Dual-Arm In-Hand Manipulation Using Visual Feedback

Silvia Cruciani[1], Kaiyu Hang[2], Christian Smith[1] and Danica Kragic[1]

***Abstract*— In this work, we address the problem of executing in-hand manipulation based on visual input. Given an initial grasp, the robot has to change its grasp configuration without releasing the object. We propose a method for in-hand manipulation planning and execution based on information on the object's shape using a dual-arm robot. From the available information on the object, which can be a complete point cloud but also partial data, our method plans a sequence of rotations and translations to reconfigure the object's pose. This sequence is executed using non-prehensile pushes defined as relative motions between the two robot arms.**

I. Introduction

In-hand manipulation is the action of changing the pose of an object inside the hand without releasing it. While it is possible to achieve the desired in-hand motions using multi-fingered hands by exploiting the several degrees of freedom (DoF) that those end-effectors provide [1]–[3], we focus on the subset of in-hand manipulation problems that are executed with parallel grippers. These end-effectors have the advantage of being more robust and reliable, as well as being the most commonly available on the current robot platforms, but they are limited to a one dimensional motion.

To achieve in-hand manipulation with simple grippers it is possible to exploit the *extrinsic dexterity* [4] that leverages supports external to the gripper, such as gravity or contact surfaces. In our case, we choose to enhance the dexterity of the gripper by exploiting the additional DoF available in a dual-arm system: the object is adjusted inside one gripper by contacts and pushes exerted against the other gripper.

In our work, we focus on obtaining the correct sequence of pushes for in-hand manipulation based on the object's shape, so that the object is moved from an initial grasp configuration to the desired one. The information about the desired grasp configuration can come from grasp planners based on affordances, such as [5]. An application example is a robot that has to readjust the grasp on a novel object that has just been handed over by a human, or that was blindly picked from a bin. The grasp configuration is not proper for the object's use, and it must be adjusted for the robot to fulfill the desired task. In the example shown in Fig. 1, the robot has to adjust the current grasp on the handle of the hammer, for future use.

[1]S. Cruciani, C. Smith and D. Kragic are with the Division of Robotics, Perception and Learning, EECS at KTH Royal Institute of Technology, Stockholm, Sweden. {cruciani, ccs, dani}@kth.se

[2]K. Hang is with the GRAB Lab, Department of Mechanical Engineering and Material Science at Yale University, New Haven, CT USA. kaiyu.hang@yale.edu

This work was supported by the Swedish Foundation for Strategic Research project GMT14-0082 FACT.

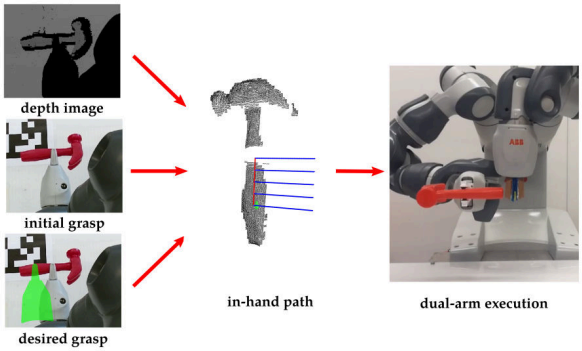


Fig. 1: Given the current grasp on the object and the desired in-hand manipulation task, the solution is obtained by analyzing the object's shape. In this case, the information about the object is provided by a depth image. The in-hand path is shown as a red line towards the desired point (in green), and the blue lines show the finger's orientation at each point. This path is then executed by the dual-arm system.

We plan an in-hand path based on the current available information on the object; while we focus on partial point clouds obtained from depth images, the method also works with precomputed 3D object's shapes that represent the whole object or a part of it. The planned in-hand path is associated to relative motions of the two robot's grippers that push the object towards the desired configuration.

We exploit the redundancy of a dual-arm robot to execute non-prehensile pushes on the object and adjust its pose inside the gripper. It is also possible to execute these pushes by using external contacts with surfaces or fixtures. However, in a context in which there are no properly placed fixtures or contact surfaces, or when the robot cannot fully perceive the environment to determine where to push, a dual-arm system is nonetheless able to perform the push execution.

The contribution of this work is a method for planning and executing dual-arm in-hand manipulation using non-prehensile pushing based only on the object's shape. The shape does not need to be known a priori and it does not have to be analyzed beforehand. The proposed solution can be used for quick corrections of the grasp, even in the absence of full information on the object. It is also suitable for manipulation tasks that require interactive perception.

II. Related Work

Our work consists of planning and executing an in-hand manipulation task with a parallel gripper, under the umbrella of extrinsic dexterity. Previous works mostly focused on one specific kind of repositioning, such as pivoting [6]–[8] or sliding [9]. We use both of these two motion types to achieve a broader grasp reconfiguration.

In our approach, the support external to the gripper is provided by the second arm of the robot. We describe the motion

of the object inside the gripper as a relative motion between the two arms and we use the Extended Cooperative Task Space (ECTS) [10] to control the robot's joints accordingly. While the second gripper could be exploited for regrasping, this operation is not always necessary or feasible, and it is not easy to plan in case of partially known objects. Therefore, we use the second gripper for non-prehensile pushing.

Dual-arm manipulation is often used for large object handling and not for fine manipulation. The works in [11] and [12] change the grasps on the object to ease the dual arm manipulation planning; however, the object is not moved within the gripper because all the grasps are firm and the final objective is not in-hand manipulation.

Bimanual manipulation of objects has also been used for tactile exploration [13], but in this case the system exploits multi-fingered hands equipped with several tactile sensors; the scope of the work is to find suitable stable grasp poses and not to execute an in-hand manipulation task.

The second arm used in our solution is similar to an external pusher that provides support to the gripper to manipulate the object. Planning in-hand manipulation with an external pusher has been explored in [14] and [15]. The main difference between these works and ours consists in the modeling needs: while in these works an accurate dynamic simulation is required, in our case we aim at achieving in-hand manipulation from the partial knowledge of the object's shape and pose provided by a depth camera.

