Reorientating Objects with a Gripping Hand and a Table Surface

Weiwei Wan and Kensuke Harada

Abstract—We present a robotic system that can perform difficult reorientation tasks like flipping. The system uses a gripping hand and takes the advantages of external surfaces. It is composed by three components. In grasping planning, the system finds the grasps to fulfill force closure and resist high external wrenches. In placement planning, the system computes the stable placements of an object on the table surface and their associated grasps. In manipulation planning, the system builds a regrasp graph by connecting the stable placements and lazily searches the regrasp graph to find a reorientation sequence. The work is evaluated by thousands of simulation experiments and demonstrated by real-world executions. With the help of the presented implementation details, we physically validate the utility of an external table surface.

I. INTRODUCTION

Developing a robotic system that can reorientate objects using pick-and-place planning is the interest of many robotic companies. Once developed, users can replace the special-purpose parts feeders with flexible robots reorientate a wide range of parts with little or no extra cost. Commercially available reorientating systems concentrate on collision-free motion planning rather than parts reorientation. They fail in tasks where regrasp is needed. For instance, the Mujin Pick Worker1 can quickly pick a bolt or a ramen box, reorientate it to the goal configuration, and place it into a target tray; But it cannot flip the object. Flipping is widely required in object orientation. Take the task in Fig.1 for example. In order to reorientate the object from its initial configuration shown in the left to its goal configuration shown in the right, human beings flip the object with either the regrasp shown in Fig.1(a) or the in-hand manipulation shown in Fig.1(b). Nobody can finish this task with simple pick-and-place motion. Like human beings, once a robot picks up the object from its initial configuration, it cannot place it down due to the collision between the robotic hand and the placing surface. Robots need more intelligent manipulation skills like regrasp or in-hand manipulation. Besides Mujin, other companies that offer bin-picking services, for example Scape2, are also suffering from this problem. The systems produced by these companies could only finish the reorientation tasks where the initial and goal configurations share collision-free grasps. If the initial and goal configurations do not share any collision-free grasps, the systems fail.

Commercial robotic systems that can finish the task shown in Fig.1 are not available. For example, Mitsubishi developed a system3 that uses three robots to reorientate industrial parts. The positive aspect of this system is it can reorientate a part from any configuration to the expected one. The negative aspect is the system is composed of three manipulators and its scale is over large. We would like to implement a robotic system composed of fewer robots to finish difficult reorientation tasks.

Promising solutions can be found in academic publications. Some seminal studies include the regrasp planning introduced in [1], [2], and [3]. These early work uses a single manipulator to regrasp objects. Due to the limited computational ability and hardware platform at the old times, these initial work concentrates on searching algorithms and computational efficiency, and lacks real-world experiments. They are highly related to our requests but cannot fulfill it. More recent work like [4][5] studies regrasp planning more extensively and has more experiments. They concentrated on regrasp rather than reorientation. They did not discuss successful rates, time costs, or difficult orientation tasks like flipping. Other recent work like [6][7][8][9] uses regrasp to perform manipulation task. They use dual-arm robots and present interesting simulations and executions. The aim of their regrasp planning is to reduce space dimension, rather than reorientation and do not fulfill our requests. The framework proposed by Chavan-Dafle et al. [10] is the most inspiring study. Using a robot and an external fixture enables fewer robots to finish difficult reorientation tasks. Especially, we are interested in the reorientation problem of a gripping hand plus an external work surface, a gripping hand plus an external supporting pin, and a gripping hand plus an external gripper fixed on the environment, etc. This paper presents a robotic system that uses a gripping hand and an external table surface to perform such tasks.

