2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Learning Category-Level Manipulation Tasks from Point Clouds with Dynamic Graph CNNs

Junchi Liang and Abdeslam Boularias[1]

***Abstract*— This paper presents a new technique for learning category-level manipulation from raw RGB-D videos of task demonstrations, with no manual labels or annotations. Category-level learning aims to acquire skills that can be generalized to new objects, with geometries and textures that are different from the ones of the objects used in the demonstrations. We address this problem by first viewing both grasping and manipulation as special cases of *tool use*, where a tool object is moved to a sequence of key-poses defined in a frame of reference of a target object. Tool and target objects, along with their key-poses, are predicted using a dynamic graph convolutional neural network that takes as input an automatically segmented depth and color image of the entire scene. Empirical results on object manipulation tasks with a real robotic arm show that the proposed network can efficiently learn from real visual demonstrations to perform the tasks on novel objects within the same category, and outperforms alternative approaches.**

I. Introduction

Category-level learning is an increasingly popular approach to training robots to manipulate unknown objects in real-world environments. A sequence of images showing how to perform a certain manipulation task is provided to the robot by a human demonstrator. The robot is then tasked with learning a manipulation policy that can be tested on objects other than those used in the demonstrations.

Prior works in category-level robot learning were mostly focused on grasping problems [1]–[7]. While there are numerous learning-based techniques that generalize grasping skills to novel objects, only a few recent ones have been shown capable of generalizing other types of manipulation skills, such as placing, painting, or pouring, to novel objects [8]–[20]. However, most of these techniques rely on annotated images wherein a human expert manually specifies *keypoints* on training objects.

In this work, we propose a new technique for learning category-level manipulation skills from unlabeled RGB-D videos of demonstrations. The proposed system is fully autonomous, and does not require any human feedback during training or testing besides the raw demonstration videos. The setup of the system includes a support surface wherein a number of unknown objects are placed, which includes task-irrelevant distractive objects. The system is composed of a high-level policy that selects a tool and a target from the set of objects in the scene at each step of the manipulation, an intermediate-level policy that predicts desired 6D keyposes for the robot's wrist, and a low-level policy that moves the wrist to the keyposes. The system employs a dynamic graph convolutional neural network that receives as inputs partial point clouds of the objects. A local frame of reference is computed for each object based on the principal component analysis of its point cloud. The 6D keyposes predicted by the network are in the frame of the predicted target object.

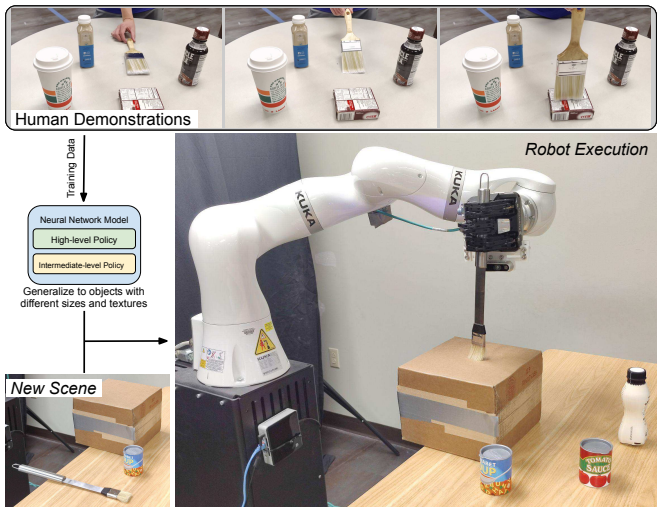

Fig. 1: System overview and robotic setup used in the experiments. In this example, the system is trained by unlabeled visual demonstrations to pick up a paint-brush in a specific manner, and to place it in a specific configuration relative to another object. The robot is tasked with repeating the demonstrated behaviour on a new scene containing novel objects with different sizes and texture.

The proposed system is tested on a real robot with demonstrations of four tasks: two variants of stacking, pouring, and painting. Pouring is performed with beads instead of liquid, for the safety of the robot. The exact same architecture and parameters are used for learning the four different tasks, the only difference is the provided visual demonstrations.