In our previous work [16] we presented a method for planning in-hand manipulation based only on the shape of the object. However, the full object's shape must be known and it must be analyzed off-line to determine the possible motions of the fingers on its surface and generate a graph-like structure. In contrast, in this work we focus on in-hand manipulation given the current available information, without the need for an off-line step to analyze the object.

III. In-Hand Manipulation Planning

In this section we detail how the in-hand manipulation sequence is generated from the available input data. The data describing the object is a point cloud. This point cloud can be the result of prior knowledge on the object, for instance obtained from 3D reconstruction [17]–[20], or any currently available information from the sensors. In our examples, we focus on point clouds obtained from a depth image. When the knowledge on the object is partial, the in-hand path could result in an infeasible solution. This case can be addressed by increasing the information on the object as the manipulation task is executed; we discuss this possibility in section VI. As such, we prefer to not use a global planning method, and we favor an on-line approach that can be quickly modified and adjusted as new information becomes available.

A. Assumptions

We assume that the object is symmetric in the sliding area; i.e. if one fingertip can slide along a certain segment, the same applies to the opposite fingertip.

From the previous assumption, it follows that the grasp on the object is an antipodal point grasp; in fact, the two fingertips' contact points are antipodal points, i.e. two points whose normals with respect to the object's surface are collinear and in opposite directions. Antipodal points guarantee force closure under the condition of soft finger contact [21]. In addition, we assume that the manipulated object is sufficiently lightweight so that the grasp closure can be ensured with the limited force applicable by the robot's gripper.

Furthermore, we assume that the object's surface is sufficiently smooth so that in-hand sliding motions can be approximated with planar motions. The design of many objects and tools of common use falls in these assumptions. The obtained in-hand path represents how the gripper's finger can slide and rotate on the object, based on the current configuration and on the available push directions that the second arm can provide.

B. Planar Motion

While both the object and the grippers move in $SE(3)$, the in-hand manipulation inside a parallel gripper allows variations in $SE(2)$. In fact, assuming the frame of reference on the gripper's fingertip, the pushing motions enable rotations and translations on a plane, because the object's motion is constrained by the grasp of the parallel gripper.

In the following sections, we use both two-dimensional and three-dimensional vectors to analyze the points in the point cloud and the in-hand translational motion. To ease the disambiguation between the two kinds of vector, we use the notation $\mathbf{v} \in \mathbb{R}^2$ and $\tilde{\mathbf{v}} \in \mathbb{R}^3$. In particular, $\mathbf{v}$ is the projection of $\tilde{\mathbf{v}}$ on the plane of in-hand motion. In our case, $\mathbf{v}$ is expressed in the xy plane of the fingertip's frame, which is a plane normal to the closing direction of the fingers, and $\tilde{\mathbf{v}}$ is expressed in the global reference frame.

C. In-Hand Motions

We describe the current grasp of the object inside the gripper as a planar configuration $c{=}\langle \mathbf{p}, \alpha \rangle$, where $\mathbf{p}$ is the contact point between one fingertip and the object's surface and $\alpha \in [0, 2\pi)$ is the current orientation of the finger in the sliding plane. This configuration describes a pose defined by one of the two gripper's fingers. Since we use a parallel gripper and we assume symmetry in the sliding area, the second finger follows the motions of the first.

We define the motion of the object as a combination of two in-hand movements:

- **Rotation**. The object rotates around the axis $\tilde{\mathbf{z}}$ that connects the two fingertips of the gripper. The position of the contact point between the object and the fingertip does not change. More specifically, a rotation moves the grasp from a configuration $c{=}\langle \mathbf{p}, \alpha \rangle$ to a new configuration $c'{=}\langle \mathbf{p}, \alpha' \rangle$.
- **Translation**. The object slides inside the fingers. The position of the contact point between the object and the fingertip follows the translation. More specifically, it moves the grasp from $c{=}\langle \mathbf{p}, \alpha \rangle$ to $c'{=}\langle \mathbf{p}', \alpha \rangle$.

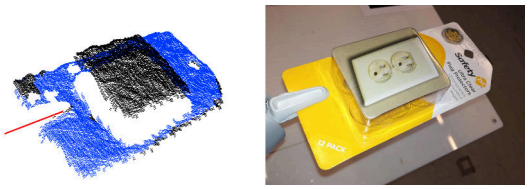


Fig. 2: An example of sliding area, considering the object shown on the right. The red line segment indicates the current grasp as the position and orientation of the gripper's finger. The blue area shows the portion of the object's surface on which the fingertip is allowed to slide.

By combining these two motions, we describe how the finger slides and rotates on the object's surface.

While we plan rotation and translation separately, there is the chance that the object slightly translates during a rotation and slightly rotates during a translation. Our control scheme manages one motion at a time, but these side effects can be compensated by using visual feedback after the termination of the first execution.

D. Point Cloud Analysis

Given $\tilde{\mathbf{p}}_0$, the 3D point describing the initial contact between the fingertip and the object, we extract a subset P_s of all the points in the point cloud $P \subset \mathbb{R}^3$ as

$$P_s = \{\tilde{\mathbf{p}}_i \in P \ : \ |\tilde{\mathbf{z}}^T \cdot (\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_0)| < \epsilon\}. \tag{1}$$

That is, P_s contains all the points of the point cloud that lie, within a certain threshold ϵ, on the plane π orthogonal to the axis $\tilde{\mathbf{z}}$ between the fingers and passing through the initial contact point $\tilde{\mathbf{p}}_0$. $\tilde{\mathbf{p}}_0$ does not have to belong to the point cloud. This set represents the area of the object along which the fingertip is allowed to slide. An example is shown in Fig. 2. The threshold ϵ accounts for imperfections in the objects and noise in the data. It can be adjusted to tolerate small curvatures for the sliding surface and how much the planar motion can be altered, for instance by taking into account the possibility of opening and closing the gripper's fingers during the execution.

