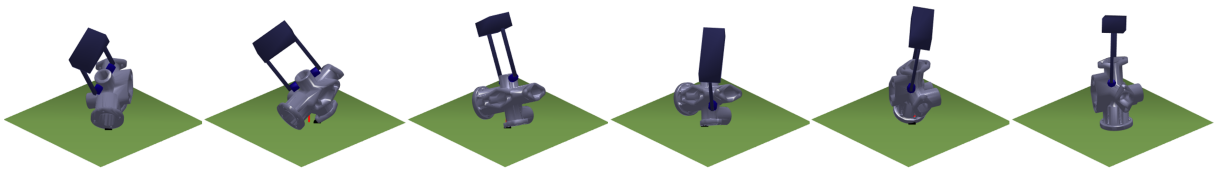


Fast Planning for 3D Any-Pose-Reorienting using Pivoting

Yifan Hou, Zhenzhong Jia and Matthew T. Mason *Fellow, IEEE*

***Abstract*— In this paper, we consider reorienting 3D objects on a table using a two-finger pinch gripper. Given the 3D mesh model of the object, our algorithm solves for the gripper motions that are required to transit between arbitrary object poses, grasping positions and gripper poses. The two motion primitives we used, *pivoting* and *compliant rolling*, enable us to decompose the planning problem and solve it more efficiently. Our algorithm can work with approximated (simplified) mesh models while being robust to approximation errors, thereby allowing us to efficiently handle object shapes with originally thousands of facets. We show the effectiveness of the proposed method by testing on objects with non-trivial geometry in both simulations and experiments. Results show that our algorithm can solve a larger range of reorienting problems with less number of making and breaking contacts when compared to traditional pick-and-place based methods, especially when the gripper workspace is highly constrained.**

I. INTRODUCTION

Many works on robot motion planning focus on doing simple grasping in complicated environments, in which a grasp is declared successful as soon as it is made. In industrial applications such as polishing, soldering and assembling, however, the objects usually lie in a clean environment but need to be quickly reoriented to a specific pose or a sequence of poses in the world frame. We call them *reorienting* problems. Traditional methods [15], [16], [30], [28], [7], [32] for 3D reorienting planning only consider the transitions between *stable* placements of the object. We call these problems *Stable-Placement-Reorienting*. A more general task is *Any-Pose-Reorienting*, in which the object transits between any two 3D poses. In this problem, the solution may not pass through a stable pose at all. In this paper, we propose a planning method for Any-Pose-Reorienting.

Without a dexterous hand, pick-and-place is the simplest motion primitive used for reorienting an object. In pick-and-place, the manipulator grasps the object firmly, drops it at a different stable pose then grasps again. However, the motion plan could be inefficient, because the gripper has to rotate together with the object; also the method could lost some potential solutions, because the robot can only attempt grasps that are feasible for both the initial and the final object poses.

*This work was supported under NSF Grant No. 1662682.

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {yifanh, zhenzjia}@cmu.edu, matt.mason@cs.cmu.edu

On the contrary, humans can perform the task elegantly even with two finger pinch grasps. Humans use a larger repertoire of motion primitives with the help of *extrinsic dexterity* [8] such as gravity, inertia forces and extrinsic contact forces. However, it remains a challenge to do planing using extrinsic dexterity for general shaped objects [8]. In this work we analyze two of the motion primitives that are helpful for planning, both utilize the table and gravity. The first one is *pivoting on the table*, in which the object is pinch-grasped but can rotate about the grasp axis. Part of the object is in contact with the table under gravity. The second motion primitive is *compliant rolling on the table*, in which the object is firmly grasped. The object rotates with the gripper, while maintaining contact with the table underneath. The robot has compliance in the table's normal direction, in order to avoid large impact forces.

When used together, these two motion primitives provide several benefits. Firstly, the gripper does not need to follow the rotation of the grasped object during pivoting, opening possibilities for more efficient solutions. Secondly, the object motion is kinematically determined by the motion of gripper under quasi-static assumptions. Comparing with throwing&catching [8], pivoting with gravity [31] or inertia force [26], [10], the proposed motion primitives have lower requirement on the bandwidth of the robot.

With the two motion primitives, our proposed strategy could quickly generate motion plan or declare infeasibility for a given grasp position, making it efficient to select a grasp. When the task cannot be fulfilled with one grasp, we perform graph search on an improved *Regrasping Graph* [32] to plan a sequence of grasps. The planning can be done in real time. Most importantly, our method makes it convenient to analyze the robustness of a motion plan towards modeling errors, so that we can always plan with a simplified object mesh model.

To summarize, our contributions are:

- Quasi-static analysis of the two motion primitives: *pivoting on the table* and *compliant rolling on the table*;
- A real-time planning framework for *Any-Pose-Reorienting* problems using these two motion primitives;
- Robustness analysis for planning with simplified 3D mesh models.

II. RELATED WORK

A. Grasp Planning

The first step of reorienting is to make a grasp. Many studies on grasping focus on evaluating the quality of a grasp under different situations [21], [35], [23]. To find a grasp, the basic idea is to detect antipodal points [6]. For complicated and non-smooth object shapes, people use simplified model for grasp point sampling, including bounding box [11], surface sampling [24] or shape primitives [19]. Model free grasping [12], [20], [25] is also a choice. However, using shape approximation or model free grasping method usually provides no guarantee on the quality of the grasp, which could cause troubles for post-grasping manipulations including reorienting. In our framework, we directly sample antipodal grasp points on the original object mesh model using the method described in [32].

