

Dexterous Manipulation Graphs

Silvia Cruciani¹, Christian Smith¹, Danica Kragic¹ and Kaiyu Hang²

Abstract—We propose the Dexterous Manipulation Graph as a tool to address in-hand manipulation and reposition an object inside a robot’s end-effector. This graph is used to plan a sequence of manipulation primitives so to bring the object to the desired end pose. This sequence of primitives is translated into motions of the robot to move the object held by the end-effector. We use a dual arm robot with parallel grippers to test our method on a real system and show successful planning and execution of in-hand manipulation.

I. INTRODUCTION

Repositioning an object inside a robot’s hand, known as in-hand manipulation, is an open problem in robotics. Human in-hand manipulation is a rich combination of various grasping movements, including regrasping, sliding and rotating the object. These motions are enabled by the complex human hands, and in particular by their dexterity.

The dexterity of a hand is the ability to perform motions that manipulate a grasped object. In robotics, two terms are commonly used:

- 1) *Intrinsic dexterity* denotes the ability of the robot’s hand to manipulate the object using its several degrees of freedom (DoF). End-effectors with high intrinsic dexterity often mimic the structure of the human hand [1], [2]. Alternatively, the hand can be simpler and the end-effector is designed specifically for the particular task it has to solve [3], [4].
- 2) *Extrinsic dexterity* is the ability to compensate for the lack of DoF using external supports, such as friction, gravity and contact surfaces [5]. It enables dexterous manipulation also with simple parallel grippers.

In this paper, we use a dual arm robot to control the pose of an object held by it. Dual arm robots are becoming widely diffused (e.g. Baxter, Yumi), but the most common end-effector is still a simple parallel gripper. We enhance the lack of dexterity of the gripper by using the two arms. Therefore, our system combines resources extrinsic to the hand but intrinsic to the robot: while it uses a poorly dexterous end-effector, it compensates with the several DoF available in a dual arm system.

The main focus of dexterous manipulation is to move an object held by the gripper to achieve different grasp poses.

¹Silvia Cruciani, Christian Smith and Danica Kragic are with the Robotics, Perception and Learning Lab, EECS at KTH Royal Institute of Technology, Stockholm, Sweden. {cruciani, ccs, dani}@kth.se

²Kaiyu Hang is with the Department of Computer Science and Engineering and the Institute of Advanced Study at Hong Kong University of Science and Technology, Hong Kong. kaiyu.hang@yale.edu

*This work was supported by the Swedish Foundation for Strategic Research project FACT, by the HKUST initiation grant IGN16EG09 and by the HKUST-KTH collaboration grant SSTSP-FP802.

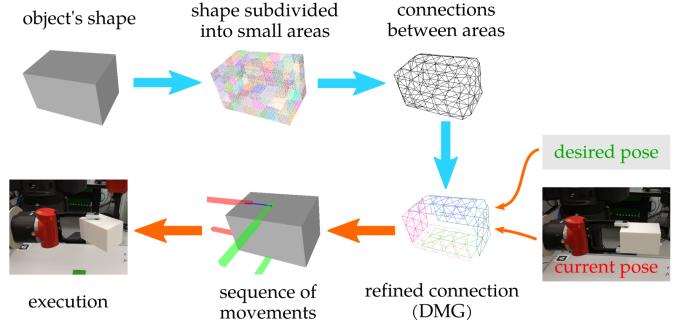


Fig. 1: The blue arrows show the DMG’s derivation: the object’s shape is subdivided into small areas, and their connectivity is refined to describe the admissible motions of one fingertip on the object. The orange arrows show the progression of the repositioning: the DMG provides a sequence of movements from the initial fingers’ contacts (in red) to the desired ones (in green); then, this sequence is executed by the robot.

Past works mostly focused on the end-effector design and control. Therefore, there is a lack of quick and intuitive solutions for planning the in-hand motion of the object.

We propose the Dexterous Manipulation Graph (DMG) as a solution for planning dexterous manipulation. The DMG is a graph representation of the possible motions of one finger in contact with the object’s surface. This graph is used to plan sequences of movements, which we call manipulation primitives, to change the contact points on the object so that it is manipulated towards the desired pose. This sequence of movements is then translated into motions of the dual arm robot. Fig. 1 summarizes the procedure for generating the DMG, and shows a simple example of its use.

In contrast with previous works, we do not require an accurate contact modeling. Our solution provides a way to move rigid objects inside the gripper, even with complex object shapes and for large scale manipulation.

II. RELATED WORK

Dual arm systems can be used to manipulate objects using two grippers at the same time [6]. In particular, we use the Extended Cooperative Task Space (ECTS) [7] to formulate the execution of the motions necessary to reposition an object to the target pose. The ECTS task specification [8] describes different manipulation scenarios, but it does not address the problem of moving an object inside the robot’s gripper.

The method for dual arm manipulation proposed in [9] exploits a state transition graph of object motions. This graph is used to achieve whole-body contact using a humanoid robot, so it exploits both contacts with the hands and with the forearms and possibly other links. In this case, the transition graph is generated by testing the outcome of basic operations on the object, e.g. lifting or sliding. Therefore, this

method requires a physical model of both the object and the environment. Moreover, it does not deal with complex object shapes because the purpose is changing the configuration of big objects for stable transportation.

The “IK-switch” move designed in [10] allows a system of two manipulators to reorient a grasped object. However, it is a regrasping move, therefore the change in configuration is obtained by releasing and grasping the object multiple times and not by moving the contact points on it, which limits the manipulation possibilities.