Given a desired contact point $\tilde{\mathbf{p}}_d$, we define a goal area A_d as

$$A_d = \{\tilde{\mathbf{p}}_i \in \mathbb{R}^3 \ : \ \|\tilde{\mathbf{p}}_i - \tilde{\mathbf{p}}_d\| \leq \delta\}, \tag{2}$$

in which δ defines the tolerance in reaching the desired position. The set $P_d = \{\tilde{\mathbf{p}}_i \in P : \tilde{\mathbf{p}}_i \in A_d\}$ contains the points in the point cloud that belong to the goal area.

When the full 3D reconstruction of the object is not available and the desired region is not currently visible, i.e. $P_d = \emptyset$, we carry out the in-hand motion planning assuming that the desired point lies on the plane of motion, free of obstacles. That is, it is assumed that the desired grasp lies on the sliding plane π. Otherwise, if $|P_d| > 0$ and $P_s \cap P_d = \emptyset$, the desired grasp on the object is assumed not reachable by sliding or rotating along the object's surface and regrasping is needed; in this case, no in-hand motion is planned.

Once the region of motion is identified, given the initial configuration $c_0 = \langle \mathbf{p}_0, \alpha_0 \rangle$ and the desired configuration $c_d = \langle \mathbf{p}_d, \alpha_d \rangle$, describing poses on this region, the object's point cloud is analyzed to generate a sequence of rotations and translations to move the gripper's finger from c_0 to c_d.

E. Pushing Direction

For a given object, some motion directions can not be achieved using non-prehensile pushing. In fact, the available motion directions depend on the contact between the pusher and the object. More specifically, they can be identified by a motion cone [22]. However, to properly describe a motion cone detailed knowledge of the involved dynamic parameters is required. Therefore, our approach approximates the motion cone and limits the available motion directions based only on the currently observable object's shape.

We assume that a motion direction $\mathbf{d}$ can be achieved by pushing the object along $-\mathbf{d}$ if the difference between $\mathbf{d}$ and the normal $\mathbf{n}$ at the object's border used for pushing, parallel to the object's plane of motion, is within a certain threshold η. More specifically, assuming $\mathbf{n}_\perp$ as the vector orthogonal to $\mathbf{n}$, the pushing direction $-\mathbf{d}$ is feasible if $\frac{|\mathbf{n}_\perp \cdot \mathbf{d}|}{\|\mathbf{n}_\perp\|\|\mathbf{d}\|} \leq \eta$. The vector $\mathbf{n}$ can be obtained in two ways:

- If the object's shape is partially reconstructed and there is an opposite surface to the desired motion direction, the normal is simply the normal to this surface.
- If there is no available opposite surface, the normal is estimated based on the currently visible object's borders, which are used as an assumption for the surface's orientation.

By checking the availability of the pushing direction, the solution provided by the in-hand path planning also includes candidate push points for the bimanual pushing execution.

We distinguish between two different kinds of pushing: translational and rotational. While the feasibility of the pushing direction is obtained in the same manner, the pushing contact point differs in the two cases, as well as the tolerable mismatch between $\mathbf{d}$ and $\mathbf{n}$, which could be higher in the rotational case. Since we execute the in-hand manipulation with a dual arm robot, we consider push contact points valid only when they are sufficiently distant from the grasping gripper, and we prefer the ones that face the second arm because they are easier to reach.

1) Translational Pushing: The translational pushing contact is obtained by intersecting the object's border, or the available opposite surface, with the desired translation direction along a line that contains the current fingertip contact.

Fig. 3 shows examples of valid and invalid pushing contacts and directions. The finger, in dark gray, is in contact with the object at the point marked by the black dot. From this configuration, the green lines show some of the possible translation directions $\mathbf{d}_2$, $\mathbf{d}_4$. In fact, by pushing at the points shown by the green arrows, it is possible to slide the finger on the object in the chosen direction thanks to the alignment with the normal at the border $\mathbf{n}_1$, $\mathbf{n}_3$.

In contrast, the blue dotted lines show translation directions that cannot be achieved, because pushing as shown by the blue arrows is not considered a valid push due to the discrepancy between $\mathbf{d}_1$, $\mathbf{d}_3$ and $\mathbf{n}_1$, $\mathbf{n}_2$ in these cases.

The red arrow on the right, not associated with any line, despite having a pushing direction that is achievable because the normal at the border is aligned, does not show a valid

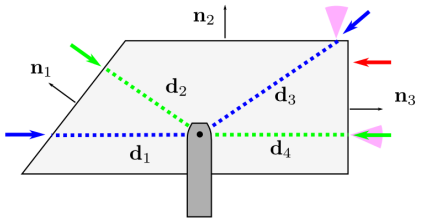


Fig. 3: Examples of valid and invalid translations for the fingertip contact point. Examples of valid pushes are shown by green arrows, and they result in translations along $\mathbf{d}_2$, $\mathbf{d}_4$. Blue arrows are non valid pushes, meaning that the fingertip cannot slide along $\mathbf{d}_1$, $\mathbf{d}_3$. The small magenta cones show the region defined by the angle η around the normal at the push point. The admissible pushes are inside this cone, while the inadmissible are outside. The red arrow shows a valid pushing direction, but it is not a valid push point for obtaining a translation because it does not lie on a line that intercepts the contact point.