B. Regrasping and Reorienting

Closely related to our work, *regrasping* problems focus on changing the pose of object with respect to the gripper. Dafle *et. al.* demonstrated by hand-coded trajectories that a robot with a simple gripper is able to perform dexterous regraspings by utilizing the extrinsic force resources [8]. Dafle and Rodriguez then used LCP trajectory optimization for planning continuous regrasping motions [3] with extrinsic contact, as well as sampling-based planning for picking discrete modes [5]. The framework is general enough for covering re-orienting, however, sampling-based methods cannot declare infeasibility in finite time, while the nature of the planning problem (long horizon, hybrid states) makes the optimization hard to solve.

C. Motion Primitives for Reorienting

Works on pick-and-place reorienting can date back to 1980s, when the mainstream approaches discretize the whole gripper-object state space with a "grasp-placement table" [15], [16], [30]. The set of grasps were selected off-line. A motion sequence can be obtained by back chaining from the goal during planning. The approach is then adopted and improved in several ways. Stoeter *et. al.* computed the stable placements of the object and the discretized gripper motion, built the list of "Grasp-placement-grasp" triples for searching online [28]. Cho *et. al.* stores collision-free states in a lookup table, then find intermediate placement incrementally by querying the table [7]. Wan *et. al.* [32] and Xue *et. al.* [33] utilize graph structure to represent gripper&object feasible poses for efficient online searching. Wan *et. al.* also decompose the search of pick-and-place sequence from the search of grasps for better efficiency. In those works, only stable placements are discretized and stored as entries in the table/nodes on the graph. However, it would require too much memory to discretize the space of 3D poses, thus those methods only deal with the finite set of stable placements for an object.

Pivoting has been used as a motion primitive with several kinds of extrinsic actuations, including inertia force [10], [26], gravity [2], [22], [27], [31] as well as extrinsic contact forces [1], [9], [29], [34]. Using inertia force or gravity with grip force control can improve the speed of object motion, however, it poses high requirement on the bandwidth of the robot itself [10], [31], [27]. In [22], the robot lifted up the object and let the object rotate passively under gravity to the desired pose, assuming a suitable grasp along the line of gravity exists. Closely related to our work, Holladay [9] and Terasaki [29] used pivoting on the table for reorienting objects. However, in both works they only considered stable object placements, and analyzed pivoting as a swinging between two stable placements.

III. PROBLEM DESCRIPTION

Consider a rigid object. Denote the points on its convex hull as $\mathbf{P} \in \mathbb{R}^{3\times N}$, its center of mass (COM) as $c \in \mathbb{R}^3$. Here the $\mathbf{P}$ and c are measured under the original object orientation $q_{\text{obj}}^{(\text{o})} \in SO(3)$. We sample a collection of grasp positions $\mathbf{G}$ for the object off-line. A certain grasp position is denoted by an id $g \in \mathbf{G}$. The position of gripper p_{grp} is defined as the middle point between the two fingertips. The orientation of the gripper is denoted as $q_{\text{grp}} \in SO(3)$.

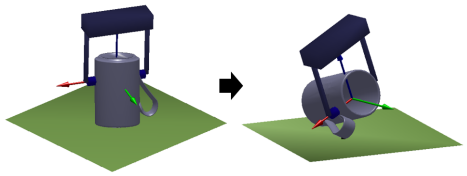


Fig. 1: Example of an Any-Pose-Reorienting problem. The arrows show the axes of the gripper frame.

An *Any-Pose-Reorienting* problem is defined as follows. Given the initial and final object poses (not necessarily stable poses), $q_{\text{obj}}^{(\text{i})}, q_{\text{obj}}^{(\text{f})} \in SO(3), p_{\text{obj}}^{(\text{i})}, p_{\text{obj}}^{(\text{f})} \in \mathbb{R}^3$, also the initial and final grasps and gripper orientations $g^{(\text{i})}, g^{(\text{f})} \in \mathbf{G}, q_{\text{grp}}^{(\text{i})}, q_{\text{grp}}^{(\text{f})} \in SO(3)$, plan a sequence of collision-free gripper motions to achieve the transition, as shown in Fig.1. Note we do not care about the translation of the object, as moving a grasped object on a clean table is trivial. The problem is a variant of regrasping problem, in that the goal specifies both the pose of the object about the gripper, and also their poses relative to the world.

We plan the gripper motion in Cartesian space. To avoid bad configuration of the manipulator, a dexterous workspace constraint is defined as follows. Consider the typical setup of an industrial robot, the gripper position has to stay in a bounding box ψ, while the gripper tilting angle has an upper limit $|\beta| < \pi/2$. Zero tilting angle corresponds to the case where the palm faces downward, with the grasp axis lying within the horizontal plane.

IV. APPROACH

We start with a list of assumptions used in this work.

1) A horizontal table is in the robot workspace.
2) Motions are quasi-static, i.e. inertia forces are negligible.

3) The pre-computed grasp positions are the actual positions of fingertips after the gripper being closed on the object.
4) The grasped object will not slip about fingertips, except for the rotation about grasp axis during a pivoting.
5) For friction, only Coulomb friction and stiction exist between the object and the table.
6) In terms of collision checking, we consider the gripper and ignore the robot arm. The workspace constraint should make sure the gripper has no collision with the table.

The first motion primitive we adopted is *pivoting on the table*. The gripper grasps two antipodal points on the object surface. The fingertips allow the object to rotate about the grasp axis, but do not allow tangential slipping. While contacting the gripper, the object also maintain contact(s) with the table under gravity, forming a kinematic chain. The object may or may not slip on the table, depending on the force exerted from the gripper. This motion primitive is the source of the additional flexibility of our method.

