2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

In-Hand Manipulation of Objects with Unknown Shapes

Silvia Cruciani, Hang Yin and Danica Kragic

Abstract— **This work addresses the problem of changing grasp configurations on objects with an unknown shape through in-hand manipulation. Our approach leverages shape priors, learned as deep generative models, to infer novel object shapes from partial visual sensing. The Dexterous Manipulation Graph method is extended to build incrementally and account for object shape uncertainty when planning a sequence of manipulation actions. We show that our approach successfully solves in-hand manipulation tasks with unknown objects, and demonstrate the validity of these solutions with robot experiments.**

I. Introduction

In-Hand manipulation of unknown objects is a particularly challenging instance of dexterous manipulation. The problems of addressing where to grasp for given tasks and how to adapt the grasp on novel objects have been addressed and studied [1]–[6], but the problem of how to reach such a grasp starting from an initial one, without accessing the object's model, is still an open challenge. Reactive controllers accommodate local shape uncertainties via motion compliance and feedback, often relying on strong assumptions (e.g. invariant contact points [7]) or explorative trials [8], without aiming at a specific goal grasp. We target general in-hand manipulation to achieve a desired grasp on objects with unknown shapes. This calls for more flexible ways to handle unknowns and the capability of planning multi-stage actions as in Dexterous Manipulation Graphs (DMG) [9].

In contexts where the desired grasp is not directly reachable, the robot must adjust the object's configuration inside the hand after an initial grasp [10], [11]. In-hand manipulation solutions rely on accurate knowledge of geometry and dynamics of the specific grasped object [12], [13], or on many data samples [14], [15]. However, target grasp configurations can be obtained for a given task or object type [1], but do not strictly depend on the particular object shape (e.g. grasp by the handle). Instead of requiring a full object model, our method can operate on the object class level. We aim at providing an initial step towards successful integration of in-hand manipulation with task-based grasp planning methods that do not rely on full knowledge of the object's shape, so that robots can successfully manipulate previously unseen objects to execute different tasks.

In this paper, we focus on achieving in-hand manipulation with a parallel gripper. Given no dexterity, grasp reconfiguration can be achieved by leveraging *extrinsic dexterity* [16], i.e. the exploitation of supports external to the gripper, such as pushing against a second gripper or external contacts [9], [17]–[19] and motions due to gravity and inertial forces [20]–[22]. In our case, we consider manipulation motions in which the object is pushed against the second robot's gripper, as in [9]. Moreover, we address in-hand manipulation by only exploiting vision; while tactile sensing improves the performances of controlling the slippage between gripper finger and object surface, visual feedback is often enough for successful in-hand manipulation executions [9], [23], even without relying on accurate dynamic models [24]–[26].

Vision information of objects can be subject to occlusions. Shape uncertainty has been tackled in grasp synthesis [27], control [28] and transferring based on part similarity [29]. Our method copes with uncertainty by explicitly estimating a full object shape based on generative models. Generative models build on a parameterized space to encode plausible object shapes. Leveraging recent progresses in 3D shape synthesis, we use a variational auto-encoder to learn shape priors from point cloud data. Previous works explored Gaussian Process Implicit Surface [30] and non-rigid transformation [5], both targeting static grasp transfer. These approaches assume a canonical shape bias while deep generative models have demonstrated the capacity of encoding more diverse shape families [31]–[33].

We extend the DMG method introduced in [9]. This method proposes to represent the object as a graph structure. The main disadvantages of DMG are that it needs a full model of the object shape and it requires an offline step for precomputing the graph. Once the graph is generated, it is used to plan any in-hand manipulation task on the given object. We propose a method that focuses on achieving a given in-hand manipulation task by exploiting only the information available to the robot when the task is given. Moreover, while in-hand manipulation is planned, we generate the graph so that successful manipulation tasks with the same object can re-use it, having the same advantages of DMG.

This work builds on our work in [9], where we introduce the idea of DMG. Here, we extend the work to consider objects of unknown shapes. We improve the DMG to represent the object reconstruction uncertainty and consider this uncertainty in the provided in-hand manipulation solution. The method presented here estimates DMG online and graph edges are added in relation to push feasibility.

II. Overview

A. Preliminaries

We represent a grasp configuration G as a tuple: $\langle \mathbf{g}_1, \mathbf{g}_2, \mathcal{Q} \rangle$. $\mathbf{g}_1, \mathbf{g}_2 \in \mathbb{R}^3$ are the two gripper's fingertips,

S. Cruciani, H. Yin and D. Kragic are with the Division of Robotics, Perception and Learning, EECS at KTH Royal Institute of Technology, Stockholm, Sweden. {cruciani, hyin, dani}@kth.se

This work was supported by the Swedish Foundation for Strategic Research, the Swedish Research Council and the Knut and Alice Wallenberg Foundation.

in contact with the object, and $\mathcal{Q}$ is the unit quaternion representing the gripper's orientation.

Since the object's model is unknown, these quantities cannot be expressed w.r.t. its reference frame $\Sigma_o \in SE(3)$. We express all the quantities in the initial gripper's frame Σ_g. Σ_o and Σ_g are rigidly attached, so a relative change in the grasp, i.e. a new gripper frame Σ_g' and a new grasp G', can be equivalently expressed in both of them.