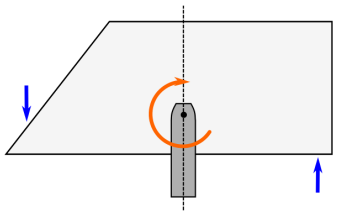


Fig. 4: Examples of push points for the rotation around the contact point shown by the orange arrow. The tolerance for alignment with the surface normal is not as strict as for the translation: the blue arrows show valid push points for this rotations. Notice that pushing as shown rotates the object in the opposite direction, therefore the finger rotates on the object as desired.

push point because it does not lie along lines that intersect the fingertip contact point. By choosing a specific push point instead of an object's side or face, we minimize the chance of unintended behavior, such as rotations during translations execution.

2) Rotational Pushing: The rotational pushing contact is obtained according to the rotation direction (clockwise or counter-clockwise), along a direction $\mathbf{d}'$ parallel to the current finger's configuration.

Fig. 4 shows an example of this pushing direction. The finger divides the object into a left and a right side. The direction of the push is along the finger's direction, but it depends on the object's side and on the rotation. In this case, since the desired rotation, shown in orange, is clockwise, the push is upwards if it lies on the right side and downwards if it lies on the left side. Since there are many valid push points, the choice is made to be the furthest away from the current fingertip contact and the side that faces the other robot's arm is preferred.

The tolerance η can be increased for the rotation push, as it is not necessary for the normal $\mathbf{n}$ to be strictly aligned with $\mathbf{d}'$. Similarly, the pushing direction itself is not required to be constrained. We search for possible push points starting from an initial candidate $\mathbf{d}'$ along the finger's direction, but if there are no suitable push points for it, this direction will be changed.

F. In-Hand Motion Planning

Given the desired grasp $c_d{=}\langle \mathbf{p}_d, \alpha_d \rangle$, an in-hand manipulation plan is successful when the gripper's finger reaches the orientation α_d and the fingertip contact lies within A_d.

To generate the desired in-hand motion we design a solution that always tries to move the grasp configuration as close as possible to the target. More specifically, it tries to move the fingertip contact towards the goal along the object's surface on the fastest possible direction and it rotates the object as early as possible to reach the desired angle. We favor this approach over a complete path planning algorithm due the lack of information on the object and the possibility of gathering it during the execution.

The planning goal is to find a sequence of rotations and translations $r^0, \mathbf{t}^0,\ r^1, \mathbf{t}^1\ \ldots,\ \mathbf{t}^{K-1},\ r^K$ that connects the initial configuration c_0 with the desired configuration c_d and that keeps the fingertip contact in the region defined by P_s. We assume that an initial rotation is executed at the beginning, and it is then followed by alternating translation and rotation. However, in many cases, during long sliding motions, consecutive translations follow the same direction and the rotations in between are 0. Therefore, the sequence of translations can be merged and the zero rotations discarded.

At a contact point $\mathbf{p}_k$, the translation direction is chosen as the feasible one closest to $\mathbf{d}{=}\mathbf{p}_d{-}\mathbf{p}_k$. The direction $\mathbf{d}$ is the one that goes towards the goal the fastest. The feasibility depends on the sliding area, on the available pushing directions and on the possible collisions that can arise between the contact point and the object and between the finger's body and the object.

In case of a collision of the contact point, the direction of the translation is rotated until it becomes a feasible motion, which is as close as possible to $\mathbf{d}$ in terms of angular distance between vectors. Notice that a change in direction of the translation does not impose any rotation to the object.

A rotation r^k is generated to move the fingers as close as possible to the desired final angle α_d, with a tolerance given by the angle resolution r_Δ. However, rotations are also introduced in case of collisions between the finger's body and the object: before changing the direction of the translation, a check on possible rotations is done to look for collision free finger's configurations at the given contact point. We execute this check by iteratively increasing and decreasing the current angle α_k by a fixed amount r_Δ. In this case, the rotation between the current angle α_k and the new angle $\alpha_k{+}r^k$ must be free of collisions in the whole circle sector spanned by the finger during the rotation. Algorithm 1 contains this procedure.

Starting from the initial configuration c_0, a sequence of rotations and translations is produced according to these steps:

1) move the contact point along a feasible translation, for a distance t_Δ, which corresponds to the closest admissible direction towards the goal.
2) rotate the finger as close as possible to the desired orientation.

This process is iterated until the contact point and the finger's angle reach the desired configuration c_d within the desired tolerance. However, we alter this simple planning sequence to obtain more sliding possibilities; the rotation

Algorithm 1: rotation_and_contact

Input : object's point cloud P, current angle α_c, desired angle α_d, angle resolution r_Δ, current contact point $\mathbf{p}_1$, next contact point $\mathbf{p}_2$
Output: rotation r, rotational push contact point $\tilde{\mathbf{p}}_{cr}$

if $\mathbf{p}_1$ is Null **then**
  if $\alpha_c \neq \alpha_d$ **then**
    $r \leftarrow$ feasible rotation closest to $\alpha_d - \alpha_c$ (resolution given by r_Δ)
    $\tilde{\mathbf{p}}_{cr} \leftarrow$ push point for r
  else
    $r \leftarrow 0$
    $\tilde{\mathbf{p}}_{cr} \leftarrow$ Null
else
  $r \leftarrow$ feasible rotation so that the finger is not in collision in $\langle \mathbf{p}_1, \alpha_c + r \rangle$ and $\langle \mathbf{p}_2, \alpha_c + r \rangle$
  if r *is Null* **then**
    return Null, Null
  $\tilde{\mathbf{p}}_{cr} \leftarrow$ push point for r
return r, $\tilde{\mathbf{p}}_{cr}$