Despite the use of a dual arm system, our main objective is dexterous manipulation with the end-effector. In-hand manipulation is a widely studied problem. However, most of the available works focus on the control aspect of this problem and often rely on high fidelity dynamic models, and on highly precise and rich sensory input from motion capture systems and force and tactile sensors.

The Hierarchical Fingertip Space grasping method [11] synthesizes a grasp by segmenting the object into small areas, analyzed according to the curvature of the surface, that can be used for contacts with the fingertip. Using tactile feedback, the grasp configuration on the object can be changed, to increase stability, by moving one finger to a new contact point. However, this change of the contact point does not lead to a new object configuration: it is only a more stable version of the previous grasp.

In [12] the authors address the problem of planning joint motions of a multi-fingered hand to change the pose of the object inside it. Their method, called relaxed-rigidity constraint, has the advantage of relying only on the kinematics, without the need for the knowledge of dynamic properties of the object. However, it achieves only small changes in the pose of the objects. Moreover, the repositioning is not obtained by changing the grasp on the object because the contact points are kept stable. Therefore, the correction of the object pose with respect to the end-effector relies on the motion of the finger joints, which are required to have more than 1 DoF.

Similarly, in [13] the authors achieve a stable grasp by moving the two fingers with 3 DoF each to reach a new configuration of the object.

A work that instead focuses on simple parallel grippers is the one described in [14], which also addresses the problem of planning a sequence of motions for repositioning the object. The extrinsic dexterity is provided by multiple external fixtures that have been designed for proper object sliding and rotation. The sequence of pushes against these fixtures necessary to change the grasp is produced by a sampling-based planner. This planner uses a combination of RRT* and an inverse dynamic solver that, given the object’s geometry, its dynamics properties, and the gripping force, models the contact between the object and the gripper or the external pusher. For a successful action, this work relies on an accurate dynamic model, including contact models, and predefined additional supports in the environment.

Strategies that allow for a simpler change in the grasp with a parallel gripper have also been explored. Pivoting [15], [16]

is the rotation of the object around a single axis. Sliding [17] is the planar translation of the object. By combining these two strategies, it is possible to obtain a wider range of changes in the object’s configuration.

In our approach, the DMG plans a sequence of simple motions, named manipulation primitives, whose combination moves the object inside the gripper to the desired pose.

III. DEXTEROUS MANIPULATION GRAPH

The Dexterous Manipulation Graph $\text{DMG} = \langle N, E_{\text{DMG}} \rangle$ is a disconnected undirected graph. A node, or vertex, $n_{ij} = \langle \mathbf{p}_i, A_{ij} \rangle \in N$ is a tuple, containing information about the position of the contact point \mathbf{p}_i and about possible finger’s orientations in that contact point A_{ij} . An edge $e_{n_{ij} n_{hk}} \in E_{\text{DMG}}$ connects two nodes n_{ij} and n_{hk} if it is possible for the fingers to move from one to the other using the manipulation primitives.

Section III-A defines these manipulation primitives. Then, we explain how to generate the DMG in section III-B and analyze the complexity of the proposed algorithm in section III-C. The graph search for obtaining a sequence that moves the object to the goal pose is described in section III-D and the obtained path of the contact points on the object is summarized in section III-E. Finally, section III-F presents the DMG as a tool to analyze the manipulability of an object.

A. Manipulation Primitives

In our method, the object is controlled to move inside the gripper using a combination of simple motions. These motions are defined with respect to the contact point between the fingertip and the object. Their execution requires an adjustment of the pressure exerted by the finger on the object to allow it to move, and external forces that drive the motion.

For the purpose of dexterous manipulation by changing the contact points between the fingers and the object, we have identified the following manipulation primitives:

- 1) Translation (Fig. 2a): in this motion, the contact point slides between two points \mathbf{p}_i and \mathbf{p}_h on the surface of the object and the orientation of the object’s reference frame with respect to the gripper’s reference frame remains constant.
- 2) Rotation (Fig. 2b): in this motion, the contact point is fixed, but the object’s reference frame changes by a planar rotation r around the contact point.
- 3) Roto-translation (Fig. 2c): in this motion, the contact point slides between two points on the object’s surface, and the orientation of its reference frame changes at the same time, not limited to planar rotation changes.

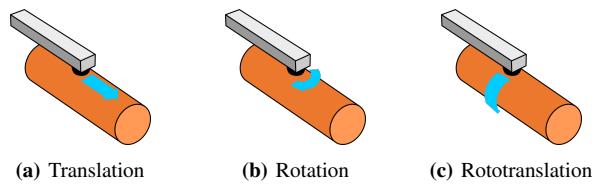


Fig. 2: The three manipulation primitives that describe the motion of the contact point between the fingertip and the object. The finger of a parallel gripper is represented in gray, and its fingertip in black.

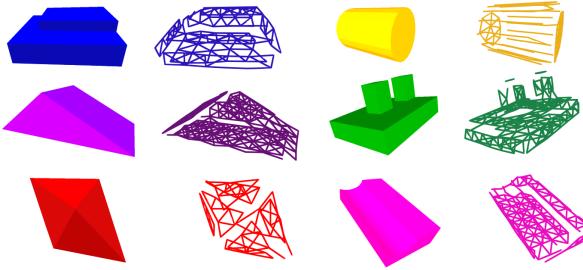


Fig. 3: Examples of Dexterous Manipulation Graphs for different objects.

