Report for CS 6751, Spring 2017
Grasping and Prehensile Manipulation Project

Vighnesh Vatsal

Sibley School of Mechanical and Aerospace Engineering
Cornell University

1 Introduction

Grasping end-effectors are one of the most commonly employed manipulation tools in robotic arms. This has followed historically from designs of anthropomorphic end-effectors that mimic the grasping motion performed by the human thumb and forefingers. While grasping is indeed an important function of our hands, it is far from the only mode of interaction between our hands and the environment.

Grasping an object in one's hand effectively turns it into an extension of the arm, being completely constrained and moving with the human arm. However, there are numerous scenarios where such a constraint is neither desired nor necessary. For example, if the task at hand is to move a large block of wood over a flat surface, it would be impractical to try and grasp it. Instead, most people would try to push it along the surface. Another mode of interaction is *caging*, where the body is free to move within the gripper but cannot escape it. An example of this mode would be sliding a mug along a table while it is loosely held within the hand. As humans, we also use a host of other factors to arrive at a manipulation mode, such as gravity, friction, the shape, size and weight of the object, and implicitly, the shape, size and effort capabilities of our own hands.

However, when imbuing robots with manipulation capabilities, the focus has not been on these less-constrained modes. The reasons for this range from safety to computational planning load. Expending extra effort to tackle uncertainties in the pose of an object being manipulated by a robot using a less-constrained mode was not desirable. The planning problem for a robotic arm remains virtually unchanged if the object is simply added as a rigid body at the end of the kinematic chain. Therefore, most state of the art robots and manipulation planners treat the problem as being divided into two parts: finding a grasping pose, and then performing a motion with the object firmly grasped. This approach is fairly efficient and useful in multiple real-world applications. However, by ignoring the possibility of other modes of manipulation, robots are unable to perform a range of tasks that humans are capable of. As another consequence of this single-minded focus on grasping, we encounter robot motions that seem highly non-intuitive and inefficient to a human observer.

I aim to extend the state of the art by generating a planner that can switch between pushing, caging and grasping based on the environment and goal for

a particular prismatic object. At present, an approach based on tree-search is being explored where the switching condition emerges as a consequence of a suitable search heuristic cost function. This approach is being implemented on a Baxter commercial robot with two-fingered grippers as the end-effectors of two 7-DoF arms.

2 Literature Review

Most literature on grasping-based manipulation [11] considers the scenario in which the robotic arm firmly grips the object, immobilizing it and making it a part of the arm. To place the object in its desired goal configuration is then considered to be a path-planning problem in the C–space of the robotic arm [3], which has been augmented with the grasped object.

On the contrary, humans tend not to manipulate objects in their environment with only pick-and-place style grasping. For instance, to move a box across a table, one may slide it over the surface by pushing [4]. This form of motion is termed as *prehensile* manipulation. In this class of motion, the object is not assumed to be an extension of the manipulator. Based on the geometry of the object and the robot's end-effector, we can identify configurations that either completely immobilize an object (grasps), or constrain it in such a way that it cannot escape to infinity, but is not completely immobilized (cages). The full geometric framework for these conditions is presented in [12].

Broadly, we can distinguish between control and planning for manipulation tasks. Planning methods take the full plan of action into account, generally simulating the whole process offline to check for conditions of feasibility and optimality [9] of the motion. Control-oriented approaches tend to be real-time, sensor feedback-based methods that aid in the performance of the motion at an implementation level [5], generally ensuring local optimality. Task Frame representational approaches [10], and methods involving friction and physics modeling [4] tend to fall into this category.