The visible part of the object is a point cloud $P_v \subset \mathbb{R}^3$, i.e. a discrete set of 3D points. The reconstructed object, which contains the invisible part, is represented as another point cloud $P_r \subset \mathbb{R}^3$. The full object shape is the point cloud $P_o = P_v \cup P_r$. By $|P|$ we indicate the number of points in a point cloud.

To move the object within the gripper, we use the same in-hand movements as [9]: rotation and translation. A rotation $\phi \in (-2\pi, 2\pi)$ moves the grasp from $G{=}\langle \mathbf{g}_1, \mathbf{g}_2, \mathcal{Q}\rangle$ to $G'{=}\langle \mathbf{g}_1, \mathbf{g}_2, \mathcal{Q}'\rangle$. Due to the parallel gripper's structure, rotations about the fingertips are planar. A translation $\mathbf{t} \in \mathbb{R}^3$ moves from $G{=}\langle \mathbf{g}_1, \mathbf{g}_2, \mathcal{Q}\rangle$ to $G'{=}\langle \mathbf{g}_1', \mathbf{g}_2', \mathcal{Q}\rangle$. These movements can be executed by pushing the object against external fixtures or an additional gripper. Combinations of the two movement types at the same time are not considered during planning, but can occur during execution and will be compensated by a controller, as in [9].

We use $\mathcal{M}$ to denote a sequence of in-hand movements, e.g. $\phi_0, \mathbf{t}_0, \mathbf{t}_1, \phi_1, ...\phi_K$. This sequence is composed of K rotations and H translations.

A Dexterous Manipulation Graph $\mathcal{G}$ is a disconnected graph composed of a set of nodes $N_\mathcal{G}$ and a set of edges $E_\mathcal{G}$. A node $n \in N_\mathcal{G}$ is a tuple $\langle \mathbf{p}, \Theta\rangle$; $\mathbf{p} \in \mathbb{R}^3$ represents a contact point between one fingertip and the object, and $\Theta \subseteq [0, 2\pi)$ is a continuous set of orientations w.r.t. the axis between the two fingertips that the gripper's finger can assume when in contact at $\mathbf{p}$. Therefore, a single contact can correspond to more nodes, depending on the object's shape. Note that a DMG node represents a single finger, while for planning the two opposite gripper's fingers must be considered. An edge $e_{n_a,n_b} \in E_\mathcal{G}$ connects two nodes n_a, n_b if: 1) it is possible for a fingertip to slide along the object from $\mathbf{p}_a$ to $\mathbf{p}_b$; 2) $\Theta_a \cap \Theta_b \neq \emptyset$. As it is a disconnected graph, the DMG is composed of several components $C \subseteq E_\mathcal{G}$, that contain only the nodes connected to each other.

B. Problem Statement and System Overview

The in-hand manipulation problem is stated as: find the sequence of motions $\mathcal{M}$ that moves the gripper from G_i to G_d, given the initial grasp $G_i{=}\langle \mathbf{g}_{i1}, \mathbf{g}_{i2}, \mathcal{Q}_i\rangle$, the desired grasp $G_d{=}\langle \mathbf{g}_{d1}, \mathbf{g}_{d2}, \mathcal{Q}_d\rangle$ and the object shape P_o.

We generate and use a partial Dexterous Manipulation Graph $\mathcal{G}$ that can be exploited for current and future in-hand manipulation tasks with the given object.

Our proposed method contains the following components:

1) Estimate the missing object shape P_r.
2) Build a partial DMG $\mathcal{G}$.
3) Plan motions $\mathcal{M}$ to move from G_i to G_d.

We discuss all of them in detail in the next sections.

III. Shape Reconstruction

In this section, we present the steps of estimating the full object shape P_o from a partial observation P_v. Simply registering the observed point cloud is inadequate in our case, since no object model is initially known. The idea is to exploit a shape prior which encapsulates a family of objects, e.g., hammers with variations on head and handle geometries. The prior can then be reasoned to complement P_v , in a similar way as humans imagine invisible parts by exploiting knowledge about shapes of a certain type of object.

We propose to learn this shape prior as a probabilistic generative model, a denoising version of Variational Auto-encoders (VAE) [34], from a set of object point clouds. The model takes a PointNet encoder [35] and a decoder with coarse intermediate reconstruction [36]. The shape prior is captured as an isotropic Gaussian $p(\mathbf{z}), \mathbf{z} \in \mathbb{R}^{d_z}, d_z \ll |P_v|$, in the latent space. Similar to [37], the full shape is recovered by searching $\mathbf{z}$ given the partial observation P_v:

$$\max_{\mathbf{z}} p(\mathrm{z}|P_v) = \min_{\mathbf{z}} \mathrm{Diff}(P_v, \mathrm{dec}(\mathbf{z})) - \beta \log p(\mathbf{z}), \quad (1)$$

with $P_r \sim \mathrm{dec}(\mathbf{z}^*)$. β denotes a weight balancing the reconstruction error and the plausibility according to the shape prior. The error term measures the distinction between evident and estimated point clouds and here we use a one-way Chamfer distance.