The key novel contributions of this work are as follows. (1) An efficient *new architecture* for learning category-level manipulation tasks. A key feature of this architecture is the capability to generate trajectories of the robot's end-effector according to the 3D shapes of the objects that are present in the scene. The proposed architecture can thus generalize to objects with significantly different sizes and shapes. This is achieved through the use of a dynamic version of graph neural networks, wherein the graph topology is learned based on the demonstrated task. Moreover, the proposed architecture can be used in scenes that contain an arbitrary number of objects. This is in contrast with most existing manipulation learning methods, which are limited to objects of similar dimensions. (2) An *empirical study*

[1]The authors are with the Department of Computer Science of Rutgers University, Piscataway, New Jersey 08854, USA. This work is supported by NSF awards 1846043 and 2132972.

comparing various architectures for learning manipulation tasks, using real objects and robot. The study shows that, amongst the compared methods, a dynamic GNN provides the most data-efficient architecture for learning such tasks. Furthermore, the same hyper-parameter values can be used to learn different tasks. (3) A *new formulation* of the problem by representing states as lists of relative 6D poses of object pairs. The proposed formulation also *unifies* the problems of grasping and manipulation by treating the end-effector as one of the objects in the scene, and viewing grasping as a special type of tool-use. (4) A *fully self-supervised* pipeline that does not require manual task decomposition, in contrast with existing category-level techniques [8]–[15], [21].

II. Related Work

Category-Level Manipulation. Recently, several techniques have been devised for learning to generalize robot manipulation skills to new objects in the same category, such as cups of different sizes, shapes and texture. Most of these techniques use semantic 3D keypoints to represent objects [21]. Keypoints were initially proposed as grasp points for the manipulation of deformable objects [8], [9]. The technique presented in [10] consists in manually annotating task-relevant keypoints on a large number of training objects, and trains an *integral neural network* to align the keypoints between intra-class object instances. The need for tedious human annotation was removed in [11] through the use of a robotic system that automatically re-arranges objects. Contrastive learning was then employed to learn feature descriptors of 2D image points [12]. Similar object-centric dense descriptors were used for learning pick-and-place tasks from demonstrations [13]. In [14], task-specific keypoints are learned from self-supervised robot interactions instead of human annotations. Semantic category-level 3D keypoints were also trained in [15] using a combination of self-supervised training and human annotations.

Category-Level Grasping. Most recent grasping techniques focus on learning directly grasp success probabilities from data [1], [3]–[7], [22]–[27]. Recent methods include training a hierarchy of supervisors from demonstrations to grasp objects in clutter [28], and convolutional neural networks, such as Dex-net [29], that are trained to detect grasp 6D poses in point clouds [30]. These data-driven techniques, however, provide grasps that are valid only for picking up objects, without taking into account the desired manipulation task. In our proposed method, the robot learns from demonstration different types of grasps for different manipulation tasks, such as pouring or stacking. Goal-conditioned grasping was considered in recent works, such as [31]. A dataset of 3D CAD models is required in [31] to learn grasps in simulation. Our method requires instead only a small number of demonstration videos without the need for CAD models.

III. Proposed Approach

A. Proposed Problem Formulation

A human demonstrates a grasping-and-manipulation task that involves picking up a rigid **tool** object from a tabletop that contains an arbitrary set of objects of various types, and moving the object to a 6D pose with respect to a rigid **target** object. Examples of such tasks include: stacking, where the demonstrator picks up a box and puts it on top of a second box; pouring, where the demonstrator picks up a cup or a bottle and places it in a position and rotation relative to another container so that liquid can flow between the two objects. No priors or 3D models of the objects are given, the objects are completely unknown. From only raw RGB-D videos of the demonstrations, and without any labeling or annotation, a robot is tasked with re-producing the tasks on new scenes containing only new objects and not including any of the objects that appeared in the demonstrations.