1See the video at https://youtu.be/D6kmnIdi6Ik
2See the video at https://youtu.be/ekKACoIsXSo
3See the slides at http://gamma.cs.unc.edu/MPIR/ICRA2014-domae.pdf
The work is the further development of our previous studies on the utility of worksurfaces \cite{11}\cite{12} and the divide-and-conquer manipulation planning \cite{13}. It is composed of grasp planning, placement planning, and manipulation planning. In grasp planning, we compute all the grasps that fulfill force closure as well as resists to certain external wrenches. In placement planning, we take advantages of an external planar surface by computing all stable placements on it. Each stable placement will be associated with different collision-free grasps after this placement planning. In manipulation planning, we build a regrasp graph by connecting the planar placements that share common grasps and lazily search the graph to find a solution for the reorientation tasks. We analyze the performance of our proposal with the HiroNX robot and an external table. The work is evaluated by thousands of simulation experiments and demonstrated by real-world executions. With the help of the presented implementation details, we physically validate the utility of an external table surface.

Organization of this paper is as follows. First, we present the whole system diagram in Section II. After that, we discuss each component of the diagram, namely grasp planning, placement planning, and manipulation planning, in Section III, IV, and V respectively. Experiments and analysis are presented in Section VI, followed by conclusions and further works drawn in Section VII.

II. SYSTEM DIAGRAM

Our system uses a gripping hand plus an external table surface to finish difficult reorientation tasks. Given the initial and goal placements of an object on the planar surface, our system can find a sequence of regrasp motion to reorientate the object from the initial placement to the goal placement. The system is composed of three components – grasp planning, placement planning, and manipulation planning. Their relationship are illustrated in Fig.2.

Grasp planning is done in the local coordinate system of the object. It uses the mesh model of the gripping hand and the mesh model of the object to find all the grasps that fulfill force closure as well as resist to certain external wrenches. The output of this component would be a set of grasps in the object’s coordinate system.

Placement planning computes the stable placements of an object on a planar surface. It finds the local coordinate system of the supporting face, transforms the grasps from the object’s local coordinate system to the supporting face’s local coordinate system, and delete the grasps that collide with the planar surface. The output of this component would be a list of placements and the available grasps associated with each of them.

In manipulation planning, we build a regrasp graph using the placements and grasps computed from step one and two, and search the graph to find a manipulation sequence to perform the reorientation task. Collision Detection (CD), Inverse Kinematics (IK) checking, and Transition-based RRT (T-RRT) motion planning are performed lazily online during the search. The output of this component is a sequence of joint motions.

We carry out both simulation experiments and real-world executions. In simulation experiments, we use the Choreonoid platform\footnote{See http://choreonoid.org/en/ for Choreonoid} with graspPlugin\footnote{See http://choreonoid.org/GraspPlugin/i/?q=en for the graspPlugin}. In real-world executions, we use the Kinect and one arm of the HiroNX robot. The manipulation planning component offers an interface to accept the initial object positions and orientations estimated by the Kinect vision system. Details will be described in following sections.

III. GRASP PLANNING

The robotic hand used in our system is equal to a simple gripper. Therefore, the given conditions and expected outputs of our grasp planning problem is based on a gripping hand.

The given condition is: the model of a robotic gripper and an object. The output is: all robust force-closure grasps of the robotic gripper on the object. More formally, given the parameters of a robotic gripper (joints, finger pad positions, etc.) and the mesh model of an object, grasp planning computes all the \( \{ p^i, R^i, d^i \} \) pairs that can grasp the object with force closure and enough resistance to external wrenches.
Here \( p^h \) and \( R^h \) are the positions and orientations of the gripper. They are described in the object’s local coordinate system. \( d^h \) is the jaw width of the gripper. It is a scalar. The output of grasp planning is described as \( g = \{ g_1, g_2, \ldots \} = \{ (p_1^h, R_1^h, d_1^h), (p_2^h, R_2^h, d_2^h), \ldots \} \).

In detail, the grasp planning is composed of a surface sampling part and a force-closure computation part. They are summarized in Fig.3.