Solving (1) requires a good initialization for the underlying nonconvex optimization. This can be obtained by encoding the partial observation P_v. In practice, the pose of P_v might not be well aligned to training data so we also propose to search a rigid transformation $\mathbf{T} \in SE(3)$ such that the initialization yields good performance:

$$\mathbf{T}^* = \operatorname*{argmax}_{\mathbf{T}} [\max_{\mathbf{z}} p(\mathrm{z}|\mathbf{T}(P_v))|_{\mathbf{z}_{\text{init}} \sim \mathrm{enc}(z|\mathbf{T}(P_v))}]. \quad (2)$$

The inner maximization can be surrogated by taking gradient steps from the parameterized initialization. Empirically we found an expansion at $\mathbf{z}_{\text{init}}$ itself is adequate. The rotational part of $\mathbf{T}$ takes an axis-angle representation and further transforms P_v through Rodrigues' formula. The main axes of P_v are decided by running a PCA and all combinations of axis directions are tried to facilitate the search of $\mathbf{T}^*$.

We generate training data from YCB objects [38], e.g., hammer and spoon, which are different from our test ones. The dataset is augmented by applying scaling and small rotations along the main axes. Point clouds are down-sampled to have a consistent number of points. The proposed VAE learns with raw point cloud data since the encoder and Chamfer distance are permutation-invariant.

IV. Online Dexterous Manipulation Graph

A. DMG Variations

The main differences from [9] are that the new DMG $\mathcal{G}$ is not generated in a separate offline step, and that it does not need the full object shape. Moreover, we change and improve the graph structure as detailed below.

Instead of focusing on all the possible motions that a fingertip can achieve, we link motions to the feasibility of

executing them. Since we address the in-hand manipulation through pushes of the object, motions of the fingertips depend on the possible pushes. In [9] two nodes can be connected despite no push can move the fingertip between them. The infeasible pushes must be carefully excluded at planning time. In this work, since we merge the steps of DMG generation and in-hand manipulation planning, we incorporate in $\mathcal{G}$ only feasible motions.

We propose to represent the new DMG $\mathcal{G}$ as a *directed* graph: in contrast with the DMG in [9], which is an *undirected* graph, the existence of an edge $e_{n_a n_b} \in E_\mathcal{G}$ does not imply the connection $e_{n_b n_a} \in E_\mathcal{G}$.

In addition, we aim at obtaining $\mathcal{G}$ also for objects whose shape is only partially known. We introduce the functions $f(\cdot) : \mathbb{R}^3 \rightarrow [0,1]$, $g(\cdot) : E_\mathcal{G} \rightarrow [0,1]$. These functions are associations of points and edges to their *certainty*. We will also write $f(n_a)$, with $n_a \in N_\mathcal{G}$, by which we mean $f(\mathbf{p}_a)$. The value depends on the position $\mathbf{p}_a$: if $\mathbf{p}_a \in P_v$, $f(\mathbf{p}_a)$ will be high, because that part of the object has been observed; in contrast, if $\mathbf{p}_a \in P_r$, $f(\mathbf{p}_a)$ will be lower. We define $g(e_{n_a n_b}) = \alpha \frac{(f(n_a)+f(n_b))}{2} + (1-\alpha) f(\mathbf{p}_p)$, where $\alpha \in [0,1]$ is a weighting factor and $\mathbf{p}_p$ is the push point associated to the movement defined by the edge.

The partial DMG $\mathcal{G}$ is generated while looking for a solution $\mathcal{M}$ to move from G_i to G_d. The returned solution will be the one with the highest certainty, to reduce the influence of the missing object knowledge on the execution.

B. DMG Generation

The graph $\mathcal{G}$ and the solution $\mathcal{M}$ are generated at the same time. The precedence goes to finding $\mathcal{M}$, but $\mathcal{G}$ is generated to facilitate future manipulations on the same object. To generate the graph, we introduce the following parameters: branching factor $b_e \in \mathbb{N}$, i.e. maximum number of edges from a node; edge length $l_e \in \mathbb{R}^+$, i.e. approximate Euclidean distance between node contact points; component threshold $\zeta \in \mathbb{R}^+$; push point normal threshold $t_{\mathbf{n}_p} \in [0,1]$.

At the beginning, the robot is grasping the object with configuration G_i. We create two nodes associated with it: $n_{i1} = \langle \mathbf{g}_{i1}, \Theta_{i1} \rangle$, $n_{i2} = \langle \mathbf{g}_{i2}, \Theta_{i2} \rangle$. Θ_{i1}, Θ_{i2} are obtained by checking the collisions with the finger and gripper when in contact at the given point, as in [9]. Similarly, G_g is associated with the nodes n_{d1}, n_{d2}. Each node is associated to a different DMG component C_{i1}, C_{i2}, C_{d1}, C_{d2}. The goal is to expand from n_{i1}, n_{i2} and add nodes and edges to $\mathcal{G}$, until connections are found so that we can verify $C_{i1} = C_{d1}$, $C_{i2} = C_{d2}$. In this way, a solution $\mathcal{M}$ can exist.

The nodes n_{i1}, n_{i2} define a grasp line, and we set this grasp line as normal $\mathbf{n}_g$ to the sliding area, regardless of the estimated normals in P_o. We also impose $f(n_{i1}) = f(n_{i2}) = 1$.