As we will show shortly, a pivoting grasp may not be stable under gravity, for which we need to switch to the second motion primitive, *compliant rolling on the table*. It involves the gripper firmly grasping the object and rotating while maintaining contact between the object and the table. A hardware implementation of the two motion primitives is explained in Sec. V.

These two motion primitives introduce the choice of mode $m = \{0, 1\}$ as an additional discrete decision variable in the problem.

A. Mechanics of Pivoting on the Table

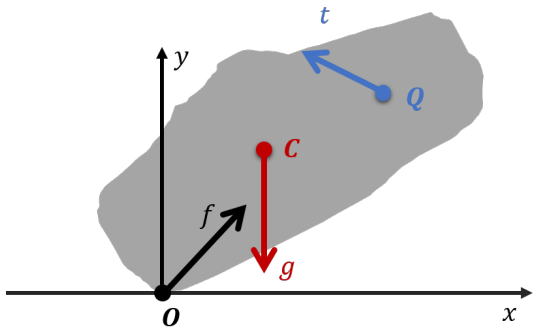


Fig. 2: Forces on the object during pivoting, viewing from the direction of grasp axis. Points O, C, Q are the contact point, center of mass and grasp point, respectively. Forces f, g, t are contact reaction force, gravity of object and force from gripper, respectively.

To simplify the analysis, in this work we fix the orientation of the grasp axis during pivoting. Then we can consider pivoting as a 2D motion in the plane perpendicular to grasp axis, as shown in Fig.2. Previous works on mechanics of pivoting on the table restricted the points O, C and Q to be co-linear, and focused on computing the required pushing force[1]. We remove this restriction and perform stability analysis for all possible pivoting scenarios.

Denote the Coulomb friction coefficient between the object and the table as μ. When the force system (f, g, t) is in equilibrium, we have (define $O_x = 0$)

$$\begin{aligned} f_x + t_x &= 0, \\ f_y + t_y &= g, \\ C_x g + Q_y t_x &= Q_x t_y. \end{aligned} \tag{1}$$

Inertia force is ignored due to the quasi-static assumption. Eliminate term t_x, t_y we have

$$f_y = \frac{Q_y}{Q_x} f_x + (1 - \frac{C_x}{Q_x}) g \tag{2}$$

A given pivoting scenario (specified by O, C, Q) is stable as long as a solution of (2) exists, otherwise the system is out of equilibrium and the object will topple over. We can discuss the existence of solutions by looking at the f_y - f_x curve.

Depending on the geometrical relations of points O, C and Q in the pivoting plane, we have six scenarios as shown in Fig.3. In each column, the top figure shows the positions of C and Q relative to O considered in that situation. The bottom figure shows the corresponding solution set of friction force. In all figures, we also show the friction cone at O (intersection of the 3D friction cone with the pivoting plane) in black dashed line.

The friction curves in Fig.3 tell us when the system will remain in static. For example, in scenario I, the object will be static if the gripper pushes towards contact point O, or pulls away with a small force. In scenario IV, the object cannot be static anyway.

There are two possible results when the object is non-static. In I, II, V and VI, the object will be sliding on the table. This is fine since we only care about the orientation. However, in III and IV, the object will topple over under gravity. We consider these situations as unstable. This observation leads to the following simple stability criteria:

Theorem 1. *A stable solution of pivoting system (1) exists as long as the following stability condition is satisfied:*

$$(Q_x - C_x) \cdot (Q_x - O_x) > 0 \tag{3}$$

i.e. in X direction, the gripper position Q is not in between contact point O and center of mass C.

Proof. The condition corresponds to I, II, V and VI in Fig.3. As per the analysis above, the object will always be either static or sliding without rotation. □

Condition (3) is not necessary. Notice that a solution exists for scenario III if the gripper pushes towards the contact point. We do NOT use this solution since it is hard to implement on hardware (requiring a certain amount of force). The condition described in theorem 1 has no requirement on force.

Using Fig.3, the following theorem tells us when the motion of the object will get stuck on the table:

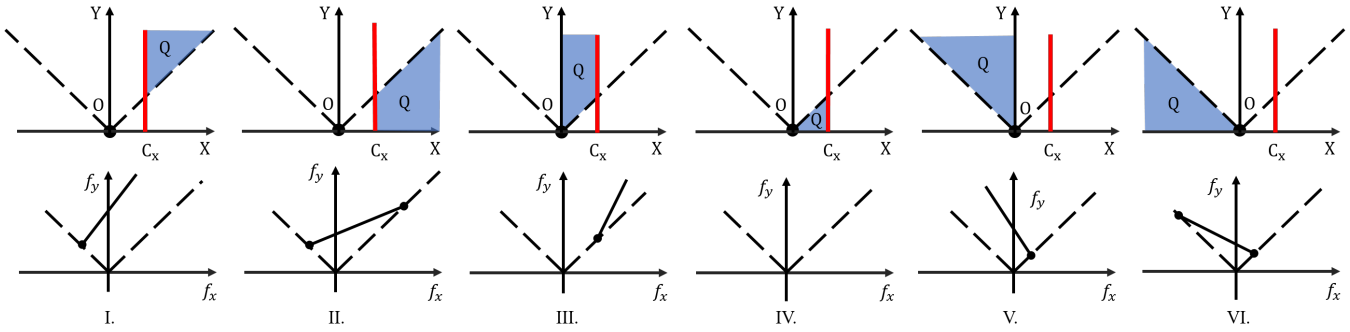


Fig. 3: All six possible scenarios and the corresponding solution sets of friction. In the first row of each sub-figure, O is shown as the black dot, x coordinate of C is shown by the red line, possible positions of Q are shown as the blue shaded area. Solid lines in second row show the possible value of friction force under force equilibrium. Dashed lines are the friction cone.