This project is aimed at developing a manipulation planner for a robot arm that uses the motion primitives of grasping, caging, and pushing, along with arm translation and rotation. Thus far, most prior work on planning has assumed the grasping primitive to be true, thus reducing the problem to finding a path in C–space of the robot. One such set of algorithms distinguishes between *transit* paths, where the object is not in contact with the robot, and *transfer* paths, where the contact is achieved [1]. In the interim of switching between these two modes, it is assumed that an immobilizing grasp can be found. Work specific to prehensile motions (caging, pushing) includes efforts to geometrically model a pushing motion to facilitate a grasp [8], and caging-specific tree-search based algorithms [6]. There have also been efforts to incorporate object physics, friction and mechanics modeling for highly dynamic motion plans that allow for the object to move as an independent object during the plan [7]. Finally, there are task-space based planners that assume a quasi-static motion of objects, while

taking into account the uncertainty in their position, implemented using an RRT-based search [2].

3 Technical Description

3.1 Motion Primitives

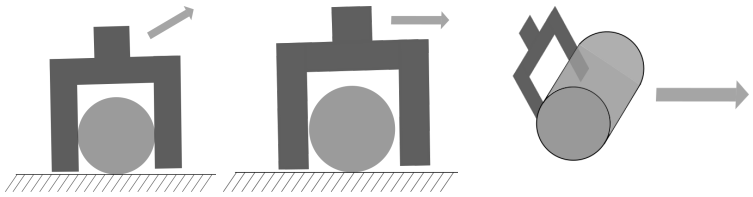

Fig. 1. The three motion primitives in the manipulation task with a two-finger gripper and a pipe: grasping, caging, and pushing

The manipulation task to be performed in this work is for a two-finger end effector-based robotic arm to take an object from an initial state to a specified goal state. To accomplish this, the robot makes use of three manipulation primitives (Fig. 1):

- *Grasping:* As mentioned in Section 2, this is the most commonly employed manipulation primitive in robotic arms. It involves a full force closure on the object, rendering it immobile and effectively converting it into an extension of the end-effector. It allows for the full 6-degrees of freedom (DoF) translation and rotation of the object.
- *Caging:* Caging involves constraining the object such that it is unable to move arbitrarily far away from the end-effector. However, it is free to move within the geometric confines of the end effector. Specifically in the case of a two-finger gripper caging a pipe about its diameter, the pipe is free to move in the axial direction, and free to rotate about any axis. The only constraint is that its center of mass must translate along with the gripper on a flat surface.
- *Pushing:* Pushing differs from caging in the way that the gripper is unable to rotate the object. It can only slide it along a flat surface while preserving orientation.

3.2 System Description

The setup for demonstrating the proposed prehensile manipulation planner consists of the following components: a Baxter robot (Fig. 2), an overhead Kinect camera, a flat table, a raised board, and a marked PVC pipe to be manipulated.

There are underlying simplifying assumptions in the structure of this setup:

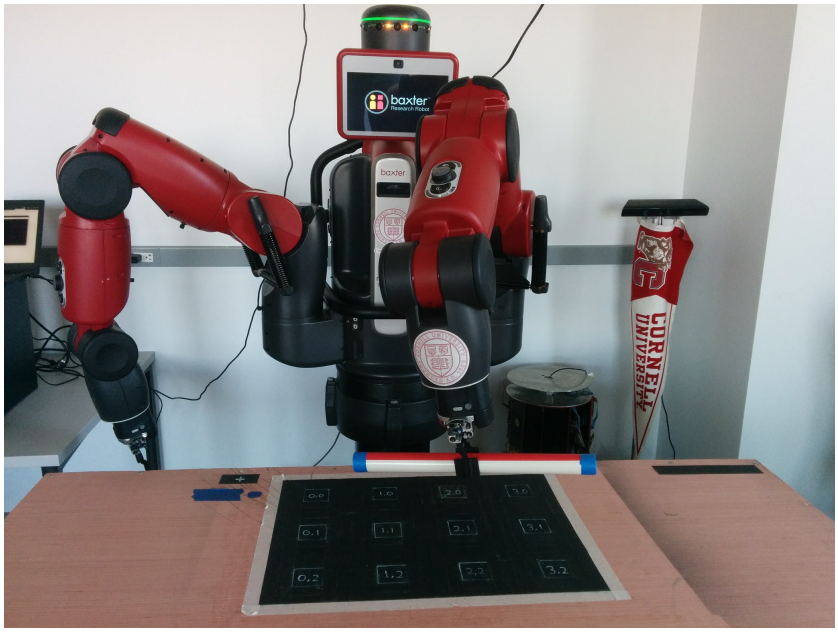

Fig. 2. Baxter robot system setup for the manipulation task

- The PVC pipe is specifically chosen because of its prismatic shape for ease of grasping, pushing and caging it by the Baxter robot's gripper. We are not investigating the effect of object affordances in the present planning scheme.
- All motions are performed in a quasi-static manner. This assumption also implies that there is sufficient friction present between every relevant surface to prevent dynamic motions such as rolling of the pipe while pushing.
- There are no obstacles present in this environment.
- The positions the center of mass of every object apart from the pipe are static.
- The pipe always remains coplanar with the flat table surfaces.

3.3 Planning Framework

The pipe is a rigid body, whose state is $s \in SE(3)$. It can lie on a series of flat planes with fixed Z coordinates and prescribed limits on the X and Y coordinates. The problem statement is then to take the pipe from $s_i \in SE(3)$ to $s_f \in SE(3)$ using a sequence of movements of the robotic arm in its C-space. The structure of the problem imposes that only the (x, y, z) and θ (rotation about the Z axis) coordinates of the pipe are variable. These are computed using RGB-sensing of the end-points of the pipe with a known radius. Since planning is performed in the C-space, it is beneficial to formulate the problem in terms of the robot's end-effector. We can use the same four coordinates for this purpose:

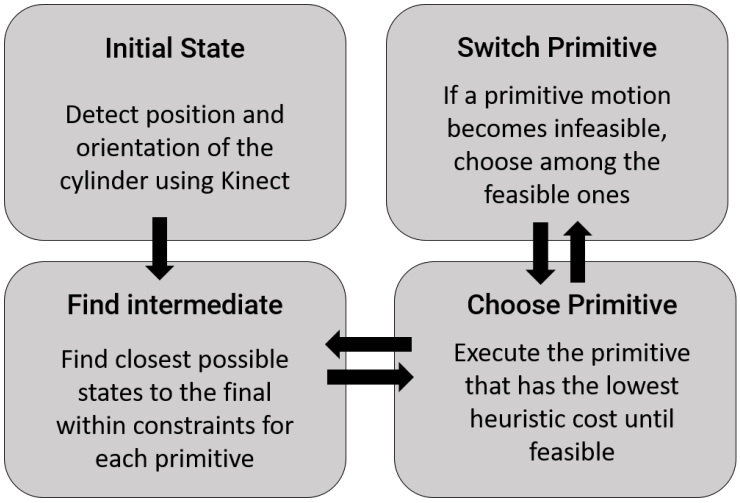

Fig. 3. Planning framework followed for the manipulation task

(x, y, z, θ), but this time of the end-effector. Each primitive imposes constraints on the motion of the robot end-effector:

- Pushing requires that θ and z remain constant, while x and y are variable, normal to θ of the pipe.
- Caging requires z to be constant, all other coordinates are variable
- Grasping imposes no extra constraints

The Open Motion Planning Library (OMPL) allows us to specify initial and final positions for the end-effector in Cartesian space and generates motion plans in the C-space for Baxter's arm using the RRT* algorithm.

Once the initial state of the pipe is known, intermediate states $s_t \in SE(3)$ is found that is as close as possible to s_f within the constraints for each primitive, e.g. if the plan involves pure translation in all three directions, the intermediate state for caging or pushing would be $(x_f, y_f, z_i, \theta_i)$, while that for grasping would be $(x_f, y_f, z_f, \theta_i)$

Then for each of these intermediate states, a heuristic cost function $h(p)$ is computed for a path p going from s_i to s_t in the RRT* algorithm:

$$h = f_{energy} + f_{config} + f_{primitive} \tag{1}$$

Where f_{energy} denotes the energy expended by the robot in performing a plan, f_{config} is the penalty for being far away from the final configuration, and $f_{primitive}$ is the "ease" of each primitive, following the order pushing $<$ caging $<$ grasping.