![Fig. 3. Surface sampling and force-closure computation. The original object is shown in (a). The result of the first step, namely the clustering step, is shown in (b). The result of the second step, namely surface sampling, is shown in (c). The result of further filtering is shown in (d). (e) illustrates the force-closure grasps at the first control points of the first cluster pair. The orientation is discretized into eight directions by parameter \( m \).](image)

The surface sampling part involves three steps. In the first step, we cluster the object surface using the algorithm presented in [14]. The algorithm is not only limited to objects with relatively large planar faces, but also applicable to noisy surfaces by considering surface thickness. It is specially designed for grasp planning. The surface of the object’s mesh model is clustered into several clusters after the first step. In the second step, we compute the first two principal axes \( \{ x_1, x_2 \} \) of each cluster, compute the bounding box \( \{ (e_1^{min}, e_1^{max}), (e_2^{min}, e_2^{max}) \} \) of the cluster on the plane defined by \( \{ x_1, x_2 \} \), and sample the cluster following \( p^i = \omega_1 x_1 + \omega_2 x_2, \omega_1 \in [e_1^{min}, e_1^{max}], \omega_2 \in [e_2^{min}, e_2^{max}] \). The granularity of \( \omega_1 \) and \( \omega_2 \) range from 0.01 to 0.02, depending on the sizes of mesh models and finger pads. After the second step, the object’s surface is sampled evenly, and a set of control points is obtained. In the third step, the sampled control points are further processed to ensure its robustness. First, we compute whether the sampled control points are on the cluster. If not, the control point will be deleted. Second, we compute whether the sampled control points have certain offset (0.01) from the cluster boundary. The control points near the cluster boundary will be deleted. Third, we compute whether the distance between the \( com \) (center of mass) of the object and the normal line of the control points are small enough. Large distances may induce high external wrenches during reorientation and are deleted to ensure enough resistance.

The force-closure computation part also involves three steps. In the first step, we find all the cluster pairs where the angle between the normals are larger than certain threshold \( \tau \). Mathematically, \( \tau \) should be smaller than \( \arctan \mu \) where \( \mu \) is the Coulomb friction coefficient between the robotic gripper and the object. In practice, we set \( \tau \) to a value near \( \pi \) to ensure robustness. After the first step, we get a list of candidate cluster pairs \( \{ c_i, c_j \} \). In the second step, we examine the control points on the cluster pairs. For each control point \( p_m^c \) on the cluster \( c_i \), we compute its projection \( p_m^{c,c_j} \) on the counter cluster \( c_j \) and check if \( p_m^{c,c_j} \) is inside \( c_j \). The control point whose projection is inside the counter cluster fulfills force closure and the pair \( \{ p_m^c, p_m^{c,c_j} \} \) will be kept as the force-closure grasping points. In step three, we compute the \( p_i^h, R_i^h \), and \( d_i^h \) using each force-closure grasping points and the model of the gripping hand. First, we decide the rotation of the hand with respect to the axis \( p_m^c - p_m^{c,c_j} \). Using the first two principal axes \( \{ x_1, x_2 \} \) of cluster \( c_i \), we compute the rotation of the hand \( R_i^h \) following \( \cos(\frac{\pi}{2}m)x_1 + \sin(\frac{\pi}{2}m)x_2, m = \{ 1, 2, \ldots, 8 \} \). Then, we compute \( p_i^h \) and \( R_i^h \) using \( \{ p_m^c, p_m^{c,c_j} \}, R_i^h \), and the gripper’s parameters like the position of finger pads, and compute \( d_i^h \) using the distance between \( p_m^c \) and \( p_m^{c,c_j} \).

IV. Placement Planning

The output of grasp planning is a set \( g \) described in the object’s local coordinate system. Using it, we compute the stable placements \( p = \{ p_1, p_2, \ldots \} = \{ (p_1^c, R_1^c), (p_2^c, R_2^c), \ldots \} \) of the objects and associate each \( p \) with the collision free grasps of \( g \). We name the associated grasps \( g_{p,c} \). It is a subset of \( g \) and is described in the local coordinate of each placement.