Theorem 2. *Under condition (3), the object can slide on the table during pivoting, as long as the grasp position Q is out of friction cone. When Q is within the friction cone, the object can only move such that the contact O moves towards Q in X direction.*

Proof. Under condition (3), Q being outside of friction cone corresponds to scenario II and VI in Fig.3, in which the frictions f_x in both directions are bounded. Q being inside of friction cone corresponds to scenario I and V in Fig.3, in which the friction f_x is bounded only when the gripper moves away from O. □

The two theorems make it possible to design simple yet effective planning algorithms.

B. Planning Any-Pose-Reorienting Given A Grasp

As a core module of our method, we firstly discuss how to solve the reorienting problem when the grasp position on the object is given. Our strategy contains two stages:

1) **Regrasping** Rotate the object with respect to the gripper till $q_{\text{obj}}^{-1} q_{\text{grp}} = (q_{\text{obj}}^{(\text{f})})^{-1} q_{\text{grp}}^{(\text{f})}$,
2) **Reorienting** Use rolling to rotate the object to final pose (gripper will also arrive the final pose).

The reorienting stage is trivial to plan. A gripper orientation trajectory generated by spherical linear interpolation(SLERP) will be feasible if the initial and final orientations are feasible.

In the regrasping stage, we fix the orientation of the grasp axis and let the object rotate about it. Note the initial grasp axis tilting angle should not be close to 90 degree, as pivoting would be hard due to insufficient gravitational torque.

Among the two motion primitives, only pivoting can change the pose of object with respect to the gripper. However, during the motion the pivoting may be unstable. Our strategy is to switch to rolling when we cannot ensure the stability condition, and do pivoting again when it is safe. Note that computing the switching point analytically is difficult since the contact position O is a discontinuous function of object orientation. As a result, we will be working with a discretized object rotation trajectory.

We now need to solve for a trajectory of continuous gripper motion as well as discrete choices of modes, under stability constraints, collision-avoidance (between object and gripper) constraints and gripper workspace constraints. The problem takes form of a Mixed-Integer-Programming which suffers from combinatorial explosion of modes. However, we can solve it much more efficiently by decomposing the planning into three levels:

- (Bottom Level) Object planning: compute a trajectory of object rotation and mode choices, rotate the object until the desired gripper orientation hit the further edge of tilting angle limit, while checking stability and workspace constraints;
- (Middle Level) Gripper planning: Given object motion, compute a trajectory of gripper rotations, achieve $q_{\text{obj}}^{-1} q_{\text{grp}} = (q_{\text{obj}}^{(\text{f})})^{-1} q_{\text{grp}}^{(\text{f})}$ and satisfy collision-avoidance constraints, tilting angle constraints;
- (Top Level) Full Cartesian planning: compute a trajectory of gripper translation satisfying the rest of workspace constraints.

The planning starts from the bottom to the top, rejects a solution immediately if one level is infeasible. In the following we explain each level in detail.

1) Object rotation planning: Firstly, we pick an object rotation direction based on $q_{\text{obj}}^{(\text{i})}$. We plan the object motion with algorithm 1. If the algorithm returns failure, we try the other direction. $\Delta\phi$ is chosen to be one degree in our implementation. $Rot(v, n, \theta)$ is the abbreviation for rotating vector v about axis n by θ. Here we can check the workspace constraints that are determined only by the fingertip positions. If there is no violation, we obtain a trajectory of object rotations and choices of modes.

2) Gripper rotation planning: To efficiently check the collision between the object and the gripper, we compute for each grasp position the array of collision-free gripper angles in the object frame during off-line computation. The array has an entry for every one degree. We can illustrate the gripper rotation planning problem in Fig.4. Object and gripper rotation are denoted by horizontal and vertical axis, respectively. Our task is to find a path from the initial gripper

Algorithm 1 Object rotation planning

Input : $q_{\text{obj}}^{(\text{i})}, q_{\text{obj}}^{(\text{f})}, q_{\text{grp}}^{(\text{f})}, \mathbf{P}$, grasp axis n

Compute the total rotation angle $\bar{\phi}$, discretize it into array Φ by $\Delta\phi$
Initialize motion mode array $M \leftarrow \emptyset$
for each element $\phi \in \Phi$ **do**
  Rotate object: $\hat{\mathbf{P}} \leftarrow Rot(\mathbf{P}^{(\text{i})}, n, \phi)$
  if Workspace constraint is violated **then**
    return failure
  end if
  Compute $m \in \{0(\text{rolling}), 1(\text{pivoting})\}$ using (3)
  $M \leftarrow [M, m]$
end for
return M, Φ

rotation angle to any point on the green line (where the desired relative angle is satisfied), while staying within tilting angle limit (grey dotted lines) and avoiding collision region (red shaded areas). Note that within the rolling zones (blue shaded areas), the change rates of gripper rotation and object rotation have to be the same. For computational efficiency, we solve for a path using a simple heuristics: use piecewise linear lines and always minimize gripper tilting. We omit the algorithm details here. We denote resulted gripper rotation plan as Ω, with the same dimension as Φ.

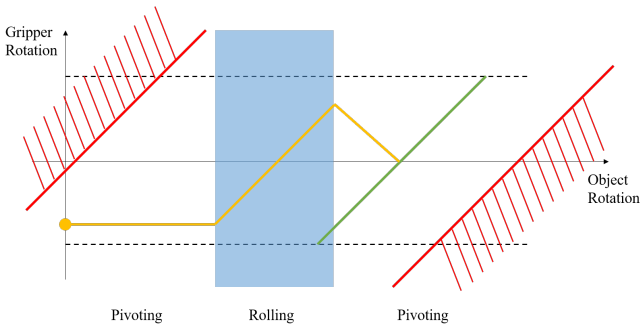


Fig. 4: Illustration of the gripper rotation planning problem in the gripper-object rotation space. An example solution is shown as the yellow line.