The primitive that has the lowest heuristic cost is executed till s_t. At this stage, new intermediates are attempted to be found for each primitive. If another

primitive has a lower cost, or the others are infeasible, e.g. only grasping can lift the pipe above the table, the planner switches over to another primitive motion, until the goal state s_f is reached.

4 Implementation

4.1 Kinect Sensing and Transformations

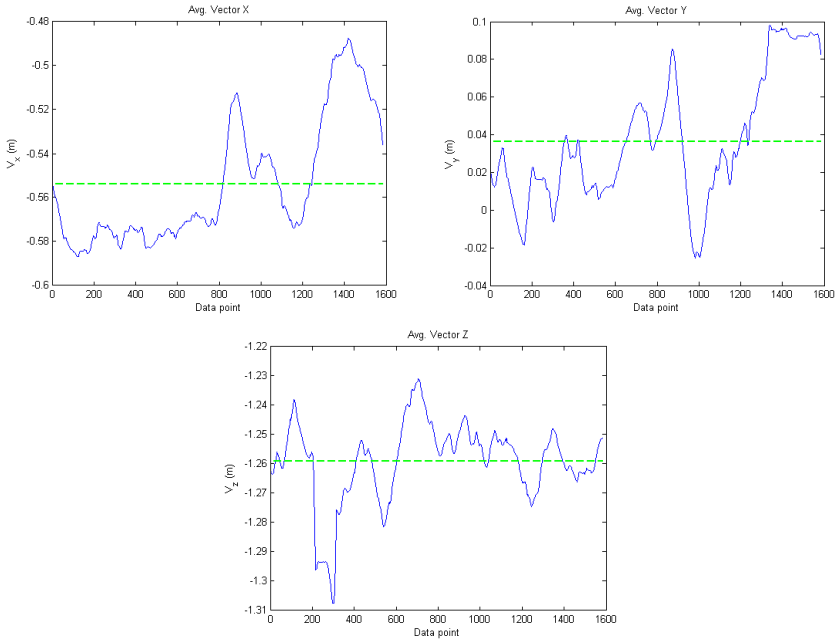

Fig. 4. Components of the position vector between the Kinect and Baxter frames of reference

The overhead Kinect camera is used to track two colored blocks attached to the end-points of the pipe. All data from the Kinect frame is transformed to the Baxter frame using a homogeneous transformation:

$$\begin{bmatrix} x_{P/B} \\ y_{P/B} \\ z_{P/B} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & -x_{B/K} \\ 1 & 0 & 0 & -y_{B/K} \\ 0 & 0 & -1 & -z_{B/K} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{P/K} \\ y_{P/K} \\ z_{P/K} \\ 1 \end{bmatrix} \tag{2}$$

Here $x_{P/B}$ is the position of a point P in the Baxter frame, $x_{P/K}$ is the position sensed in the Kinect frame, and $x_{B/K}$ is the position vector of the Baxter's origin in the Kinect frame.

A calibration routine is performed to find the position vector between the Baxter's origin and the Kinect frame's origin using end-effector position detection built into the robot. The robot's arm is moved through various positions in 3-D such that the end-effector can be tracked by the Kinect sensor. After reducing the signal noise with a moving average filter, the mean value of the position vector is obtained (Fig. 4).

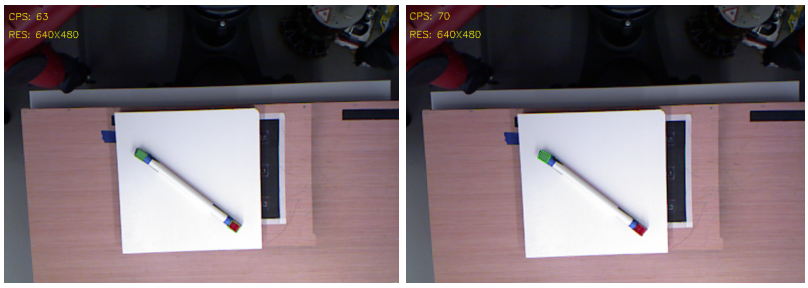

Fig. 5. Images of the end points of the pipe captured by RGB sensing on the Kinect