The generation procedure is shown in Algorithms 1 and 2, where some steps have been omitted for the sake of conciseness (e.g. when used for planning, the graph generation stops when the goal becomes reachable). The graph is built starting from expanding the nodes n_{i1}, n_{i2} and proceeding in a FIFO manner. We derive b_e angles $\varphi_1, ..., \varphi_{b_e} \in \Phi \subset [0, 2\pi]$. These angles define directions on the object's surface and are used

Algorithm 1 Graph Generation

Input P_o, G_i, G_d, max_it
1: n_{i1}, n_{i2} ←nodes from G_i
2: n_{d1}, n_{d2} ←nodes from G_d
3: $N_\mathcal{G} \leftarrow \{n_{i1}, n_{i2}, n_{d1}, n_{d2}\}$
4: $E_\mathcal{G} \leftarrow \{\}$
5: n_{i1}.opposite← n_{i2}, n_{i2}.opposite← n_{i1}
6: $N_v \leftarrow \{\}$ ▷ keep track of visited nodes
7: q←empty queue
8: q.push(n_{i1})
9: i←0
10: **while** q.size>0 **and** i<max_it **do**
11: i++
12: n ←q.pop()
13: N_v.insert(n)
14: $\Phi \leftarrow n$.expandableAngles
15: **for all** $\varphi \in \Phi$ **do**
16: **if** n.opposite≠Null **then**
17: EXPANDNODE(n, φ)
18: **else**
19: EXPANDNODEWITHOPPOSITE(n, φ)
20: q.remove(n.opposite)

to expand the nodes and form new edges. The procedure of expanding a single node is detailed in section IV-E.

C. Push Points

We rely on pushes on the object to move it toward the desired pose. Hence, it is important to correctly identify a push point $\mathbf{p}_p$ that can be used for exerting the desired push. In this section, we focus on push points assuming translation motions. Similar processes are adopted for rotation, but with more relaxed constraints, as the push point and direction can be changed more freely without affecting the outcome.

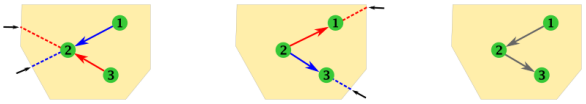
A point $\mathbf{p}_1 \in P_o$ can be associated with (at least) one opposite point $\mathbf{p}_2 \in P_o$, found intersecting the object with the ray defined by $\mathbf{p}_1$ and the vector opposite to its normal to the surface. Given a translation motion $\mathbf{t}$ from $\mathbf{p}_1$, the corresponding $\mathbf{p}_p \in P_o$ is found by intersecting the object with the ray defined by $\frac{\mathbf{p}_1+\mathbf{p}_2}{2}$ and $-\mathbf{t}$. In case of multiple solutions, we choose $\mathbf{p}_p$ as the furthest away from $\mathbf{p}_1$.

To analyze feasible motions, we rely on the concept of *motion cone* [39], although we do not measure friction parameters and exploit a predefined tolerance. $\mathbf{t}$ is achievable if the normal to the object's surface $\mathbf{n}_p$ at the corresponding push point $\mathbf{p}_p$ is aligned with it. To consider the uncertainty in the object's shape, we verify it as

$$\left| \mathbf{n}_p \cdot \frac{\mathbf{t}}{\|\mathbf{t}\|} \right| \geq \xi t_{\mathbf{n}_p}, \tag{3}$$

with $\xi \in [0,1]$ being a chosen adjustment for normal alignment depending on $f(\mathbf{p}_p)$. We prefer to relax the constraint when the object's shape is uncertain, and then correct this assumption afterwards rather than not considering the push admissible since the start. Uncertain pushes will be considered with lower priority, so that preferred pushes are those on the visible part of the object.

Fig. 1a shows two examples of translations, and Fig. 1b shows the opposite motions. The blue arrows show feasible translations, considering the normal at the push point, which



(a) (b) (c)

Fig. 1: Example of relation between graph edges and push directions.

is traceable through the dotted line; infeasible translations are in red. The constraint (3) results in a connectivity between those points in $\mathcal{G}$ as shown in Fig. 1c. Noticeably, it is possible to move from point 1 to point 3, but not vice-versa. This non reversibility is why we changed the DMG structure to be a *directed* graph.

D. DMG Components

A component $C_a \subseteq N_\mathcal{G}$ contains all nodes between which it is possible for a single finger to slide or rotate without releasing the grasp. Due to the use of parallel grippers, motions are constrained to be locally planar along smooth surfaces. We associate a frame Σ_{C_a} with each component. By convention, we assume that the x-axis $\hat{\mathbf{x}}_{\Sigma_{C_a}}$ is aligned with the normal to the planar surface associated with the component. Given $n_a \in C_a$, a necessary condition for a neighbor node n_b to belong to C_a is

$$|\hat{\mathbf{x}}^T_{\Sigma_{C_a}}(\mathbf{p}_b - \mathbf{p}_a)| < \zeta, \tag{4}$$

where $\zeta \in \mathbb{R}^+$ is a threshold to enable tolerance of slightly curved surfaces and account for noise in perception. The condition in (4) is not a sufficient condition, because a component must contain nodes that are connected. Note that the same point $\mathbf{p}_a$ can correspond to more nodes $n_{a1}{=}\langle\mathbf{p}_a, \Theta_{a1}\rangle$, ..., $n_{aM}{=}\langle\mathbf{p}_a, \Theta_{aM}\rangle$, and these nodes can belong to separate components. Similarly, certain translations might be infeasible due to (3) not being fulfilled.

E. Node Expansion Step

A node n_a is associated with a set of *expandable angles* Φ_a. For each $\varphi \in \Phi_a$, the expansion of n_a is detailed in the following and summarized in Algorithm 2.