3) Full Cartesian planning: Finally we compute the 3D translations of the gripper for each time-step of motion. At the beginning of the algorithm, we re-sample the rotation plan $q_{\text{obj}}, q_{\text{grp}}, M$ to avoid large jumps in gripper rotations. For each sampled time-step, if sliding is possible (determined by theorem 2), we move the gripper towards the center of workspace. Otherwise the gripper position is decided by rotating the object about the current contact point on the table.

C. Robust Planning with Simplified Mesh Model

The computation in section IV-B involves manipulating the convex hull of the object mesh model. For delicate models with thousands of facets, the algorithm can be slow. We instead work with a simplified (triangle counts reduced) mesh model, and plan robustly against the modeling error.

Denote the Hausdorff distance between the original and simplified convex hull as d_H:

$$d_H(\mathbf{P}, \hat{\mathbf{P}}) = \max\{ \sup_{p\in\mathbf{P}} \inf_{\hat{p}\in\hat{\mathbf{P}}} d(\mathbf{P}, \hat{\mathbf{P}}), \sup_{\hat{p}\in\hat{\mathbf{P}}} \inf_{p\in\mathbf{P}} d(\mathbf{P}, \hat{\mathbf{P}}) \}, \tag{4}$$

we have the following theorem:

Theorem 3. *Denote $\hat{\mathbf{P}}$ as the vertices of the simplified mesh model under a certain object pose. Denote $a \in \hat{\mathbf{P}}$ as the bottom point on the simplified model, i.e.*

$$a_z = \min_{\hat{p}\in\hat{\mathbf{P}}} \hat{p}_z$$

Then the actual contact point $e \in \mathbf{P}$ between the original model and the table, defined by

$$e_z = \min_{p\in\mathbf{P}} p_z$$

must be within the following set:

$$e \in \left\{ B_{d_H}(\hat{p}_i) | \hat{p}_i \in \hat{\mathbf{P}}, \quad \hat{p}_{iz} \le a_z + 2d_H \right\}$$

$B_r(p)$ denotes the ball of radius r centered at point p.

Proof. The actual height O_z of the table surface is upper bounded by:

$$O_z \le a_z + d_H.$$

From the definition of Hausdorff distance we know that points on $\mathbf{P}$ is bounded in balls of radius d_H centered at points on $\hat{\mathbf{P}}$:

$$\mathbf{P} \in \left\{ B_{d_H}(p_i) | p_i \in \hat{\mathbf{P}} \right\}.$$

As a result, the balls that could possibly touch the table are centered at

$$\left\{ \hat{p}_i | \hat{p}_i \in \hat{\mathbf{P}}, \hat{p}_{iz} \le O_z + d_H \le a_z + 2d_H \right\}.$$

□

To plan robustly, for any condition in Section IV-B that involves the contact point O, we check it against the range of possible O instead.

There are many off-the-shelf algorithms for mesh simplification developed in computer graphics community [13], [14]. We use an algorithm similar to the one described in [14] to produce simplified mesh under user-defined Hausdorff distance.

D. Search for A Sequence of Reorientings

When we cannot finish the task within one reorienting, we need to search for a sequence of reorientings connected by place and pick at stable placements. Our method is built upon Wan's *Regrasp Graph*[32], in which a graph connecting different object-gripper poses is computed off-line. We improve Wan's work in two ways. Firstly, we connect two nodes in our graph as long as they have common grasp locations, rather than common grasp poses. Thus we remove the second layer in Wan's *Regrasp Graph*, only keep

the nodes correspond to stable placements. Secondly, we add the user specified initial and final object poses to the graph as the starting and ending node during planning time. Since our graph is very small, this step also add little time to online computation.

The off-line computation is shown in Algorithm 2. MeshSim denotes the mesh simplification described in Section IV-C. GraspSample and GetStable are the same as in [32].

Algorithm 2 Off-line Computation

Input : Object mesh model $\mathbf{S}$

$\mathbf{P} \leftarrow \text{convex}_\text{h}\text{all}(\mathbf{S})$
$\hat{P} \leftarrow \text{MeshSim}(\mathbf{P})$
Sample grasp points $\mathbf{G} \leftarrow \text{GraspSample}(\mathbf{S})$
Compute stable placements $\mathbf{V} \leftarrow \text{GetStable}(\hat{\mathbf{P}})$
for each $g \in \mathbf{G}$ **do**
  Compute collision-free angles for grasp g
end for
for each $v \in \mathbf{V}$ **do**
  Find the set of feasible grasp positions $\mathbf{G}_v \in \mathbf{G}$
end for
Compute the connectivity matrix $\mathbf{M}$ for $\mathbf{V}$.

The online search procedure is shown in Algorithm 3. We use Dijkstra to search for the shortest path, minimizing the number of grasp changes. For each edge on the path, we run planner for each grasp that the two nodes of the edge share. If none of them has a solution, we remove this edge from graph and run graph search again. Function OneGrasp() can be implemented as either our method (Section IV-B) or pick and place. We implement both for comparison.

V. EXPERIMENTS

A. Simulation and Comparison with Pick&Place

We simulate reorienting tasks on 12 objects with non-trivial shapes (over 2000 facets per object on average) obtained from Dex-Net [17], as shown in Fig.5. We sample at most 100 grasp positions for each object. After trimming similar grasps, around 20 to 80 remaining grasps are used in planning.