Once the positions of the end points of the pipe are obtained (Fig. 5), its orientation on the table can be determined. This allows us to set a position and orientation of the end-effector appropriately for pushing, caging or grasping. For caging and grasping, the gripper is over the mid-point with the orientation parallel to the pipe's, while for pushing it is at a distance equal to the radius of the pipe away from the mid-point along the normal to the pipe, and oriented along the normal.

4.2 MoveIt! Planning

The motion planning using an RRT* algorithm mentioned in Section 3.3 is implemented using the MoveIt! framework for ROS [13]. MoveIt is convenient because it has built-in support for Kinect sensing, and it presents a simplified framework for writing a motion planner. Additionally, the Baxter robot has existing packages containing SRDF robot description models.

MoveIt allows us to add geometric objects in the planning framework, e.g. the desk in our environment represented as a cuboid in front of the robot. This is translated into constraints for the RRT* algorithm. It also allows us to visualize the planned path (Fig. 6).

4.3 Results

In the final implementation, the task was to move the pipe along the desk, ideally achieved using caging, and then to lift it and place it at an elevated position on a wooden board kept on top of the table.

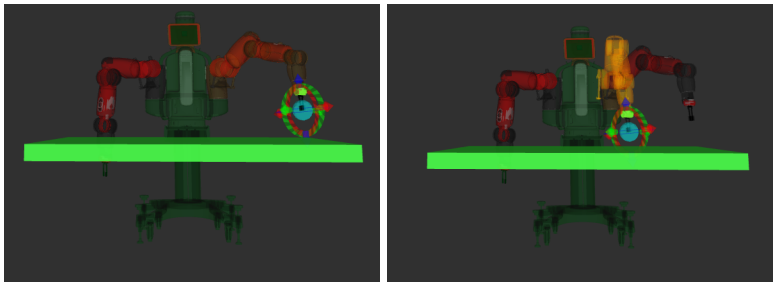

Fig. 6. Visualization of Baxter's MoveIt model and environment in ROS Rviz

The RRT* planner accomplished this in stages: the first being to position the arm on top of the pipe in a pre-caging position, with the center of the end-effector directly above the mid-point of the pipe, and the gripper handles parallel to the pipe. After that, the gripper is brought down to complete the cage (Fig, 7).

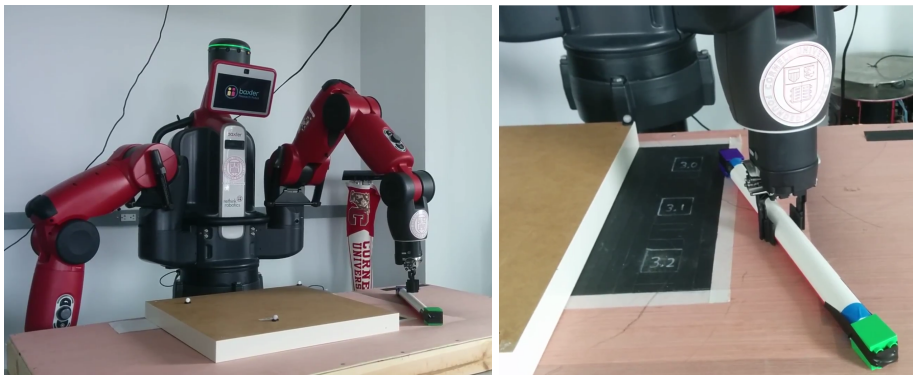

Fig. 7. The two stages of caging: positioning, and completing the cage

Once a cage is established, the robot drags the pipe along the table until the mode switches to grasping. This occurs when it gets close to the edge of the board and cannot raise the pipe by caging. If the pipe had been rotated while performing the dragging motion after caging, a re-orientation of the end-effector is carried out so that the gripper's fingers are parallel to the pipe again, with the gripper's center above the pipe's mid-point. Then a grasp is completed by bringing down the gripper onto the pipe. One the grasp is successful, the pipe is lifted off the table and placed onto the board at the desired position and orientation (Fig. 8).