The first two motions can be obtained with a non-prehensile pushing on the object in certain directions. A subset of motions falling in the third category can be described as a sequence of the first two. The remaining roto-translations require additional torque that is more difficult to provide. For instance, the second arm of the robot can grasp the other end of the object and move it while the first gripper adjusts the gripping force to allow the contact point to slide.

In this work, we focus on motions that do not require additional grasp planning and multiple grasps on the object. Therefore, we designed the DMG to provide a solution that is composed only of the first two primitives. However, we aim to provide a solution for problems that involve multiple grasps and regrasps in our future developments of this work.

In our implementation, the manipulation primitive is executed by providing an additional contact point on the object using the second arm of the robot and pushing with one or both arms. The description of the dual arm motions for this execution is presented in section IV.

B. Graph Generation

The DMG is generated by analyzing the shape of the object, represented as a point cloud. This point cloud is segmented into small areas, and their adjacency is analyzed and refined. The example graphs in Fig. 3 represent only a partial visualization of the information contained in the DMG. However, this visualization is intuitive and shows a rough example of what the graph provides.

The first step towards obtaining the DMG is dividing the object into small areas. To segment the object into connected areas, we use the method of supervoxels for point clouds [18]. After selecting the desired resolution, chosen according to the object’s dimensions and the desired precision of the graph, this segmentation subdivides the object into areas that are grouped according to common characteristics, such as the geometric information. To each area, called supervoxel, correspond a centroid, which is a point on the surface of the object, and a normal to this surface. The centroid of the i -th supervoxel, \mathbf{p}_i , is considered a possible contact point between the fingertip and the object.

Each supervoxel is adjacent to its neighboring areas, and we keep this adjacency information in an initial connected graph $G = \langle V, E \rangle$. Its vertexes $v \in V$ correspond to points \mathbf{p}_i on the object. In contrast, the nodes of the final DMG contain additional information other than the centroid of the corresponding supervoxel. The process for generating the

Algorithm 1: graph_generation

```

Input :  $G, l, a_s$ 
Output: DMG,  $C_{\text{DMG}}$ 
1  $V \leftarrow$  vertices of  $G$  //  $v \in V$  is a point on the object
2  $E \leftarrow$  edges of  $G$  //  $e_{ij}$  edge between  $i$  and  $j$ 
3 foreach  $v \in V$  do
4    $\hat{\mathbf{n}}_v \leftarrow$  normal in  $v$ 
5   foreach  $w$  adjacent to  $v$  do
6      $\hat{\mathbf{n}}_w \leftarrow$  normal in  $w$ 
7     if  $\|\hat{\mathbf{n}}_v - \hat{\mathbf{n}}_w\| > \delta$  then
8        $E \leftarrow E \setminus \{e_{vw}\}$ 
9   if  $v$  is isolated then
10     $V \leftarrow V \setminus \{v\}$ 
11  $C_G \leftarrow$  connected subgraphs in  $G$ 
12  $N \leftarrow \emptyset$  // empty set of nodes
13 foreach  $v \in V$  do
14    $A \leftarrow \emptyset$  // empty set of admissible angles
15   foreach  $a \in \{0 : a_s : 360\}$  do
16     if  $a$  is admissible with finger length  $l$  then
17        $A \leftarrow A \cup \{a\}$ 
18   if  $A$  is empty then
19     remove  $v$  from  $V$  and from its component
       $C_v \in C_G$ 
20   else
21      $A_v \leftarrow \emptyset$  // empty set of angle components
22      $j \leftarrow 0$ 
23     foreach sequential set in  $A$  do
24        $A_{vj} \leftarrow j^{\text{th}}$  sequential set in  $A$ 
25        $n_{vj} \leftarrow \langle v, A_{vj} \rangle$ 
26        $N \leftarrow N \cup \{n_{vj}\}$ 
27        $A_v \leftarrow A_v \cup \{A_{vj}\}$ 
28        $j++$ 
29  $E_{\text{DMG}} \leftarrow$  empty set of edges
30 foreach  $C_i \in C$  do
31    $V_i \leftarrow$  points in  $C_i$ 
32    $E_i \leftarrow$  connections in  $C_i$ 
33   foreach  $v \in V_i$  do
34     foreach  $A_{vj} \in A_v$  do
35       foreach  $w$  adjacent to  $v$  in  $C_i$  do
36         foreach  $A_{wk} \in A_w$  do
37           if  $A_{vj} \cap A_{wk} \neq \emptyset$  then
38              $E_{\text{DMG}} \leftarrow E_{\text{DMG}} \cup \{e_{n_{vj} n_{wk}}\}$ 
39  $\text{DMG} \leftarrow \langle N, E_{\text{DMG}} \rangle$ 
40  $C_{\text{DMG}} \leftarrow$  connected subgraphs in DMG
41 return DMG,  $C_{\text{DMG}}$ 

```

DMG from G is summarized in Algorithm 1 and explained below.

Once the object has been subdivided, a first refinement of the connections is performed. Given the normals $\hat{\mathbf{n}}_i$ and $\hat{\mathbf{n}}_h$ of the i -th and h -th adjacent supervoxels, the connection between \mathbf{p}_i and \mathbf{p}_h is removed if $\|\hat{\mathbf{n}}_i - \hat{\mathbf{n}}_h\|_2 > \delta$. The threshold δ ensures that the fingertip does not move across sharp edges, and it can be adjusted to allow for motions on uneven or curved surfaces to the desired degree.