A new point $\mathbf{p}_b$ in P_o is derived as

$$\mathbf{p}_b = \underset{\mathbf{p}\in P_o}{\operatorname{argmin}} \|\mathbf{p} - (\mathbf{p}_a + l_e \mathbf{R}_{\hat{\mathbf{x}}_{\Sigma_{C_a}}}(\varphi)\ \hat{\mathbf{y}}_{\Sigma_{C_a}})\|, \tag{5}$$

where $\mathbf{R}_{\hat{\mathbf{x}}_{\Sigma_{C_a}}}(\varphi)$ is the axis-angle rotation matrix, and $\hat{\mathbf{y}}_{\Sigma_{C_a}}$ is the y-axis of frame Σ_{C_a}. If $\mathbf{p}_b$ does not satisfy (4), then no edge will be added. If there are no nodes in proximity of $\mathbf{p}_b$, then (at least) one new node n_b is added to $N_\mathcal{G}$, associated with a new component C_b. If a finger in contact at $\mathbf{p}_b$ can rotate between B disconnected angle sets $\Theta_{b1}, ...\Theta_{bB}$, then B disconnected nodes are added.

If (4) is satisfied, then the translation $\mathbf{t}{=}\mathbf{p}_b{-}\mathbf{p}_a$ from $\mathbf{p}_a$ is associated to the push point $\mathbf{p}_{pa}$. If (3) is fulfilled, then there is the chance to add new edges to the graph.

If $\mathbf{p}_b$ is close (i.e. within distance λ) to (at least) one existing node n_b, then for each node n_a^j with position $\mathbf{p}_a$ an edge $e_{n_a^j n_b}$ is added to $E_\mathcal{G}$ only if $\Theta_a^j \cap \Theta_b \neq \emptyset$. Notice that despite starting from n_a, other nodes can be expanded because to the same position $\mathbf{p}_a$ more nodes might correspond. If $C_b \neq C_a^j$, then we replace C_a^j with $C_a^j \cup C_b$, and C_b is deleted; the axes of the frame $\Sigma_{C_a^j}$ are updated based on all the new nodes to satisfy (4). If $\mathbf{p}_b$ is not in proximity of any node, then J new nodes n_b^j are added to $N_\mathcal{G}$, with J the number of orientations $\Theta_b^1, ..., \Theta_b^J$. For each n_a^j, $e_{n_a^j n_b^j}$ is added to $E_\mathcal{G}$ and n_b^j to C_a^j only if $\Theta_a^j \cap \Theta_b^j \neq \emptyset$.

At the same time, the translation $-\mathbf{t}$ from $\mathbf{p}_b$ is associated with the push point $\mathbf{p}_{pb}$. If $\mathbf{p}_{pb}$ satisfies (3), then the edges towards n_a are examined and added in the same way as the edges that depart from it.

Some nodes have an opposite node associated (e.g. n_{i1}, n_{i2}). In this case, the expansion is executed for both nodes at the same time, along the same direction. The vector $\hat{\mathbf{x}}_{\Sigma_C}$ is replaced by the one defined by the grasp line $\mathbf{l}_g{=}\frac{\mathbf{p}_{i2}-\mathbf{p}_{i1}}{\|\mathbf{p}_{i2}-\mathbf{p}_{i1}\|}$. If both components C_{i1} and C_{i2} are expanded, then the nodes keep carrying the opposite node with them, otherwise the expansion will be executed with only one node. See lines 1-7 in Algorithm 2.

When a node n_b is generated or reached from an expansion obtained with angle φ, the opposite angle φ' is removed from Φ_b. φ' is an approximation of $\varphi{+}\pi$. Therefore, n_b will not be expanded along an area that has already been explored.

Fig. 2 shows a simple example of an expansion step. The initial $\mathcal{G}$ is shown in Fig. 2a. Each component corresponds to a different color. The node $n_a \in C_3$ is expanded in Fig. 2b, into three points. The first one is associated with the node

Algorithm 2 Node Expansion Step

```
1:  procedure ExpandNodeWithOpposite(n_a, φ)
2:      n_b ← n_a.opposite
3:      C_a.setAxis(l_g)
4:      C_b.setAxis(−l_g)
5:      N_a ←ExpandNode(n_a, φ)
6:      N_b ←ExpandNode(n_b, φ)
7:      assignOpposites(N_a, N_b)                    ▷ based on C_a, C_b
8:  procedure ExpandNode(n_a, φ)
9:      p_b ←from (5)
10:     if not p_b close to any n ∈ N_G then
11:         for all Θ_x for finger in contact at p_b do
12:             n_x ← ⟨p_b, Θ_x⟩                     ▷ create new node
13:             N_G.insert(n_x)
14:             C_x ← {n_x}                          ▷ new (temporary) component
15:             if n_x ∉ N_v then
16:                 q.push(n_x)
17:     N_x ←{n_x : N_G s.t. ‖p_x−p_b‖<λ}
18:     if (4) then
19:         p_p^from ←pushPoint(p_a, p_b)
20:         p_p^to ←pushPoint(p_b, p_a)
21:         for all n_x ∈ N_x do
22:             n_x.removeExpandableAngle(φ+π)
23:             if Θ_x ∩ Θ_a ≠ ∅ then
24:                 if p_p^from verifies (3) then
25:                     E_G.insert(e_{n_a n_x})
26:                 if p_p^to verifies (3) then
27:                     E_G.insert(e_{n_x n_a})
28:                 if at least one edge added then
29:                     C_a ← C_a ∪ C_x              ▷ Σ_{C_a} must be updated
30:                     delete(C_x)
31:     return N_x
```