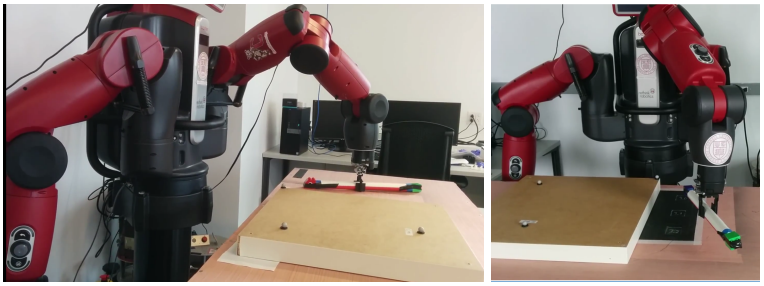

Fig. 8. Completing a grasp, and moving the pipe to the goal position on the board

5 Conclusions

This project served as a proof of concept for a manipulation planner that was able to switch between modes of manipulation based on a cost function that depended on the object and task. Presently, the limitations include long durations between planning stages ($\sim$ 20 seconds), imprecise positioning due to Kinect tracking errors, and minimal tolerance to such errors due to the open-loop nature of control. Also, the completeness of this approach has not been theoretically guaranteed. Going forward, a closed-loop controller based on visual servoing could be explored, along with offline generation of trajectories to speed up the physical implementation.

References

1. Alami, R., Laumond, J.P., Siméon, T.: Two manipulation planning algorithms. In: WAFR Proceedings of the workshop on Algorithmic foundations of robotics. pp. 109–125. AK Peters, Ltd. Natick, MA, USA (1994)
2. Berenson, D., Srinivasa, S.S., Kuffner, J.J.: Addressing pose uncertainty in manipulation planning using task space regions. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. pp. 1419–1425. IEEE (2009)
3. Brock, O., Kuffner, J., Xiao, J.: Motion for manipulation tasks. In: Springer Handbook of Robotics, pp. 615–645. Springer (2008)
4. Chavan-Dafle, N., Rodriguez, A.: Prehensile pushing: In-hand manipulation with push-primitives. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. pp. 6215–6222. IEEE (2015)
5. Dafle, N.C., Rodriguez, A., Paolini, R., Tang, B., Srinivasa, S.S., Erdmann, M., Mason, M.T., Lundberg, I., Staab, H., Fuhlbrigge, T.: Extrinsic dexterity: In-hand manipulation with external forces. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on. pp. 1578–1585. IEEE (2014)
6. Diankov, R., Srinivasa, S.S., Ferguson, D., Kuffner, J.: Manipulation planning with caging grasps. In: Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on. pp. 285–292. IEEE (2008)
7. Dogar, M., Hsiao, K., Ciocarlie, M., Srinivasa, S.: Physics-based grasp planning through clutter (2012)

8. Dogar, M.R., Srinivasa, S.S.: Push-grasping with dexterous hands: Mechanics and a method. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. pp. 2123–2130. IEEE (2010)
9. Kavraki, L.E., LaValle, S.M.: Motion planning. In: Springer handbook of robotics, pp. 139–162. Springer (2016)
10. Prats, M., Sanz, P.J., Del Pobil, A.P.: A framework for compliant physical interaction. Autonomous Robots 28(1), 89–111 (2010)
11. Prattichizzo, D., Trinkle, J.C.: Grasping. In: Springer handbook of robotics, pp. 955–988. Springer (2016)
12. Rodriguez, A., Mason, M.T., Ferry, S.: From caging to grasping. The International Journal of Robotics Research 31(7), 886–900 (2012)
13. Sucan, I.A., Chitta, S.: Moveit! Online at http://moveit. ros. org (2013)