After refining the connectivity, the graph is divided into separate connected components C_G , on which the fingertip can slide and rotate while maintaining contact. However, we

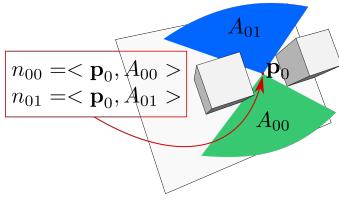


Fig. 4: The point \mathbf{p}_0 on the surface of this plug-shaped object corresponds to two different and disconnected nodes in the DMG, n_{00} and n_{01} , because a gripper's finger can grasp it with two separate angle ranges, A_{00} and A_{01} .

need to take into account the finger's body, and not just the fingertip, when executing different movements on the objects. Therefore, we further refine the adjacency assuming a finger of length l of a parallel gripper. In fact, collisions can occur due to non convex shapes and due to objects being longer than the finger.

We discretize the possible orientations that a finger can assume in the set $[0^\circ, 360^\circ]$ with a step a_s . A finger can be in contact with the object at a point \mathbf{p}_i and it can assume only orientations that are not in collision. These orientations belong to the set of possible angles A_i . If this set is empty, the point is removed from the graph because it is impossible for the gripper to grasp there.

The set of possible angles A_i is then further analyzed to finally obtain the DMG and the connection between its nodes. The angular component $A_{ij} \subseteq A_i$ of a node n_{ij} is a set of admissible orientations that are in a continuous sequence among the ones in A_i . More specifically, the finger is able to rotate from the minimum angle in A_{ij} to the maximum angle using a single continuous rotation, without encountering collisions. A connection between two nodes n_{ij} and n_{hk} , corresponding to the contacts \mathbf{p}_i and \mathbf{p}_h , is inserted in the DMG only if $A_{ij} \cap A_{hk} \neq \emptyset$.

If an area has multiple angular components, the corresponding nodes in the DMG are disconnected because the finger cannot rotate at that contact point between the two disjoint sets of orientations. In fact, a contact occurring in the position \mathbf{p}_i on the object's surface can correspond to more than one node in the graph, because there is a disconnection in the angular components, as in the example in Fig. 4.

When refining the adjacency by taking into account the orientation, the amount of connected components in the graph can increase with respect to the ones in C_G , depending on the object's geometry. Therefore, we define the set of connected components in the DMG as C_{DMG} .

C. Complexity Analysis

The complexity of the generation of the DMG can be easily derived. Although the process in Algorithm 1 can be optimized, we prefer this notation for readability purposes, and we assume a Breadth-first search for the operation of deriving the connected components.

Given $|V|$ as the number of vertexes in G , which depends on the chosen resolution in the segmentation of the object, a as the number of considered angles, which is $\lceil \frac{360}{a_s} \rceil$, b as the maximum number of edges from a node, and c as the maximum number of angular components, the complexity is

$$O((\max(c^2 b, ac))|V|), \quad (1)$$

where b is usually at most 4 or 5 because of the geometric adjacency of the supervoxels, c is 1 for most of the nodes and it can be expected to be a small number (2 or 3) for a few of them, only for unusual and complex shapes, and a depends on the chosen resolution for the angles (a_s).

D. Graph Search

The DMG is composed of separate connected components, in which the contact point can move. If the desired contact on the object lies in the component $C_D \in C_{\text{DMG}}$, and this component is different from the component $C_I \in C_{\text{DMG}}$ of the initial contact, it is not possible to achieve the desired repositioning without releasing the contact between the gripper and the object.

When searching for a path within a connected component, optimal and fast strategies for graph search can be used. In our particular implementation, we use Dijkstra's algorithm. Fig. 5 shows an example of two paths. These paths are found between the same contact points, but with two different target orientations of the gripper.

We address the problem of in-hand manipulation with a parallel gripper, which has two contacts on the object. However, the DMG is built taking into account only one contact point. The contact points of the two fingers on the object are associated to the node with the closest position and with angular components that contains the current orientation. We assume that one finger is the *principal finger*, arbitrarily chosen, and the other one, the *secondary finger*, follows it on the opposite side in its own corresponding connected component of the DMG. When looking for a feasible solution, both fingers must remain in their component during the whole motion, named C_p for the principal finger and C_s for the secondary finger. While the principal finger moves along the nodes in the graph, the secondary one has a more relaxed association with them to ensure that the parallel gripper moves properly on the object.

The graph search algorithm is executed from the initial principal finger contact n_{init}^p to the desired one n_{goal}^p . At each iteration, to a node n_{ij}^p in C_p corresponds a secondary finger's contact point. This point is found as the intersection between the object and the line starting from the position of the principal contact \mathbf{p}_i with the same direction of the line that connects the two fingers in the previous configuration. In fact, a motion between nodes with different positions of

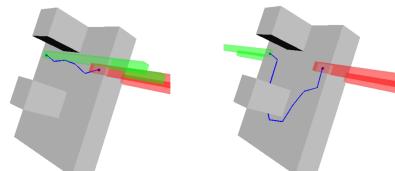


Fig. 5: A path for the contact point to move the gripper from the initial configuration (in red), to the goal configuration (in green). The contact points are the same, but the two goals have a different gripper's orientation. The different angular components provide two different paths.



Fig. 6: An example of the same in-hand repositioning computed assigning a different gripper’s finger as principal. The found paths are both valid, but they lead to different motions of the object.

the contact point corresponds to the translation movement primitive, and therefore the orientation of the fingers with respect to the object remains constant. The obtained point is associated to a node n_{xy}^s in the DMG. If n_{xy}^s is not in C_s , or the gripper is in an invalid configuration, the distance between the two considered principal nodes n_{ij}^p and n_{hk}^p is set to ∞ to ensure that this path is not chosen.

