Kuperberg, "Problems on polytopes and convex sets," in *DIMACS Workshop on Polytopes*, 1990.

[7] G. A. S. Pereira, V. Kumar, and M. F. M. Campos, "Decentralized algorithms for multirobot manipulation via caging," in *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2002.

[8] A. Sudsang, J. Ponce, and N. Srinivasa, "Algorithms for constructing immobilizing fixtures and grasps of three-dimensional objects," in *In J.-P. Laumont and M. Overmars, editors, Algorithmic Foundations of Robotics II*, 1997.

[9] M. Vahedi and A. F. van der Stappen, "Geometric properties and computation of three-finger caging grasps of convex polygons," in *Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering*, 2007.

[10] O. Khatib, "A unified approach to motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, 1987.

[11] M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," *ASME Journal of Dynamic, Measurements and Control*, vol. 103, 1981.

[12] M. Stilman, K. Nishiwaki, and S. Kagami, "Learning object models for humanoid manipulation," in *IEEE International Conference on Humanoid Robotics*, 2007.

[13] J. de Schutter, T. de Laet, J. Rutgeerts, W. Decre, R. Smits, E. Aertbelien, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *International Journal of Robotics Research*, vol. 26, no. 5, pp. 433–455, 2007.

[14] A. Jain and C. C. Kemp, "Behaviors for robust door opening and doorway traversal with a force-sensing mobile manipulator," in *Proceedings of the Manipulation Workshop in Robotics Science And Systems*, 2008.

[15] M. Prats, P. J. Sanz, and A. P. del Pobil, "A control architecture for compliant execution of manipulation tasks," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[16] M. Prats, P. Martinet, A. P. del Pobil, and S. Lee, "Vision/force control in task-oriented grasping and manipulation," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[17] M. Prats, P. J. Sanz, and A. P. del Pobil, "A sensor-based approach for physical interaction based on hand, grasp and task frames," in *Proceedings of the Manipulation Workshop in Robotics Science And Systems*, 2008.

[18] V. Ng-Thow-Hing, E. Drumwright, K. Hauser, Q. Wu, and J. Wormer, "Expanding task functionality in established humanoid robots," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2007.

[19] J. Kuffner and S. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[20] R. Diankov and J. Kuffner, "Randomized statistical path planning," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[21] R. Diankov, "http://openrave.programmingvision.com."

[22] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner, "Grasp planning in complex scenes," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2007.