We normalize each object to fit into an $1m \times 1m \times 1m$ cube. For each object, we sample 100 Any-Pose-Reorienting problems, each with feasible initial and final gripper orientations. We count a full run of the 1200 problems as a batch. We run eight batches in total, with tilting angle limit ranging from 10 to 80 degrees. The performance of our method and pick&place are shown in Fig.6. Notice that our method can solve more problems under all conditions. As the tilting angle limit becomes tighter, the advantage of our method becomes more obvious. Average rotation and translation shown in Fig.6b and Fig.6c do not include the gripper motion when the object is not grasped, and do not include the gripper motion needed to lift the object off the table during pick&place. We can observe that our method takes similar rotations and more translation comparing with pick&place. Finally Fig.6d shows that our method requires significantly less number of breaking contact than what pick&place requires.

Algorithm 3 Online Searching

Input : $q^{(\text{i})}_{\text{obj}}, q^{(\text{f})}_{\text{obj}}, p^{(\text{i})}_{\text{obj}}, (q^{(\text{i})}_{\text{grp}}, q^{(\text{f})}_{\text{grp}}), \mathbf{V}, \mathbf{M}, \mathbf{G}$

$\mathbf{V} \leftarrow \{\mathbf{V}, v^{(\text{i})}, v^{(\text{f})}\}$, where $v^{(\text{i})} = q^{(\text{i})}_{\text{obj}}, v^{(\text{f})} = q^{(\text{f})}_{\text{obj}}$
Find feasible grasp $\mathbf{G}^{(\text{i})}, \mathbf{G}^{(\text{f})} \in \mathbf{G}$ for new nodes
Re-compute the connectivity matrix $\mathbf{M}$ for $\mathbf{V}$.
while true **do**
  $path \leftarrow \text{Dijkstra}(\mathbf{M}, v^{(\text{i})}, v^{(\text{f})})$
  Terminate with failure if $path = \emptyset$
  $plan \leftarrow \emptyset$
  for each edge $e_{jk} \in path$ **do**
    $\mathbf{G}_{jk} \leftarrow \mathbf{G}_j \cap \mathbf{G}_k$
    for each $g \in \mathbf{G}_{jk}$ **do**
      $plan_{jk} \leftarrow \text{OneGrasp}($
        $g, q^{(\text{j})}_{\text{obj}}, q^{(\text{k})}_{\text{obj}}, p^{(\text{j})}_{\text{obj}}, (q^{(\text{j})}_{\text{grp}}, q^{(\text{k})}_{\text{grp}}))$
      break if $plan_{jk} \neq \emptyset$
    end for
    if $plan_{jk} \neq \emptyset$ **then**
      $plan \leftarrow [plan\ plan_{jk}]$
    else
      Remove e_{jk} from $\mathbf{M}$
      $plan \leftarrow \emptyset$ and **break**
    end if
  end for
  Terminate with success if $plan \neq \emptyset$
end while

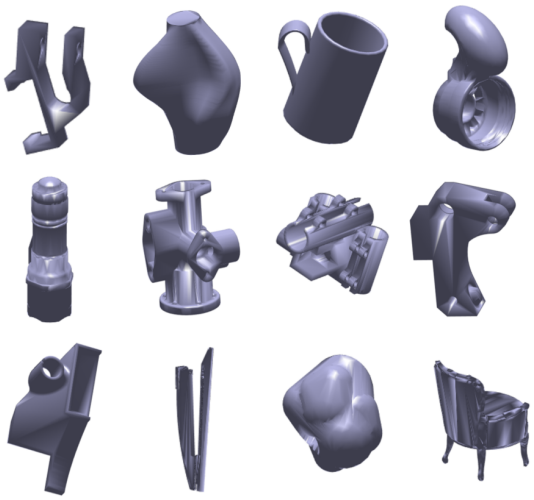
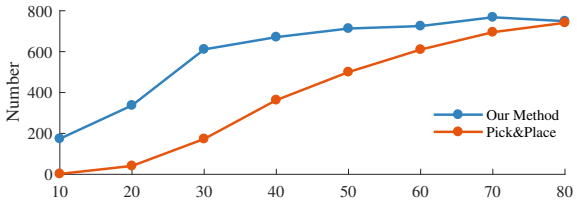


Fig. 5: The 12 objects used in our simulation.

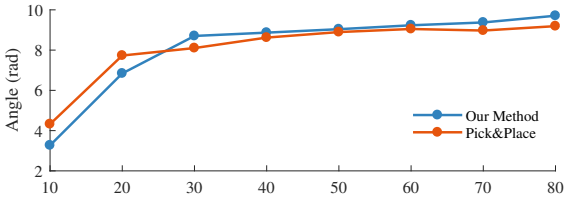
We haven't optimize the code for speed, yet the average computation time for solving one problem (or declare failure) is 43ms in single-thread Matlab, measured on a desktop with Intel Xeon 3.10GHz CPU. The off-line computation described in Algorithm 2 takes several minutes per object, depending on the complexity of the object shape.

B. Hardware Implementation

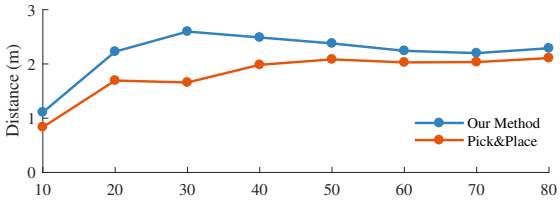
For experiments, we use an ABB IRB120 robot and a parallel gripper with a pair of specially designed "two-phase"



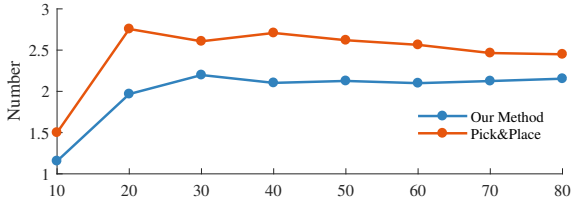
(a) Number of solved problems.