The obtained path may not be the shortest path for the principal finger, because it depends also on the secondary finger on the opposite side of the object. As shown in Fig. 6, in the second image the path avoids the center of the object, although for that contact point alone moving there would be the optimal solution.

As final comment, we briefly discuss how to use the distance between two nodes n_{ij}^p and n_{hk}^p during the search. This distance can be the simple euclidean distance $\|\mathbf{p}_h - \mathbf{p}_i\|_2$, or it can be chosen according to how expensive a movement is with the available robot. For instance, if a rotation is required because the current angle γ of the finger is so that $\gamma \in A_{ij}$ but $\gamma \notin A_{hk}$, this rotational movement can be considered as more expensive by increasing the distance.

In addition, the parallel gripper can increase or decrease the opening between the two fingers when it is manipulating an object whose surfaces are not parallel. This opening is obtained from the distance between the contacts of the principal and secondary fingers. Depending on the precision and efficiency in actuation of the available gripper, changes in distance between the fingers can be penalized during the graph search, according to how preferable it is to keep a stable fingers’ opening versus a possibly longer path.

Finally, some motion directions can be also penalized by increasing the distance between the corresponding nodes, to avoid undesirable behaviors in the execution, such as the examples mentioned later in sections IV-B and IV-C.

E. In-hand Manipulation Sequence

The path obtained from the DMG is a sequence of K nodes $\{n^0, n^1, \dots, n^{K-1}\}$, with $n^0 = n_{init}^p$ and $n^{K-1} = n_{goal}^p$. This path must be translated into a sequence of manipulation primitives.

Given the nodes $n^k = \langle \mathbf{p}^k, A^k \rangle$ and $n^{k+1} = \langle \mathbf{p}^{k+1}, A^{k+1} \rangle$ in sequence in the path and the current orientation of the finger γ^k in the contact point of the node n^k , two manipulation primitives are needed to move the contact from n^k to n^{k+1} :

- 1) A rotation r^k so that the next orientation $\gamma^{k+1} = \gamma^k + r^k$ is so that $\gamma^{k+1} \in A^k \cap A^{k+1}$. This rotation can be chosen to minimize the change, so that it is 0 when $\gamma^k \in A^k \cap A^{k+1}$, to get closer to the goal angle γ^{K-1} , or to fulfill additional requirements.

- 2) A translation \mathbf{t}^k that slides the contact point on the object from \mathbf{p}^k to \mathbf{p}^{k+1} . This translation is obtained as $\mathbf{t}^k = \mathbf{p}^{k+1} - \mathbf{p}^k$. Therefore, from a path of K nodes, the corresponding translations are $K - 1$.

The sequence of manipulation primitives is an alternation of rotations and translations, plus a final rotation to reach the final angle: $\{r^0, \mathbf{t}^0, r^1, \mathbf{t}^1, \dots, r^{K-2}, \mathbf{t}^{K-2}, r^{K-1}\}$. It is possible to have several rotations with value 0, which can then be ignored during the execution. In contrast, the translations are always nonzero because there is no connection between nodes n_{ij} and n_{ik} , with the same contact point but different angular component. Since the path is fragmented into many segments, we unite these segments to avoid a slow execution. For instance, two translations are joint if there is a rotation equal to 0 in between and the change in direction of the path on the object’s surface is negligible.

F. From DMG To Manipulability Analysis

In addition to planning dexterous manipulation, the DMG can be used to analyze the manipulability of an object. In particular, we focus on how much the manipulation primitives can reposition an object inside the gripper without the need for regrasping.

We build a *manipulability matrix* by computing a path between several different poses of the gripper, distributed in the object’s bounding box. This matrix contains 1 in the ij position if a path exists between pose i and pose j , and 0 otherwise. The resulting matrix is symmetric and it is an adjacency matrix. By setting the diagonal elements to 0, loops between nodes are removed. This matrix can be seen as an extended representation of the DMG, but it strictly depends on the chosen parallel gripper, while in the DMG each node is associated to a single contact point and this abstraction is an advantage in case our method is extended to multi-fingered hands.

This matrix can be ordered into blocks, which are a simple tool to visualize the manipulability of the objects. The manipulability matrix provides an overview on how easy it is for the robot to manipulate a certain object without the need to regrasp. In fact, if there are fewer blocks, the likelihood of manipulating an object while maintaining contact is higher. This matrix can also be used in future implementations as an aid to a regrasping strategy that takes into account both the object’s shape and the current gripper’s configuration, because it provides information on an area where to regrasp. Examples of manipulability matrices are analyzed in section V-B.

IV. DUAL ARM FORMULATION

As mentioned, we use the ECTS [7] to describe the motion of our dual arm system. The ECTS allows for a specification of the absolute and relative velocities of the two end-effectors and it translates them into velocities for each robot’s arm. Additionally, the degree of coordination of each arm can be adjusted to obtain symmetrical or asymmetrical executions.