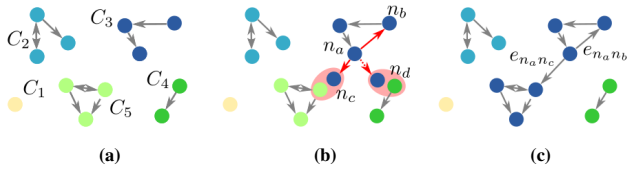


Fig. 2: Example of graph expansion step. Nodes of the same color belong to the same component.

n_b. Since $n_b \in C_3$, $\Theta_a \cap \Theta_b \neq \emptyset$, and (3) is verified, the edge $e_{n_a n_b}$ is added to $\mathcal{G}$. The next point is close to $n_c \in C_5$; (3) is verified, and n_c satisfies (4) for belonging to C_3; moreover, $\Theta_a \cap \Theta_c \neq \emptyset$. Therefore, C_3 is replaced with $C_3 \cup C_5$, C_5 is removed from $\mathcal{G}$ and the edge $e_{n_a n_c}$ is added. Finally, the last point is close to the existing node $n_d \in C_4$. But, in this case, (3) is not verified, so no connection is added and no component is changed. The updated $\mathcal{G}$ is shown in Fig. 2c.

F. In-Hand Manipulation Planning

Once $\mathcal{G}$ is obtained by expanding nodes starting from n_{i1}, n_{i2}, it is used to plan in-hand manipulation. $\mathcal{G}$ is searched for a path from n_{i1}, n_{i2} to n_{d1}, n_{d2}. We use the same Dijkstra procedure as [9] to account for both fingers moving at the same time on the object's surface, i.e. indicating a *principal* finger, and a *secondary* finger that follows it in the opposite node in $\mathcal{G}$ to ensure the validity of the whole path. However, we relax the constraint for the secondary node based on the value of f. For instance, a translation of the principal finger from n_a to n_b corresponds to a translation of the secondary finger from n_c to n_d. If $C_c \neq C_d$, this solution should not be allowed, but we admit it if $f(n_c)$ and $f(n_d)$ are small, so that solutions are still considered and we account for possible errors in shape reconstruction.

Unlike in [9], the feasibility of pushes is already contained in $\mathcal{G}$, so it is not necessary to check it at planning time. Instead of obtaining the shortest path, we prefer to maximize the *certainty* of it, i.e. find the sequence of edges so that

$$e_{n_0 n_1}, ..., e_{n_{H-1} n_H} = \text{argmax} \sum_{i=1}^{H} g(e_{n_{i-1} n_i}), \qquad (6)$$

where $n_0{=}n_{i1}$ and $n_H{=}n_{d1}$.

The solution from DMG is a path of nodes; this path has to be converted into $\mathcal{M}$ so that it can be executed. An edge $e_{n_a n_b} \in E_{\mathcal{G}}$ is easily converted into a translation $\mathbf{t}{=}\mathbf{p}_b{-}\mathbf{p}_a$, which can be executed by pushing against the corresponding push point. In contrast, rotations are not directly expressed in the DMG solution. We prefer to introduce rotations only when necessary, to minimize the amount of pushes the robot has to execute. Given two consecutive nodes n_a, n_b, and the gripper's finger in contact at $\mathbf{p}_a$ grasping with angle θ_a, a rotation ϕ is introduces only if $\theta_a \notin \Theta_b$. A new angle $\theta_b \in \Theta_a \cap \Theta_b$ is chosen. We choose θ_b to minimize the amount of rotations in the overall path, but different options can be used (e.g. minimize $|\theta_a - \theta_b|$, reduce the distance from goal angle, etc).

V. Experiments

A. Building DMG Online

In this section we evaluate our new DMG computation method and compare with the planning time of the offline DMG [9]. Since the original offline DMG uses the full object's shape, we used the full object point-cloud in our new DMG using objects from the YCB set [38], to provide both methods with the same input, and we disregard the object reconstruction for the results in this section. The parameters of the offline DMG were: seed resolution 1.3 cm, angle resolution 10°, normal threshold 0.17. We refer to [9], [17] for details on these parameters and their effects on the solution. For our online DMG, we used the following parameters, chosen so that the graph outcome would be similar to the one of the offline DMG method in terms of connectivity and distance between node points: $l_e{=}0.012$, $b_e{=}5$, $t_{\mathbf{n}_p}{=}0.8$, $\xi{=}1$, $\zeta{=}0.002$, $\lambda{=}\frac{l_e}{2}$. We represented Θ with a discretization of 10° to follow what was used in the offline DMG. The maximum iterations were 200, but we stopped the graph generation once the goal was reachable.

Fig. 3 shows a comparison in terms of time required for finding a solution for given in-hand manipulation tasks using two of the YCB objects: gelatin box and potted meat can. The offline DMG requires a first step to obtain the graph, which is then used for fast planning on the same object. In our online DMG method, the graph is built incrementally whenever a new task is added. We report times for finding solution for the tasks when no graph has been built (i.e. the object is seen for the first time), labeled online DMG from start. Compared to the offline DMG, these times are roughly 1 or 2 orders of magnitude smaller.