(b) Average gripper rotation per solved problem.



(c) Average gripper translation per solved problem.



(d) Average number of regrasps per solved problem.

Fig. 6: Simulation results. In all the sub-figures, the horizontal axes are the tilting angle limits in degrees.

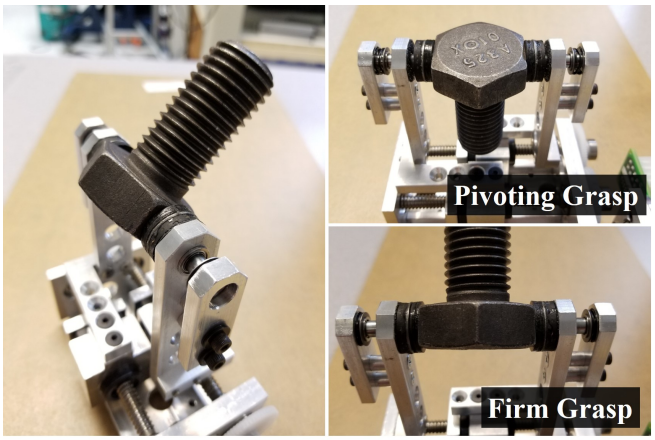


Fig. 7: Our customized gripper is able to switch between the firm grasp phase and the pivoting grasp phase.

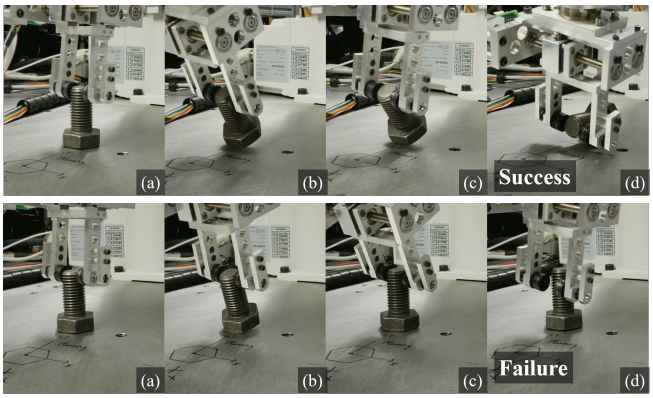


Fig. 8: Rotating a screw by 90 degrees. The first row shows a successful experiment. The second row is a failure caused by errors in the initial grasping position and the friction coefficient between the object and the table.

fingertips for performing both firm grasp and pivoting, as shown in Fig.7. Comparing with other designs [4], [29], our fingertips make it possible to maintain the same contact patches on the object during the pivoting phase and the switching between phases, eliminating a source of uncertainty. The contacts between fingertips and the object are always sticking, reducing the sensitivity of our method to friction coefficients.

For compliant rolling, the gripper grasps firmly while the robot performs force control in Cartesian space [18]. We control the gripper orientation by exact position, while letting the translation being determined passively by maintaining a certain force in the Z direction and moving freely in the X and Y directions. For pivoting, the fingertips rotate freely, the gripper executes the planned trajectory with position control.

Successful and failed solutions of an example problem are shown in Fig. 8. The goal is to rotate the screw by 90 degrees about the X axis. We constraint the maximum gripper tilting angle to be 40 degrees, under which limit the pick&place planner has no solution. Our method finds a solution without breaking contacts (one edge on the regrasp graph). In Fig. 8 (a) to (b), the solution starts with a rolling to move the object out of unstable scenario III to stable scenario V in Fig. 3. Then in subfigures (b), (c) and (d) of the first row, the solution switches to pivoting to finish the rest of object rotation without too much gripper rotation. In the bottom row of Fig. 8, the plan fails to move out of unstable scenario III before switching to pivoting, causing the object to topple over towards the wrong direction. This failure is caused by errors in the initial grasp position and an underestimation of the friction coefficient between the object and the table.

VI. DISCUSSION AND FUTURE WORK

One thing we learned in planning is that the existence of reorienting solutions depends heavily on the availability of grasps. A good number of grasp points distributed all over the object surface is crucial for both our method and pick&place.

In the current form, our method is not robust against uncertainties in grasp positions and object COM. Also, the experiment could fail if our estimation of friction coefficient between the object and the table is too low. If our estimation is too high, the planner becomes overly conservative and

could lose potential solutions. We are working on handling these deficiencies.

There are a few other interesting directions for future work. For example, how can we perform pivoting with normal fingertips (probably equipped with slip detection sensors) instead of a special mechanism. Slip is inevitable on normal fingertips, which means the planning algorithm needs to be robust against uncertainties caused by slip. Also, it is possible to integrate more motion primitives into our method, which would appear as additional connections in the "Regrasping Graph".

REFERENCES

[1] Y. Aiyama, M. Inaba, and H. Inoue. Pivoting: A new method of graspless manipulation of object by robot fingers. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 136–143 vol.1.

[2] David L Brock. Enhancing the dexterity of a robot hand using controlled slip. In *1988 IEEE International Conference on Robotics and Automation*, pages 249–251, 1988.

[3] N. Chavan-Dafle and A. Rodriguez. Prehensile pushing: In-hand manipulation with push-primitives. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6215–6222.

[4] Nikhil Chavan-Dafle, Matthew T Mason, Harald Staab, Gregory Rossano, and Alberto Rodriguez. A two-phase gripper to reorient and grasp. In *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, pages 1249–1255. IEEE, 2015.