With $\dot{\mathbf{x}} = (\dot{\mathbf{x}}_1, \dot{\mathbf{x}}_2)^T$ being the vector containing the Cartesian velocities of the first and second end-effectors, and

$\dot{\mathbf{x}}_a = (\mathbf{v}_a, \boldsymbol{\omega}_a)^T$ and $\dot{\mathbf{x}}_r = (\mathbf{v}_r, \boldsymbol{\omega}_r)^T$ being their absolute and relative motions expressed as linear and angular velocity, the relationship between these velocities is

$$\dot{\mathbf{x}} = \begin{bmatrix} \alpha \mathbf{I}_6 & -(1-\alpha) \mathbf{I}_6 \\ -\beta \mathbf{I}_6 & \alpha \mathbf{I}_6 \end{bmatrix} \dot{\mathbf{x}}_E, \quad (2)$$

where $\dot{\mathbf{x}}_E = (\dot{\mathbf{x}}_a, \dot{\mathbf{x}}_r)^T$, \mathbf{I}_6 is the 6-dimensional identity matrix, and α and β are coordination coefficients. In particular, $\beta \in \{0, 1\}$ and $\alpha \in [0, 1]$. With β set to 0, the motion is uncoordinated, therefore in our case it is always set to 1, as the two arms are manipulating the same object with the purpose of repositioning it inside one of the two grippers. α defines the degree of coordination. For instance, if it is set to 0.5, the motion of the arms is symmetrical. If it is set to 0, only one arm moves and this mode can be seen equivalent to a system of a single arm robot and an external support.

The motion of the object inside the gripper can be described using the relative motion of the two arms $\dot{\mathbf{x}}_r$. In our experiments, we keep the absolute motion $\dot{\mathbf{x}}_a$ to zero, but it can be exploited to move the arms while the robot is manipulating the object.

In this section, we first provide an overview of the execution procedure and then we explain in detail how to obtain the translation and rotation manipulation primitives with the dual arm system.

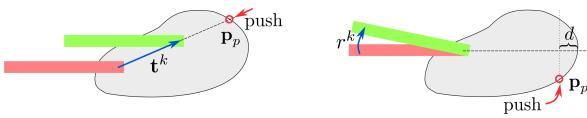
A. Execution Procedure

In our dual arm system, the robot grasps the object with its first gripper, and its second gripper helps pushing the object to adjust its pose. The object must be moved according to the sequence of manipulation primitives derived in section III-E. To each translation \mathbf{t}^k and to each rotation r^k correspond the following steps executed by the robot:

1) *Find contact point*: In this step, the robot does not move, but it analyzes the shape of the object. In fact, for the second gripper to be able to push the object in the desired pose, the pushing must begin from a proper contact point with the object. Fig. 7 shows examples on how to find the contact point between the object and the second gripper.

2) *Approach contact point*: In this step, the robot moves its arms so that the second gripper is in contact with the object at the chosen point.

3) *Execute manipulation primitive*: In this step, the robot moves the arms so that the object's pose inside the first gripper changes according to the corresponding manipulation primitive. The first gripper also enlarges and tightens the



(a) Given \mathbf{t}^k , the object is pushed from a point \mathbf{p}_p found as the intersection between its surface and the line passing through the contact point \mathbf{t}^k with the direction of the translation. (b) Given r^k , \mathbf{p}_p is one of the two intersections between a line orthogonal to the finger, put inside the object of a distance d , chosen according to the direction of r^k .

Fig. 7: The derivation of the contact point between the second gripper and the object. The initial finger's pose is red, and the desired one is green.

fingers to facilitate the object's motion. The explanation of the translation execution is given in section IV-B, and the rotation execution is described in section IV-C.

4) *Leave contact point*: In this step, the robot moves the arms so that the second gripper is no longer in contact with the object. The first gripper is instead still grasping the object, in the new pose that derived from the execution of the manipulation primitive.

B. Translation Execution

If the manipulation primitive to be executed is translation, then

$$\dot{\mathbf{x}}_r = (\mathbf{v}, \mathbf{0})^T. \quad (3)$$

$\mathbf{0}$ is a 3-dimensional zero vector, indicating that there is no angular velocity because the orientation of the object inside the gripper must remain constant. The linear velocity \mathbf{v} depends on the desired translational motion of the object.

Given the translation \mathbf{t}^k of the contact point on the surface of the object, the direction of the translation of the contact point in the robot's base frame is $\mathbf{t}'^k = {}^b\mathbf{R}_{g_1} \tilde{\mathbf{t}}^k$, where ${}^b\mathbf{R}_{g_1}$ is the rotation matrix containing the orientation of the gripper in the base reference frame and $\tilde{\mathbf{t}}^k$ is a 3-dimensional vector containing the 2-dimensional \mathbf{t}^k expressed in the gripper's reference frame. To achieve a translation of the contact point directed as \mathbf{t}'^k , the object must move in the opposite direction, denoted as $-\tilde{\mathbf{t}}'^k$. We adjust the magnitude of the velocity m using a P controller during the motion execution, until the fingertip reaches the desired contact point. In conclusion, the linear velocity is $\mathbf{v} = -m\tilde{\mathbf{t}}'^k$.

We assume that the friction between the object and the second gripper is so that the latter can push the object also along diagonal directions. However, since the DMG is independent from the initial grasping configuration, it contains translational motions along directions that may not be feasible for pushing, and which instead require pulling. A pulling motion would require the second gripper to grasp the object, and since we do not deal with regrasping problems in this work, we prefer to discard translational motions that require a pulling motion at the planning stage. More specifically, these motions are penalized in the cost of the path during the graph search, so that a preferred solution contains a rotation of the object, a push and a second rotation rather than a pull.

C. Rotation Execution

If the motion to be executed is a rotation, the relative motion of the second gripper's contact point is along an arc of a circle. We induce also an angular velocity so that the second gripper follows the object during the rotation without altering the contact surface.