Then, we show how building the DMG progressively reduces the required time for planning each task, because a partial graph is already available, by executing the tasks in sequence (online DMG 1-2-3 sequence).Moreover, when a task refers to nodes that have already been added to the graph, the planning time decreases even more, by 1 order of magnitude, as it is not necessary to expand the graph to explore not covered areas (online DMG 1-2-1 sequence).

The planning time of the offline DMG once the full DMG

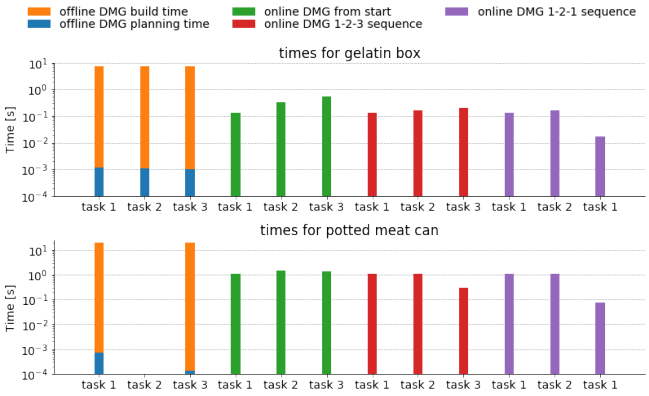


Fig. 3: The amount of time spent on a task, by the offline DMG [9] and by the proposed online DMG. For task 2 in potted meat can, the offline DMG failed to find a solution. The y axis is in logarithmic scale.

TABLE I: Planned solution errors with full object shape

	gelatin box	potted meat can	hammer
err_p (cm)	0.56	1.01	0.52
$err_o\%$	3.2	4.8	4.8

TABLE II: Planned solution errors with partial object shape

	red hammer	black hammer	wooden spoon
err_p (cm)	0.50	0.47	0.67
$err_o\%$	4.7	2.7	4.4
no solution %	13	20	12

is built is still lower, but we believe this is due to the fact that the online method uses point clouds, while the offline method relies of the full object shape as a mesh, making it faster to check for the *secondary* finger's position during the graph search. However, we rely on the point cloud because this method is built to take into account direct sensory input.

Table I shows the average errors of 15 different tasks in the planned solution by online DMG for 3 of the YCB objects, obtained by simulating (kinematics only) reaching the desired grasps by using the found $\mathcal{M}$. Since in this case there is no reconstruction uncertainty, instead of using (6) to plan the solution, we use the shortest path criteria, as in [9].

The distance between $\mathbf{g}_{i_1}$ and $\mathbf{g}_{d_1}$ varied within 1.8 and 7.8 cm (5.9 in average). The desired rotation around the grasp line varied within 0° and 90°. err_p and $err_o\%$ are respectively the average position and average orientation errors, as defined in the benchmark in [40].

Irregular and noisy object's shape increased the challenge in finding a good solution with the used parameters. A relaxation of thresholds might lead to better results when the object's reconstruction is noisy, and we use this strategy in the next section.

B. DMG with Unknown Shapes

In this section, we analyze the performance of our method including the shape reconstruction. We used a Kinect v2 sensor, mounted on top of an ABB Yumi robot, to collect the visible object P_v. We removed the robot's body from the depth image using [41]. We placed objects inside the robot's gripper and defined the desired grasp with respect to the initial grasp configuration. We used ζ=0.003, ξ=0.5 if $f(\mathbf{p}_p)$<0.9, and 1 otherwise, and increased the maximum iterations to 300 to account for sensor noise and sparse reconstruction, b_e=8, l_e=0.01 and

$$f(\mathbf{p}) = \begin{cases} 0.9 & \text{if } \mathbf{p} \in P_v \\ 0.5 & \text{if } \mathbf{p} \in P_r \\ f(\tilde{\mathbf{p}}^\star) & \text{otherwise; } \tilde{\mathbf{p}}^\star = \underset{\tilde{\mathbf{p}} \in P_o}{\operatorname{argmin}} \|\mathbf{p} - \tilde{\mathbf{p}}\|_2 \end{cases}.$$

We trained the VAE with point clouds from the YCB objects. On the test set of augmented data, the obtained reconstruction Chamfer error relative to three object dimensions were 0.78%, 2% and 7.8%, respectively. In particular, for the presented experiments, we used two classes: hammer and spoon. Then, we run our in-hand manipulation method on objects grasped by the robot that were of the same kind but different shape from the objects present in the YCB set.

Table II shows the average errors over the different tasks in the planned in-hand manipulation solution when the object's shape is only partially visible. It also shows the times in which our method failed in finding a solution, although a solution exists. The distance from the initial to the desired grasp varied between 1.0 and 7.2 cm. We noticed that the main challenge in this situation is due to the sparsity of P_r with respect to P_v, which constitutes a source of error in the estimation of normals to the surface.

The higher failure rate for the black hammer with respect to the red hammer is likely due to a higher difference from the YCB hammer. Because of this difference, the reconstructed shape was not as close to the real object as it was for the red hammer, but it was still sufficient to provide good solution most of the times.