[5] Nikhil Chavan-Dafle and Alberto Rodriguez. Sampling-based planning of in-hand manipulation with external pushes. *arXiv preprint arXiv:1707.00318*, 2017.

[6] I-Ming Chen and Joel W Burdick. Finding antipodal point grasps on irregularly shaped objects. *IEEE transactions on Robotics and Automation*, 9(4):507–512, 1993.

[7] Kyoungrae Cho, Munsang Kim, and Jae-Bok Song. Complete and rapid regrasp planning with look-up table. *Journal of Intelligent & Robotic Systems*, 36(4):371–387, 2003.

[8] N. C. Dafle, A. Rodriguez, R. Paolini, B. Tang, S. S. Srinivasa, M. Erdmann, M. T. Mason, I. Lundberg, H. Staab, and T. Fuhlbrigge. Extrinsic dexterity: In-hand manipulation with external forces. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1578–1585, 2014.

[9] Anne Holladay, Robert Paolini, and Matthew T Mason. A general framework for open-loop pivoting. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3675–3681.

[10] Yifan Hou, Zhenzhong Jia, Aaron M Johnson, and Matthew T Mason. Robust planar dynamic pivoting by regulating inertial and grip forces. In *The 12th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Springer, 2016.

[11] Kai Huebner, Steffen Ruthotto, and Danica Kragic. Minimum volume bounding box decomposition for shape approximation in robot grasping. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1628–1633. IEEE, 2008.

[12] Eric Jang, Sudheendra Vijaynarasimhan, Peter Pastor, Julian Ibarz, and Sergey Levine. End-to-end learning of semantic grasping. *arXiv preprint arXiv:1707.01932*, 2017.

[13] Alan D Kalvin and Russell H Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, 16(3):64–77, 1996.

[14] Reinhard Klein, Gunther Liebich, and Wolfgang Straßer. Mesh reduction with error control. In *Proceedings of the 7th conference on Visualization'96*, pages 311–318. IEEE Computer Society Press, 1996.

[15] Tomas Lozano-Perez, J Jones, Emmanuel Mazer, P O'Donnell, W Grimson, Pierre Tournassoud, and Alain Lanusse. Handey: A robot system that recognizes, plans, and manipulates. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 843–849. IEEE, 1987.

[16] Tomás Lozano-Pérez and Leslie Pack Kaelbling. A constraint-based method for solving sequential manipulation planning problems. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3684–3691. IEEE, 2014.

[17] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[18] J. Maples and J. Becker. Experiments in force control of robotic manipulators. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 695–702, Apr 1986.

[19] Andrew T Miller, Steffen Knoop, Henrik I Christensen, and Peter K Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1824–1829. IEEE, 2003.

[20] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3406–3413. IEEE, 2016.

[21] Nancy S Pollard. Closure and quality equivalence for efficient synthesis of grasps from examples. *The International Journal of Robotics Research*, 23(6):595–613, 2004.

[22] Anil Rao, David J Kriegman, and Kenneth Y Goldberg. Complete algorithms for feeding polyhedral parts using pivot grasps. *IEEE Transactions on Robotics and Automation*, 12(2):331–342, 1996.

[23] Máximo A Roa and Raúl Suárez. Computation of independent contact regions for grasping 3-d objects. *IEEE Transactions on Robotics*, 25(4):839–850, 2009.

[24] Jean-Philippe Saut, Mokhtar Gharbi, Juan Cortés, Daniel Sidobre, and Thierry Siméon. Planning pick-and-place tasks with two-hand regrasping. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4528–4533. IEEE, 2010.

[25] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.

[26] Jian Shi, J Zachary Woodruff, and Kevin M Lynch. Dynamic in-hand sliding manipulation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 870–877, 2015.

[27] Avishai Sintov, Or Tslil, and Amir Shapiro. Robotic Swing-Up regrasping manipulation based on the ImpulseMomentum approach and cLQR control. *Ieee T Robot*, 32(5):1079–1090, 2016.

[28] Sascha A Stoeter, Stephan Voss, Nikolaos P Papanikolopoulos, and Heiko Mosemann. Planning of regrasp operations. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 245–250. IEEE, 1999.

[29] Hajime Terasaki and Tsutomu Hasegawa. Motion planning of intelligent manipulation by a parallel two-fingered gripper equipped with a simple rotating mechanism. *IEEE Transactions on Robotics and Automation*, 14(2):207–219, 1998.

[30] Pierre Tournassoud, Tomás Lozano-Pérez, and Emmanuel Mazer. Regrasping. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 1924–1928. IEEE, 1987.

[31] Francisco E Vi, Yiannis Karayiannidis, Christian Smith, Danica Kragic, et al. Adaptive control for pivoting with visual and tactile feedback. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 399–406.

[32] Weiwei Wan, Matthew T Mason, Rui Fukui, and Yasuo Kuniyoshi. Improving regrasp algorithms to analyze the utility of work surfaces in a workcell. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4326–4333. IEEE, 2015.

[33] Zhixing Xue, J Marius Zoellner, and Ruediger Dillmann. Planning regrasp operations for a multifingered robotic hand. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 778–783. IEEE, 2008.

[34] Eiichi Yoshida, Mathieu Poirier, Jean-Paul Laumond, Oussama Kanoun, Florent Lamiraux, Rachid Alami, and Kazuhito Yokoi. Pivoting based manipulation by a humanoid robot. *Autonomous Robots*, 28(1):77–88, 2010.

[35] Yu Zheng and Wen-Han Qian. Coping with the grasping uncertainties in force-closure analysis. *The International Journal of Robotics Research*, 24(4):311–327, 2005.