Assuming the reference frame of the gripper as in Fig.8, the velocity in the first gripper frame is

$${}^{g_1}\dot{\mathbf{x}}_r = (\cos(\theta + \phi_t)\dot{\phi}_t, 0, -\sin(\theta + \phi_t)\dot{\phi}_t, 0, \dot{\phi}_t, 0)^T. \quad (4)$$

θ is the initial angle of the second gripper's contact point \mathbf{q} on the object with respect to the first gripper's orientation,

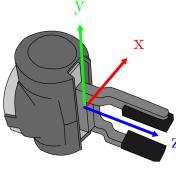


Fig. 8: The gripper's reference frame.

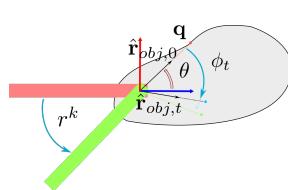


Fig. 9: A rotation r^k , from a finger's initial orientation (in red) to the desired one (in green).

and ϕ_t is its variation. Given the desired rotation of the finger in contact r^k , ϕ_t produces a rotation in the opposite direction. As represented in Fig. 9, the gripper's finger has to rotate of r^k , and therefore the point q on the object rotates around the grasp contact point in the opposite direction. ϕ_t evolves during time so that at the beginning it is equal to 0 and at the end it is equal to $-r^k$. The point q is initially at an angle θ with respect to the gripper. $\hat{\mathbf{r}}_{obj,t}$ is the unitary vector pointing in the direction of q during the motion. Its evolution in the zx plane is described as $\hat{\mathbf{r}}_{obj,t} = (\cos(\theta + \phi_t), \sin(\theta + \phi_t))^T$, from which the velocity in (4) is derived. This velocity is then transformed into the relative velocity $\dot{\mathbf{x}}_r$ expressed in the global reference frame.

Similarly to the translation velocity, we adjust the magnitude of this rotation velocity during the execution, until the object reaches the desired orientation. Excessive rotations that could lead to the second gripper colliding with the first arm should be penalized as the pull motions for the translations.

V. EXPERIMENTS

We tested our dexterous manipulation strategy on a Baxter robot. In addition, we used the DMG to evaluate the manipulability of objects, and to analyze possible aids to a regrasping strategy that compensates for desired configurations that are not reachable by moving the contact point along the object's surface.

A. Robot Experiments

We used a Baxter robot to execute the dual arm strategy defined in section IV. We used hemispherical fingertips for better controlling the friction at the contact point when enlarging or tightening the grasp. We detected the pose of

the object using April tags and the camera built on Baxter's end-effector, with a frequency of 28 fps. This visual feedback was used to adjust the object's motion during the execution.

As mentioned, the strategy consists of a sequence of pushes on the object and an adaptation of the distance between the fingers to allow the object to move. We have kept the absolute motion $\dot{\mathbf{x}}_a$ to zero, leaving only the relative motion $\dot{\mathbf{x}}_r$ to be executed by the two arms.

Due to the compliant nature of Baxter, the executions of the pushing was subject to position errors. However, the accuracy was good enough to achieve the repositioning within a range of about 10mm and 5° around the desired configuration of the gripper's finger on the object.

We have also verified that the error was smaller when the gripper that was grasping the object was kept fixed and only the second gripper was moving ($\alpha=1$). In the opposite case ($\alpha=0$), in which the first gripper moves against the second one, or in blended mode ($0 < \alpha < 1$), we noticed that the second gripper was significantly pushed away due to the compliance in the arm. Therefore, precise repositionings were more difficult to achieve in this dual arm mode. However, we highlight that the mode $\alpha=0$ is a mode that can be used to substitute the dual arm system with a single robot arm that pushes against the environment, non-compliant, instead of against its second gripper.

Fig. 10 shows an example of the execution of an in-hand repositioning. The desired final configuration is shown on the right side of the image, as well as the executed path of the contact point on the object's surface. The DMG was built using a resolution of 13mm of the supervoxels, which produced 117 segmented areas, and refined with $\delta=0.07$, $a_s=5^\circ$ and $l=100\text{mm}$. The P values were 0.7 for the fingers' distance, 0.32 for the linear velocity and 16 for the angular velocity. The execution of this experiment and other experiments with different objects are available in the supplementary video.

B. Object's Manipulability

We have used the DMG to analyze the manipulability of an object by building its manipulability matrix. After removing the rows and column corresponding to invalid poses, and hence containing non useful data, the matrices were ordered

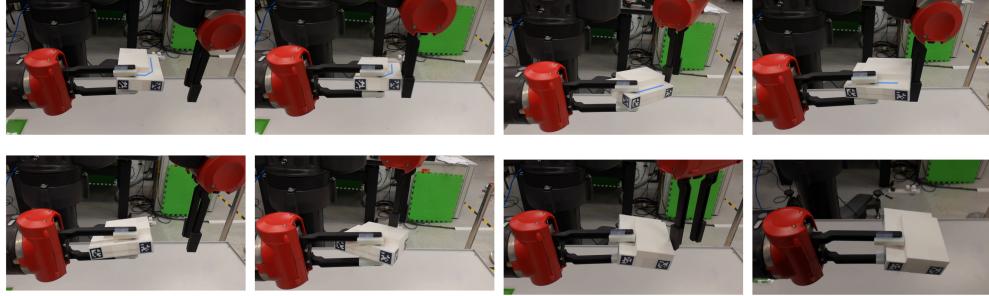


Fig. 10: The in-hand manipulation executed with Baxter. The desired motion of the contact point is shown on the right and is superimposed to the pictures as a blue line. The execution is composed of an alternation of four rotations (of which the first two are 0) and three translations: the second gripper approaches the object and pushes it for translating the contact points with the first gripper, then it changes the approach point and the push direction to impose a rotation on the object, and it repeats these motions until the object reaches the desired pose.