Algorithm 2: find_admissible_reconfiguration

Input : object's point cloud P, current angle α_c, desired angle α_d, next angle α, angle resolution r_Δ, sliding distance t_Δ, current point $\mathbf{p}_1$, next point $\mathbf{p}_2$
Output: rotation r, rotational push contact point $\tilde{\mathbf{p}}_{cr}$, translation $\mathbf{t}$, translational push contact point $\tilde{\mathbf{p}}_{ct}$

if the finger's body is in collision in $\mathbf{p}_2$ **then**
  r, $\tilde{\mathbf{p}}_{cr} \leftarrow$ rotation_and_contact(P, α_c, α_d, r_Δ, $\mathbf{p}_1$, $\mathbf{p}_2$)
  if r is Null **then**
    go to 6
else
  $\mathbf{d} \leftarrow$ feasible direction so that $\mathbf{p}_1 + t_\Delta \mathbf{d}$ is admissible (resolution given by r_Δ)
  $\mathbf{t} \leftarrow t_\Delta \mathbf{d}$
  $\tilde{\mathbf{p}}_{ct} \leftarrow$ push point for $\mathbf{d}$
  $\mathbf{p} \leftarrow \mathbf{p}_1 + t_\Delta \mathbf{d}$
  r, $\tilde{\mathbf{p}}_{cr} \leftarrow$ rotation_and_contact(P, α, α_d, r_Δ, Null, Null)
return r, $\tilde{\mathbf{p}}_{cr}$, $\mathbf{t}$, $\tilde{\mathbf{p}}_{ct}$

can be affected not only by moving it towards the desired angle, but also to ease the translation in case of possible collisions. A translation $\mathbf{t}^k$ is infeasible at the configuration $c_k{=}\langle \mathbf{p}_k, \alpha_k \rangle$ if:

- The corresponding pushing direction $-\mathbf{d}^k{=}-\frac{\mathbf{t}^k}{\|\mathbf{t}^k\|}$ is not achievable given the current contact point.
- The region P_k in the point cloud that is the closest to the point $\mathbf{p}_{k+1}{=}\mathbf{p}_k{+}\mathbf{t}^k$ along the direction of $\tilde{\mathbf{z}}$ is so that $P_k \cap P_s{=}\varnothing$. In this case, the fingertip will enter in collision with the object.
- The finger's body at the configuration $c_{k+1}{=}\langle \mathbf{p}_{k+1}, \alpha_k \rangle$ enters in collision with the object.

In the first two cases, the translation direction will not be used and a new one should be selected. In the third case, instead, before proceeding with the selection of a new translation direction, the possibility of adding a rotation is also explored. In fact, if a rotation r^k is so that the finger is not in collision at the configurations $c'_k{=}\langle \mathbf{p}_k, \alpha_{k+1} \rangle$ and $c'_{k+1}{=}\langle \mathbf{p}_{k+1}, \alpha_{k+1} \rangle$, with $\alpha_{k+1}{=}\alpha_k{+}r^k$, and the circle

Algorithm 3: in_hand_path

Input : object's point cloud P, initial grasp c_0, desired grasp c_d, sliding distance t_Δ, angle resolution r_Δ, maximum number of iterations max_it
Output: sequence of translations T, sequence of rotations R, set of translational push points P_{ct}, set of rotational push points P_{cr}

$P_s \leftarrow$ from (1)
$A_d \leftarrow$ from (2)
$P_d \leftarrow A_d \cap P$
if $P_d \cap P_s = \varnothing \wedge P_d \neq \varnothing$ **then**
  return Null
$\mathbf{p}_c, \mathbf{p} \leftarrow \mathbf{p}_0$
T, R, P_{ct}, $\mathrm{P}_{cr} \leftarrow [\,]$
r^0, $\tilde{\mathbf{p}}^0_{cr} \leftarrow$ rotation_and_contact(P, α_0, α_d, r_Δ, Null, Null)
R.insert(r^0)
P_{cr}.insert($\tilde{\mathbf{p}}^0_{ct}$)
$\alpha_c, \alpha \leftarrow \alpha_0 + r^0$
$k \leftarrow 1$
while $(\tilde{\mathbf{p}} \notin A_d \vee \alpha \neq \alpha_d) \wedge k < \max\_\mathrm{it}$ **do**
  $\mathbf{d}^{k-1} \leftarrow$ feasible direction closest to $\mathbf{p}_c - \mathbf{p}_d$ (resolution given by r_Δ)
  $\mathbf{p} \leftarrow \mathbf{p}_c + t_\Delta \mathbf{d}^{k-1}$
  if ***p** is admissible* **then**
    $\mathbf{t}^{k-1} \leftarrow t_\Delta \mathbf{d}^{k-1}$
    $\tilde{\mathbf{p}}^{k-1}_{ct} \leftarrow$ push point for $\mathbf{d}^{k-1}$
    r^k, $\tilde{\mathbf{p}}^k_{cr} \leftarrow$ rotation_and_contact(P, α, α_d, r_Δ, Null, Null)
  else
    r^k, $\tilde{\mathbf{p}}^k_{cr}$, $\mathbf{t}^{k-1}$, $\tilde{\mathbf{p}}^{k-1}_{ct} \leftarrow$ find_admissible_reconfiguration(P, α_c, α_d, α, r_Δ, t_Δ, $\mathbf{p}_c$, $\mathbf{p}$)
  T.insert($\mathbf{t}^{k-1}$)
  P_{ct}.insert($\tilde{\mathbf{p}}^{k-1}_{ct}$)
  R.insert(r^k)
  P_{cr}.insert($\tilde{\mathbf{p}}^k_{cr}$)
  $\mathbf{p}_c \leftarrow \mathbf{p}$
  $\alpha_c \leftarrow \alpha$
  $\alpha \leftarrow \alpha_c + r^k$
  k+=1
if $k \geq$*max_it* **then**
  return Null
return T, R, P_{ct}, P_{cr}

sector spanned by the finger in $\mathbf{p}_k$ between α_k and α_{k+1} is free of collisions, the translation can be executed. This reconfiguration of the finger is described in Algorithm 2.

The process of obtaining an in-hand path is summarized in Algorithm 3 and the output solution is described in detail in the following section.