C. Robot Experiments

This sections evaluates the new DMG method, including object shape reconstruction, by executing the planned solution on an ABB Yumi robot. For this execution, we used the same control strategy of [9]. The pushes on the object were executed by using Yumi's second arm. The only difference from [9] is that in the dual-arm push we aligned the orientation of the pusher gripper's fingers (against which the object is pushed) with the desired translational motion, so that the side of the fingers in contact with the object's surface would be aligned with it, in order to minimize slippage and unwanted motions.

We used the same red hammer and corresponding tasks as in section V-B. We placed an April tag [42] on the object to measure the relative displacement between initial and final grasp frames Σ_g, Σ_g', and compare it with the desired change in the grasp. The average errors were err_p=0.64 cm and $err_o\%$=7.6. In our experiments, the robot was able to adjust the object inside the gripper to the desired pose by using the planned in-hand path, converted into corresponding pushes against the object. In fact, despite slightly higher errors compared to the planned in-hand manipulation (Table II), the execution outcome can still be considered successful.

VI. Conclusions and Future Work

We presented an improvement over the Dexterous Manipulation Graph method for in-hand manipulation planning. The DMG can now take into account partial object shapes, it is generated progressively as manipulation tasks are given and takes into account the feasibility of the in-hand motions in the connections between nodes.

As future work, we plan to exploit the incorporation of uncertainty in the DMG to adapt the graph structure as new information on the object becomes available. In fact, after one or more pushes, more parts of the object shape become visible. This new information can be added to the graph by correcting the uncertain nodes and the uncertain edges. This will provide an instrument to adapt and improve the in-hand manipulation graph and the in-hand manipulation solution.

REFERENCES

[1] 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.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, April 2014.

[3] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, 2019.

[4] R. Antonova, M. Kokic, J. Stork, and D. Kragic, "Global search with bernoulli alternation kernel for task-oriented grasping informed by simulation," in *Conference on Robot Learning*, 2018.

[5] D. Rodriguez, C. Cogswell, S. Koo, and S. Behnke, "Transferring grasping skills to novel instances by latent space non-rigid registration," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2018.

[6] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 2442–2447.

[7] Q. Li, C. Elbrechter, R. Haschke, and H. Ritter, "Integrating vision, haptics and proprioception into a feedback controller for in-hand manipulation of unknown objects," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 2466–2471.

[8] H. van Hoof, T. Hermans, G. Neumann, and J. Peters, "Learning robot in-hand manipulation with tactile features," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 121–127.

[9] 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.

[10] 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.

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

[12] R. Higo, Y. Yamakawa, T. Senoo, and M. Ishikawa, "Rubik's cube handling using a high-speed multi-fingered hand and a high-speed vision system," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 6609–6614.

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

[14] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," 2018.

[15] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 378–383.

[16] 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.

[17] S. Cruciani, K. Hang, C. Smith, and D. Kragic, "Dual-arm in-hand manipulation and regrasping using dexterous manipulation graphs," *arXiv preprint arXiv:1904.11382*, 2019.

[18] N. Chavan-Dafle and A. Rodriguez, "Prehensile pushing: In-hand manipulation with push-primitives," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 6215–6222.

[19] D. L. Brock, "Enhancing the dexterity of a robot hand using controlled slip," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, April 1988, pp. 249–251 vol.1.

[20] 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.

[21] 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.

[22] N. Chavan-Dafle and A. Rodriguez, "Sampling-based planning of in-hand manipulation with external pushes," in *International Symposium of Robotics Research*, December 2017.

[23] F. E. Viña B., Y. Karayiannidis, K. Pauwels, C. Smith, and D. Kragic, "In-hand manipulation using gravity and controlled slip," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 5636–5641.

[24] R. Antonova, S. Cruciani, C. Smith, and D. Kragic, "Reinforcement learning for pivoting task," *arXiv preprint arXiv:1703.00472*, 2017.

[25] S. Cruciani, K. Hang, C. C. Smith, and D. Kragic, "Dual-arm in-hand manipulation using visual feedback," in *IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, Oct 2019, pp. 411–418.

[26] B. Calli and A. M. Dollar, "Robust precision manipulation with simple process models using visual servoing techniques with disturbance rejection," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 406–419, Jan 2019.

[27] M. Popović, G. Kootstra, J. A. Jørgensen, D. Kragic, and N. Krüger, "Grasping unknown objects using an early cognitive vision system for general scene understanding," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2011, pp. 987–994.

[28] M. Li, K. Hang, D. Kragic, and A. Billard, "Dexterous grasping under shape uncertainty," *Robotics and Autonomous Systems*, vol. 75, pp. 352 – 364, 2016.

[29] R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragic, "Generalizing grasps across partly similar objects," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3791–3797.

[30] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 2845–2850.

[31] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 82–90.

[32] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas, "Learning representations and generative models for 3d point clouds," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[33] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[34] D. P. Kingma and M. Welling, "Stochastic gradient vb and the variational auto-encoder," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[35] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[36] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 International Conference on 3D Vision (3DV)*, 2018, pp. 728–737.

[37] H. Yin, F. S. Melo, A. Billard, and A. Paiva, "Associate latent encodings in learning from demonstrations," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Francisco, USA, 2017.

[38] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

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

[40] S. Cruciani, B. Sundaralingam, K. Hang, V. Kumar, T. Hermans, and D. Kragic, "Benchmarking in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 588–595, April 2020.

[41] [Online]. Available: https://github.com/blodow/realtime_urdf_filter

[42] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.