Fig. 11: The manipulability matrices associated with the object’s surface. The first matrix is 1332×1332 and the second one is 1284×1284 . The matrices are binary, and black represents 0. The colors are superimposed to the blocks of 1 to show the different areas of gripper’s contact.

into blocks. Fig. 11 shows an example of the manipulability matrix with two different objects. The correspondence of the blocks is not with a connected component of the DMG, but with an area of the object in which the gripper can manipulate it without losing contact. For instance, the bottom part of the second object is associated with both the red and the blue blocks, while it corresponds to a single connected component in the DMG. In fact, while in the DMG only one contact point is considered, the matrix contains information related to both of the gripper’s fingers.

The manipulability matrices also provide useful information for future implementations of a regrasping strategy, because they connect the desired pose to two connected components, but they also provide an area inside the connected component that is preferred, in case a connected component is associated to multiple blocks, hence restricting the search area for a grasp.

VI. CONCLUSIONS

We have introduced a novel tool for planning in-hand manipulation that we named Dexterous Manipulation Graph. First, we defined manipulation primitives that can be used by the robot to move an object inside the grasp while maintaining contact. Then, We suggested to analyze the shape of the manipulated object to generate a graph that describes the possible motions of the contact point between the fingertip and the object. This graph is used to search for sequences of manipulation primitives to reposition the object from an initial configuration inside the gripper to a desired one. The DMG can also be used to analyze the manipulability of an object using its manipulability matrix. In addition, we formulated the execution of the translation and rotation movement primitives in a dual arm framework. However, the same movement can be executed with a single arm and an external contact without modifying our definitions. Finally, we have tested our approach to in-hand manipulation using a Baxter robot.

As future work, we plan to include the possibility to grasp the object with the second arm of the robot and execute manipulations that include the roto-translation primitive. This inclusion will likely change the shape of the graph for objects without sharp edges. Additionally, we will include manipulations that require a regrasp, and we will use the DMG and the manipulability matrix to plan where the regrasp should be and where the second gripper should grasp so to not hinder the repositioning inside the first gripper. Moreover, we plan to exploit the difference in possible solutions given the different selection of the principal finger to improve

the overall planning of the in-hand manipulation sequence. Finally, we also want to extend the DMG to be used with multi-fingered hands, and exploit the knowledge of contact-based connected components in the graph and the additional intrinsic dexterity to achieve even more dexterous in-hand manipulations.

REFERENCES

- [1] R. Ozawa and K. Tahara, “Grasp and dexterous manipulation of multi-fingered robotic hands: a review from a control view point,” *Advanced Robotics*, vol. 31, no. 19–20, pp. 1030–1050, 2017.
- [2] A. Bicchi, “Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity,” *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 652–662, Dec 2000.
- [3] N. Rahman, L. Carbonari, M. D’Imperio, C. Canali, D. G. Caldwell, and F. Cannella, “A dexterous gripper for in-hand manipulation,” in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, July 2016, pp. 377–382.
- [4] W. G. Bircher, A. M. Dollar, and N. Rojas, “A two-fingered robot gripper with large object reorientation range,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3453–3460.
- [5] 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.
- [6] C. Smith, Y. Karayannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, “Dual arm manipulation—a survey,” *Robotics and Autonomous Systems*, vol. 60, no. 10, pp. 1340 – 1353, 2012.
- [7] H. A. Park and C. S. G. Lee, “Extended cooperative task space for manipulation tasks of humanoid robots,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6088–6093.
- [8] ———, “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.
- [9] M. Murooka, Y. Inagaki, R. Ueda, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba, “Whole-body holding manipulation by humanoid robot based on transition graph of object motion and contact,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 3950–3955.
- [10] 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.
- [11] K. Hang, M. Li, J. A. Stork, Y. Bekiroglu, F. T. Pokorný, 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.
- [12] B. Sundaralingam and T. Hermans, “Relaxed-rigidity constraints: In-grasp manipulation using purely kinematic trajectory optimization,” in *Robotics: Science and Systems*, 2017.
- [13] E. Psomopoulou, D. Karashima, Z. Doulgeri, and K. Tahara, “Stable pinching by controlling finger relative orientation of robotic fingers with rolling soft tips,” *Robotica*, vol. 36, no. 2, pp. 204–224, 2018.
- [14] N. Chavan-Dafle and A. Rodriguez, “Sampling-based planning of in-hand manipulation with external pushes,” *arXiv preprint arXiv:1707.00318*, 2017.
- [15] A. Sintov, O. Tslii, and A. Shapiro, “Robotic swing-up regrasping manipulation based on the impulse-momentum approach and clqr control,” *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1079–1090, Oct 2016.
- [16] 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.
- [17] J. Shi, J. Z. Woodruff, P. B. Umbohanwar, and K. M. Lynch, “Dynamic in-hand sliding manipulation,” *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 778–795, Aug 2017.
- [18] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter, “Voxel cloud connectivity segmentation - supervoxels for point clouds,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, June 2013, pp. 2027–2034.