G. In-Hand Manipulation Solution

Given the initial grasp configuration c_0 and the desired one c_d, the in-hand manipulation solution is composed of:

- A sequence of $K{-}1$ translations $\mathbf{t}^0, ..., \mathbf{t}^{K-1}$ so that $\mathbf{p}_d{-}\mathbf{p}_0{=}\sum_{k=0}^{K-1} \mathbf{t}^k$. That is, by applying this sequence of translations the contact point on the object will be moved from $\mathbf{p}_0$ to $\mathbf{p}_d$.
- A sequence of K rotations $r^0, ..., r^K$ so that $\alpha_d{-}\alpha_0{=}\sum_{k=0}^{K} r^k$. Each rotation must be executed at every new contact point; more specifically, the rotation

r^k is executed when the fingertips are at the contact point $\mathbf{p}_k$=$\mathbf{p}_{k-1} + \mathbf{t}^{k-1}$.

- A sequence of $K-1$ contact points $\tilde{\mathbf{p}}_{ct}^0, ..., \tilde{\mathbf{p}}_{ct}^{K-1}$ on the object for pushing to obtain a translational motion. These contact points are so that the expected outcome of executing a pushing action is the desired translation.
- A sequence of K contact points $\tilde{\mathbf{p}}_{cr}^0, ..., \tilde{\mathbf{p}}_{cr}^K$ on the object for pushing to obtain a rotational motion. When the rotation r^k is 0, the corresponding contact point is set to a null value, because this rotation does not require execution.

The overall plan contains a sequence of $N \leq 2K-1$ pushes for the robot to execute. The total number of pushes can differ from the length of the translation and rotation sequences because, as mentioned in section III-F, some consecutive translations can be obtained with a single push, and rotations of 0 need no execution.

Since the pushing contact points are estimated from a partial reconstruction of the object, their location may not be always the real contact that the robot will obtain when approaching the object. The approach to this situation is described in section IV-B.

IV. Dual Arm Manipulation Strategy

Similarly to our previous work [16], we use a dual arm robot to execute the planned in-hand manipulation. One gripper is used to hold the object and the other one is used as a non-prehensile pusher.

A. Relative Arm Motions

The desired motion of the object inside the gripper is used to describe the desired relative motion between the two robot arms. These motions are expressed assuming the xy plane as the plane of motion in the grasping gripper's frame.

Given a desired translation $\mathbf{t}$ of the object, which is opposite to the desired translation of the fingertip, the relative motion of the two grippers is described by the Cartesian velocity and angular velocity, assuming Euler angles representation

$$^g\dot{\mathbf{x}}_r = (v_t\hat{t}_x,\ v_t\hat{t}_y,\ 0,\ 0,\ 0,\ 0)^T, \tag{3}$$

where v_t is the desired magnitude and $\hat{\mathbf{t}} = (\hat{t}_x,\ \hat{t}_y)^T$ is the planar direction of the translation with unitary norm.

Likewise, given a desired rotation r of the object, we describe the relative motion of the grippers in the grasping gripper frame as

$$^g\dot{\mathbf{x}}_r = (-v_r \sin(\phi+\phi_r)\dot{\phi}_r,\ v_r \cos(\phi+\phi_r)\dot{\phi}_r,\ 0,\ 0,\ 0,\ \dot{\phi}_r)^T, \tag{4}$$

where v_r is the desired magnitude, ϕ is the initial angle between the grippers at the pushing contact point and $\dot{\phi}_r$ is its variation. Given T_s the time in which the rotation starts and T_e the time in which it ends, the variation of the angle is so that $\phi_r(T_s) = 0$ and $\phi_r(T_e) = r$.

The relative velocity $^g\dot{\mathbf{x}}_r$ is then expressed in the robot's base frame as $\dot{\mathbf{x}}_r$ and it is converted into the robot's joint velocities $\dot{\mathbf{q}}$ by the ECTS framework [10] using the relation

$$\mathbf{J}\,\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{I}_6 & -(1-a)\mathbf{I}_6 \\ \mathbf{I}_6 & a\mathbf{I}_6 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{x}}_a \\ \dot{\mathbf{x}}_r \end{bmatrix}, \tag{5}$$

where $\mathbf{J}$ is the Jacobian of the robot, which includes both arms, $\mathbf{I}_6$ is the 6-dimensional identity matrix, $\dot{\mathbf{x}}_a$ is an absolute velocity that can be imposed without affecting $\dot{\mathbf{x}}_r$ and $a \in [0, 1]$ is the parameter that manages the degree of coordination between the arms.

B. Robot's Motion Sequence

In order to execute the planned in-hand manipulation, the robot follows these steps during the sequence of rotations and translations:

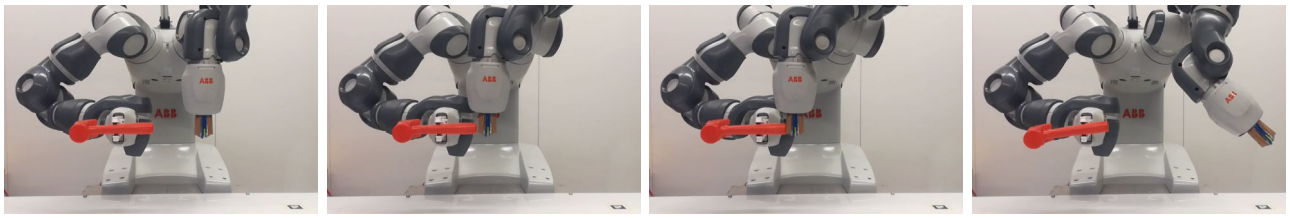
1) **Move to the approach pose**. This approach pose is so that the gripper that will be used for pushing is close enough to the object, but not yet in contact.
2) **Approach the contact point**. The gripper is moved towards the contact point with the object. Since occlusions or missing parts of the object in the point cloud can lead to a wrong estimate of the contact point, the gripper moves slowly and stops if contact with the object is reached earlier than expected.
3) **Push**. The two arms move so that the relative motion between the two grippers pushes the object in the desired direction (rotation or translation). The grasp on the object is kept loose to enable in-hand sliding when executing a translational motion.
4) **Move back**. The gripper used for pushing is moved away from the object. If the in-hand path is composed by more translations and more rotations, each of them is executed in sequence starting again from point 1.