![Fig. 4. Procedure of placement planning. In the upper row, the stable placements of the object are computed and described in the supporting surface’s local coordinate system. In the lower row, the grasps computed in grasp planning are filtered and associated with each of stable placement.](image)

The placement evaluation part includes two steps. In the first step, we compute the convex hull of the original object mesh and perform surface clustering on the convex hull. This surface clustering process is the same as the surface clustering in grasp planning and its output is a list of clusters with each one has its own local coordinate system defined as \( R^c = [x_1 | x_2 | -n] \). Here, \( x_1 \) and \( x_2 \) are the two principal eigen vectors and \( -n \) is the inverse of the cluster’s normal. In the second step, we rotate the object to each \( R^c \) and compute the projection of its \( com \) on the cluster \( c_i \). If the projected \( com \), namely \( com^c \), is inside the cluster \( c_i \) and has a large enough offset from the boundary of \( c_i \), the placement on this cluster \( c_i \) is a stable placement, and its coordinates will be saved as

\*The steps are performed similarly with the control points on \( c_j \).
one element of \( p \). The \( p' \) value this element is defined as
the center of the cluster \( c_i \), and the \( R' \) value is defined as its
local coordinate system.

The grasp association part associates the \( g \) computed from
grasp planning to each of the stable placements and deletes
the colliding candidates. For each \( g_k \) in \( g \), we transform it
into the coordinate system defined by \( p_i = (p_i^t, R_i^t) \) following
\( p_i = R_i^o \cdot p_i^t, R_i^o = R_i^t \cdot R_k^t \) and \( d_i = d_k^o \), and check if
the mesh at \( p_i^o, R_i^o, d_i^o \) collides with the plane defined by
\( p_i^t \) and the first two columns of \( R_i^t \). If there is no collision,
the grasp \( g_k \) will be associated with the placement \( p_i \) – it
will be kept as an element of \( g_{p_i} \).

The output of placement planning will be used to build a
regrasp graph in manipulation planning.

V. MANIPULATION PLANNING

We build the regrasp graph with the \( g \), \( p \), and the \( g_{p_i} \)'s
computed in grasp planning and placement planning.

In the first step, we consider the grasps associated with
one placement. In that case, a robot can change freely from
one grasp to another. We connect the grasps associated with
the same placement and build a fully connected sub-graph.
Fig.5(a) illustrates this sub-graph. It is arranged into a circle.
Each vertex on the circle indicates one grasp. Each edge of
the sub-graph indicates the change between grasps. During
building the sub-graph, all vertices and edges are assumed to
be IK-feasible and CD-free. Exact IK and CD checking
will be performed later during the lazy search.

![Fig. 5. A regrasp graph is built on the grasps and placements computed in grasp planning and placement planning. First, we connect the grasps associated with the same placements and build a fully connected sub-graph. This step is shown in (a). Second, we connect the shared grasps of different placements and further connect the sub-graphs. This step is exemplified by the upper and lower rows of (b).](image)

In the second step, we consider the grasps associated with
different placements. The grasps associated with a certain
placement \( p_i \) are essentially a subset of the grasps in \( g \) plus
the transformation defined by \( p_i \). Ignoring the transformation,
if two grasps associated with two different placements are
transformed from the same \( g \), or namely if two grasps
associated with two different placements are the same in
the object’s local coordinate system, they are considered as a
shared grasp. Take Fig.5(b) for example. In both the upper
and the lower rows, the grasp in left plot is the same as the
grasp in right plot in the object’s local coordinate system.
It is a shared grasp of the two placements and we connect
the shared grasps associated with the different placements.
Graphically, this step interconnects those sub-graphs built in
the first step. Like the first step, all interconnecting edges
here are assumed to be IK-feasible and CD-free. Exact IK
and CD checking will be performed later during the lazy
search.

After these two steps, we build a regrasp graph composed of
inner-circle edges and inter-circle edges. The inner-circle
edges connect the grasps associated with the placement
represented by that circle. The vertices on the same circle
are fully connected with each other. The inter-circle edges
circle the shared grasps. The two vertices at the two ends
of the inter-circle edges indicate that the two grasps are the
same in the object’s local coordinate system. We lazily search
the regrasp graph to find a solution for reorientation tasks.