At the end of one push execution, to correct for possible mismatches between the desired and the executed in-hand motion, it is possible to exploit visual feedback and plan for corrections. For instance, given a sequence of N pushes, the partial result of $n \leq N$ pushes can be checked and the in-hand motion plan updated if the mismatch with the desired outcome is found to be significant.

V. Experiments

We used the proposed method to execute in-hand manipulation tasks with an ABB Yumi robot. This robot, including the grippers, has a load capacity of 250 grams and a maximum gripping force of 20 Newtons. We printed slightly deformable hemispherical fingertips for Yumi's fingers; the deformation allowed the grasped force to be adjusted by moving the fingers, but the resulting contact area was still sufficiently small to ensure that the contact between the fingertips and the object could be approximated with a single point. The depth image was obtained from a Kinect v2 mounted on top of the robot; the Kinect's pose was calibrated with respect to the robot's base frame. To remove the robot's shape from the depth image, we used the package realtime_urdf_filter [23].

During our experiments, we set $\dot{\mathbf{x}}_a$ to the zero vector. We kept the grasping gripper fixed and only the pusher gripper was being moved, which corresponds to a choice of a=1.



(a) Move to the approach pose. **(b)** Approach the contact point. **(c)** Push. **(d)** Move back.

Fig. 5: The in-hand manipulation execution with Yumi, following the steps defined in section IV-B. The second gripper, on the right, keeps its fingers closed. The different color of the fingers is only due to the different color in the 3D printer plastic used.

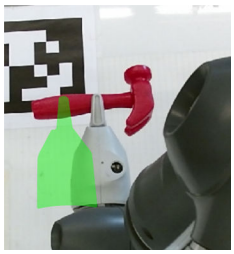


Fig. 6: The initial grasp configuration of Yumi holding the hammer. The desired grasp is shown in green.

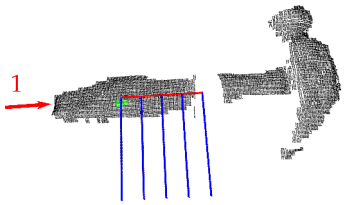


Fig. 7: The proposed in-hand path. In this case, only one push direction, labeled 1, is necessary to achieve the desired in-hand manipulation task.

Fig. 8: The initial grasp configuration of Yumi holding the glue. The desired grasp is shown in green.

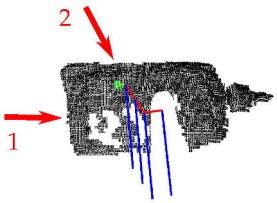


Fig. 9: The proposed in-hand path. Since a direct pushing from initial to goal contact point is not considered valid, the in-hand path uses two separate pushing directions.

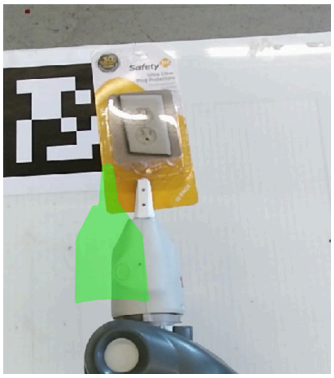


Fig. 10: The initial grasp configuration of Yumi holding the plug box, and the desired configuration, shown in green.

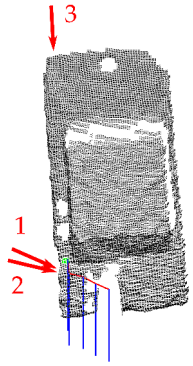


Fig. 11: The proposed in-hand path for the plug box object, with three pushes.

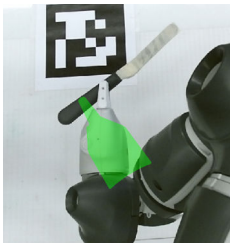


Fig. 12: The initial grasp configuration of Yumi holding the spatula. The desired grasp is shown in green, and it involves both a rotation and a translation.

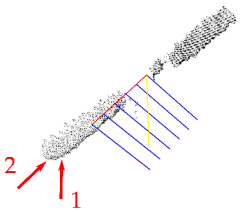


Fig. 13: The proposed in-hand path. The initial grasp pose is shown in yellow. Two push points are identified, one for rotation and one for translation in red. The finger is immediately rotated to the desired angle, and then translated.

Since this behavior depends only on the chosen parameters in the ECTS framework, it can be modified by changing the degree of coordination between the two arms. For instance, with a=0 only the grasping gripper would move. This case, with the pusher kept fixed, is equivalent to a single-arm system pushing against an external contact. Therefore, an external contact can be modeled in our framework by exploiting the possibility of changing the parameter a.

We used everyday objects to test the generation of the in-hand path with irregular shapes. We used the following quantities in our procedure for obtaining the in-hand path: ϵ=0.008, δ=0.02, η=0.06, r_Δ=0.17 radians, t_Δ=0.01 m. The initial contact $\tilde{\mathbf{p}}_0$ was derived from forward kinematics. The goal configuration was set as relative pose w.r.t. the initial grasp pose, and A_d was derived around the resulting $\tilde{\mathbf{p}}_d$.

Fig. 6 shows an example of desired in-hand manipulation task: the robot is grasping a hammer, and it has to adjust the grasp on the handle for a proper use. The desired grasp is shown in green. The marker on the table is used to check the calibration of the camera with respect to the robot when the system is turned on. Fig. 7 shows the proposed solution obtained from analyzing the depth image. The translation of the fingertip on the object is shown in red, and the blue lines show the direction of the gripper's fingers along the path. This desired motion is executed with the help of the second robot's gripper that pushes the object, as shown in Fig. 5.

Figs. 8 and 9 show an example in which the translation from the initial contact point to the desired one is not achievable with a single push. In fact, the motion direction from the initial contact to the desired one is not considered valid due to the shape of the bottle of glue. Therefore, the suggested in-hand path moves the contact point as close as possible to the desired position, while following motion directions that can be obtained with pushes on the object's side. The first push slides the fingertip towards left. Only after the first push it is possible to execute a second push towards the goal.

Similarly, Figs. 10 and 11 show the in-hand manipulation solution to modify the grasp on a plug package. In this case, the proposed solution includes three different push points, although two of them are very close to each other in location and also in the push direction.

Figs. 12 and 13 show an example of in-hand manipulation task that requires both translation and rotation. The proposed solution rotates the finger immediately to the desired orientation, since it is achievable at the initial contact point, using a first push. Then, it translates the fingertip towards the desired position with the new finger's orientation with a second push.

During the dual-arm in-hand manipulation execution, we found that the main challenge was in accurately pushing an object after a large rotation or translation, such as the example in Fig. 13. In this case, since we noticed that the errors between two consecutive pushes was negligible, the in-hand path was not updated during the execution. However, since the pushing points are estimated on the object in the initial frame of reference, small mismatches would prevent the gripper to reach perfectly the desired push point on the object after the execution of a few pushes. Therefore, despite the final error being negligible, it is preferable to keep updating the in-hand path after a few pushes to obtain a smooth execution.

VI. Conclusions and Future Work

We proposed a method to achieve bimanual in-hand manipulation using the currently available information on the object's shape. Our method provides a sequence of pushes that move the object inside the gripper towards the desired configuration, ans it exploits a dual-arm robot to execute these pushes.

To obtain in-hand manipulation with more general objects, and also under more uncertainty in the object's shape, we plan to extend this work using an interactive perception approach: the object information is increased at the same time as the object is manipulated. This addition will also ease the correction of the in-hand motion plan during the execution of several subsequent pushes.

Moreover, due to the rich contact interaction, we plan to integrate tactile sensing in the gripper to enhance the control of the in-hand motion execution; this addition also allows us to explore solutions to merge visual and tactile inputs to obtain a better object model and manipulation. More specifically, by having an estimate of the involved forces, apart from achieving better grips at the contact point during sliding motions, we can exploit the ECTS absolute motion to move the robot's arms so that the object is kept stable while the pushes are still executed to obtain the desired in-hand motion.

References

[1] B. Sundaralingam and T. Hermans, "Relaxed-rigidity constraints: In-grasp manipulation using purely kinematic trajectory optimization," in *Proceedings of Robotics: Science and Systems*, July 2017.

[2] OpenAI, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[3] K. Hang, M. Li, J. A. Stork, Y. Bekiroglu, F. T. Pokorny, A. Billard, and D. Kragic, "Hierarchical fingertip space: A unified framework for grasp planning and in-hand grasp adaptation," *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 960–972, Aug 2016.

[4] 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)*, May 2014, pp. 1578–1585.

[5] M. Kokic, J. A. Stork, J. A. Haustein, and D. Kragic, "Affordance detection for task-specific grasping using deep learning," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov 2017, pp. 91–98.

[6] Z. J. Yifan Hou and M. T. Mason, "Fast planning for 3d any-pose-reorienting using pivoting," in *International Conference on Robotics and Automation (ICRA) 2018*. IEEE Robotics and Automation Society (RAS), May 2018, pp. 1631–1638.

[7] F. E. Viña, Y. Karayiannidis, C. Smith, and D. Kragic, "Adaptive control for pivoting with visual and tactile feedback," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 399–406.

[8] S. Cruciani and C. Smith, "In-hand manipulation using three-stages open loop pivoting," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 1244–1251.

[9] J. Shi, J. Z. Woodruff, and K. M. Lynch, "Dynamic in-hand sliding manipulation," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 870–877.

[10] H. A. Park and C. S. G. Lee, "Dual-arm coordinated-motion task specification and performance evaluation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 929–936.

[11] Z. Xian, P. Lertkultanon, and Q. C. Pham, "Closed-chain manipulation of large objects by multi-arm robotic systems," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 1832–1839, Oct 2017.

[12] P. Lertkultanon and Q. Pham, "A certified-complete bimanual manipulation planner," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 1355–1368, July 2018.

[13] N. Sommer, M. Li, and A. Billard, "Bimanual compliant tactile exploration for grasping unknown objects," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6400–6407.

[14] N. Chavan-Dafle and A. Rodriguez, "Sampling-based planning of in-hand manipulation with external pushes," *arXiv preprint arXiv:1707.00318*, 2017.

[15] N. C. Dafle, R. Holladay, and A. Rodriguez, "In-hand manipulation via motion cones," in *Proceedings of Robotics: Science and Systems*, June 2018.

[16] S. Cruciani, C. Smith, D. Kragic, and K. Hang, "Dexterous manipulation graphs," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 2040–2047.

[17] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3d model acquisition," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, 2002, pp. 438–446.

[18] T. Weise, T. Wismer, B. Leibe, and L. V. Gool, "Online loop closure for real-time interactive 3d scanning," *Computer Vision and Image Understanding*, vol. 115, no. 5, pp. 635 – 648, 2011, special issue on 3D Imaging and Modelling.

[19] D. Tzionas and J. Gall, "3d object reconstruction from hand-object interactions," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 729–737.

[20] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3d object modeling," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1311–1327, 2011.

[21] V. . Nguyen, "Constructing force-closure grasps," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, April 1986, pp. 1368–1373.

[22] M. T. Mason, "Mechanics and planning of manipulator pushing operations," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 53–71, 1986.

[23] "Realtime urdf filter," https://github.com/blodow/realtime_urdf_filter, accessed: 2019-06-26.