Fig.6(a) shows the regrasp graph of the object shown in
Fig.3(a). This regrasp graph is composed of four circles
where each circle represents one stable placement of the
object. This is the same as our observation in Fig.4 that only
the placements in Fig.4(a), (b), (e), and (f) have associated
grasps. No grasps are associated with the placements in
Fig.4(c) and (d) and no circles are plotted.

![Fig. 6. The regrasp graph of the object shown in Fig.3(a). (a) is the original one that connects both the grasps associated with each placements and the shared grasps associated with different placements. (b) is the extended one which has a new placement in the upper layer. The new placement encodes the initial position and orientation of the object.](image)

To use the regrasp graph, we extend the one shown in
Fig.6(a) by considering the initial placement of an object
which is defined as a position \( p_{init} \) and an incremental rotation
\( R_{init} \), from one of the stable placements, \( R^o \). We compute
the initial placement using \( p_{init} = (p_{init}^t, R_{init}^t)^t \) and the
associated grasps using \( g_{p_{init}} = \{g_1^p_{p_{init}}, g_2^p_{p_{init}}, \ldots, g_k^p_{p_{init}}\} \).
\( \{p_i^p_{p_{init}}, R_i^o \cdot \alpha_{init}, d^o \cdot \alpha_{init}\}, \ldots, \{p_j^p_{p_{init}}, R_j^o \cdot \alpha_{init}, d^o \cdot \alpha_{init}\} \} \), where
\( p_i^p_{p_{init}} = R_i^o \cdot (R^o_i \cdot p_i^t - p_{init}^t) + p_{init}^t, R_i^o \cdot \alpha_{init} = R^o_i \cdot R_{init}^t \cdot R^t_i, \) and \( d^o \cdot \alpha_{init} = d_i^o \). The computation is
performed online and is inserted as a new circle to the graph
in Fig.6(a). Fig.6(b) exemplifies the extended regrasp graph.
Here, the circle in the top layer represents the placement \( p_{init} \).
It is an incremental value of \( p_i \) and has the same vertices as
the first circle. In more general simulation experiments,
the \( p_{init} \) is randomly generated. In real-world execution
with vision systems, the \( p_{init} \) is detected by the vision system.
We use the Clustered Viewpoint Feature Histogram (CVFH),
the cameras roll histogram [16], together with planar con-
straints to perform model-based recognition. Details will be
discussed in the experiments section.

Given the initial and goal placements of the object, we use
lazy search [17] to find a path. We randomly pick a grasp in
the initial circle and see if there is path connecting it to any
of the vertices in the goal circle. Collision Detection (CD),
Inverse Kinematics (IK) checking, and Transition-based RRT (T-RRT) motion planning are lazily performed online during the search. If the search fails, we delete the vertices that suffer from CD or IK problems and search again. After times of iteration, the algorithm would either give a good motion sequence or tell us the expected reorientation is invalid. Interested readers may refer to the Fig.7 of [13] for the details of lazy searching.

VI. EXPERIMENTS AND ANALYSIS

We analyze the performance of our work with both simulation experiments and real-world executions using the objects shown in the upper-left of Fig.7. The objects are Fig.7(a) a wooden block, Fig.7(b) an aluminum part, and Fig.7(c) a plastic tube. Our robot platform is the HiroNX.

A. Simulation Experiments

During simulation, we set \( p_{\text{init}} \) to a fixed position in front of the robot’s right hand, and use a random rotation around the normal of the external table surface to generate \( R_{\text{init}} \). Screenshots of the simulation experiments can be found in Fig.8.

The results of simulation experiments are shown in Fig.7. The left column of this figure lists the initial placements and the upper row lists the goal placements. The \( R_{\text{init}} \) is added to the left column. For each pair of initial and goal placements, we generate 100 \( R_{\text{init}} \) and run our program for each of them. The average successful rate and the average time cost of successful runs are shown in each grid of Fig.7. The last row of each matrix plot shows the placements and their associated grasps. Readers may inspect the successful rates and time costs by considering the last row. The aluminum part has six stable placements but only four of them have associated grasps. All stable placements of the wooden block and the plastic tube have associated grasps.

The most time-consuming reorientation is the (row-1, column-2) of the wooden block. It takes 10.2s. That is because the program wasted much time to lazily test some grasps which become obstructed at the goal placement. The lowest successful rate appears at the (row-1, column-1) of the plastic tube. It is 38%. That is because the robot is trying to rotate the object around the normal of external table surface. Such reorientation is near singularity. Many other results at the diagonal grids of Fig.7 also suffer from this problem. They are marked with blue boxes.

The results of flipping are marked with red boxes in Fig.7. Our algorithm can find a flipping reorientation motion in a few seconds. The lowest successful rate of flipping is at the (row-1, column-5) of the wooden block. In this case, the robot cannot flip that wooden block at all. The most time-consuming flipping is at the (row-5, column-6) of the plastic tube.

One lesson we learnt from the simulation is planar contact fails. They are (1) the offset between control points and cluster boundaries, (2) the distance between the \( \text{com} \) of the object and the normal line of the control points, and (3) the offset between the projected \( \text{com}^{c} \) and the boundary of \( c_{i} \). Here, (1) and (2) are the parameters of grasp planning. (3) is the parameter of placement planning. In our experiments, we set all these parameters to 0.01m. This parameter value disables some placements. For instance, the “L” wooden block cannot stand on the long end. It lowers the connectivity of the regrasp graph and consequently lowers the successful rates of reorientation. But it enables successful real-world execution. Automatic tuning the parameters is an open problem.

B. Real-world executions

Screenshots of the real-world executions can be found in Fig.9.7. It needs much engineering work to implement the simulation results on real robots, and the major two lessons we learnt are as follows.

One lesson we learnt is Kinect with the single-modal CVFH feature is not precise enough for the regrasp reorientation. We use the planar constraints to correct noisy estimations. Fig.10 show an example. The position and orientation of the real object is shown in Fig.10(a). The raw estimation result is plotted in Fig.10(b). Comparing with the actual position and orientation, the raw result is quite noisy and is not stable on the planar surface. We correct the noisy raw estimation by rotating it to the nearest stable orientation shown in Fig.10(c). The estimation becomes usable after the correction.

Fig. 10. Correcting the results of vision estimation with planar constraints. We correct the noisy raw estimation of CVFH by rotating it to the nearest stable orientation on the planar surface. (c) shows the corrected result. (d) shows the associated grasps of the corrected pose.

The other lesson we learnt is values of some parameters are crucial to successful regrasp. They are (1) the offset between control points and cluster boundaries, (2) the distance between the \( \text{com} \) of the object and the normal line of the control points, and (3) the offset between the projected \( \text{com}^{c} \) and the boundary of \( c_{i} \). Here, (1) and (2) are the parameters of grasp planning. (3) is the parameter of placement planning. In our experiments, we set all these parameters to 0.01m. This parameter value disables some placements. For instance, the “L” wooden block cannot stand on the long end. It lowers the connectivity of the regrasp graph and consequently lowers the successful rates of reorientation. But it enables successful real-world execution.

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, we present a robot system that uses a gripping hand and an external table surface to perform difficult reorientation tasks like flipping. Simulation results demonstrate the efficacy and efficiency of the approach. Real-world executions demonstrate the feasibility of the system. In the future, we would explore more about robotic manipulation with external fixtures and compare them with dual arm manipulation.
Fig. 7. Upper-left images in this figure show the objects used in our experiments: (a) a wooden block, (b) an aluminum part, and (c) a plastic tube. The other part of this figure shows the results of simulation experiments. Each grid indicates the successful rate and time cost of a task where the robot reorientates the object from the configuration shown in the left column to the configuration shown in the upper row. Random rotations are added to the left column in each run.

Fig. 8. Screen shots of the simulation results where the robot reorientates the second object. In this case, the task requires the robot to flip the object. (d-e) is the regrasp procedure which corresponds to the inner-circle edges. (b-c) and (e-f) are the motion sequences that correspond to the inter-circle edges.

Fig. 9. Real-world results where the robot reorientates the first object. The robot has to regrasp as no grasp is associated with the short end of “L”.

REFERENCES