We denote the set of demonstrations by $T = \{\tau^1, \ldots, \tau^n\}$, wherein each demonstration $\tau^i = (s_0^i, s_1^i, \ldots s_h^i)$ is a sequence (or trajectory) of recorded scene states s_t^i at different time-steps $t \in \{0, \ldots, h\}$. Note that different demonstration trajectories can have different lengths. A state is a tuple $s_t^i = (< P_{0,t}^i, I_{0,t}^i >, < P_{1,t}^i, I_{1,t}^i >, \ldots, < P_{m,t}^i, I_{0,t}^i >)$, wherein each element $P_{j,t}^i$ is a point cloud that corresponds to an object j at time t, and $I_{j,t}^i$ is its cropped RGB image. The list of objects in the scene includes the hand of the human demonstrator during training, and the robotic hand during testing. These two are treated similarly to the other objects, as we argue in this work that grasping is only a special case of tool use, with the hand being the tool object. In the first time-step $t = 0$ of a demonstration τ^i, point clouds $\{P_{j,0}^i\}_{j=0}^m$ are obtained by segmenting the scene's depth image. Each point cloud is then tracked over time, and updated based on the images received at time-steps $t \in \{t, \ldots, h\}$. The sets of objects used in the demonstrations and during testing are denoted by $\mathcal{O}_{\text{training}}$ and $\mathcal{O}_{\text{testing}}$, respectively, with $\mathcal{O}_{\text{training}} \cap \mathcal{O}_{\text{testing}} = \emptyset$. At the beginning of each demonstration, objects are randomly drawn from $\mathcal{O}_{\text{training}}$ and placed on the support surface. Therefore, each scene contains a number of "distraction" objects that are not relevant to the demonstrated tasks. Presence of distraction objects reflects how real-world manipulation tasks are performed in uncontrolled environments.

In a scene $s_t = (< P_{0,t}, I_{0,t} >, \ldots, < P_{m,t}, I_{0,t} >)$, each point cloud $P_{i,t}$ is assigned an intrinsic frame of reference $X_{i,t} = (c_{i,t}, \vec{x}_{i,t}, \vec{y}_{i,t}, \vec{z}_{i,t})$, wherein $c_{i,t}$ is the 3D centroid of the point cloud, and $\vec{x}_{i,t}, \vec{y}_{i,t}$ and $\vec{z}_{i,t}$ are the principal, secondary and tertiary axes of the 3D point cloud of object j at time t, all expressed in the camera's coordinates system. The intrinsic frames of reference are computed automatically by performing a PCA on $P_{i,t}$. A **high-level policy**, denoted by π_h, is used to select a pair of (tool, target) objects from the set of objects present in the scene. This policy receives as input state s_t and returns $o_{\text{tool}} \in \{0, \ldots, m\}$ and $o_{\text{target}} \in \{0, \ldots, m\}$. A **intermediate-level policy**, denoted by π_m, is used to select a desired **keypose** $K \in \mathbb{R}^3 \times \mathbb{SO}(3)$ of the robotic arm's wrist. The intermediate-level policy takes as input state s_t as well as the (tool, target) objects returned by the high-level policy. Finally, we denote by π_l a **low-level policy**. The policy receives as inputs the current robot configuration in a world coordinates frame, denoted by $c_t \in$

$\left(\mathbb{R}^3 \times \mathbb{SO}(3)\right)^J$, where J is the number of joints of the robot arm/hand, in addition to a desired keypose K of the robotic arm's wrist. The policy returns changes Δc_t to apply on c_t to move the robot's wrist to keypose K.

B. System Overview

The proposed system receives as input at each step an RGB-D image of the scene containing an end-effector and a number of unknown objects. The system returns a pair of tool and target objects, and a 6D key-pose that indicates where the tool's frame of reference should be displaced to relative to the target's frame. The system also returns a sequence of low-level actions that move the robot's joints so that the tool is placed in the returned key-pose. At the beginning of a task, the system always selects the robotic hand as the tool object. The robotic gripper is initially open, and closes once it reaches the 6D keypose generated by the system. In a pouring task for example, the first target selected by the system is a bottle that the robot needs to grasp. In a small-on-large stacking task, the first target would be the second-largest box. Once an object is grasped, which is detected by the system from the point cloud input, the system returns a different pair of (tool, target) objects. In our pouring example, the grasped bottle becomes now the new tool object, and a second object (a cup, for example) is returned by the system as the new target object. In the small-on-large stacking example, the grasped box becomes the tool and the largest box becomes the target object. This process is repeated until the task is performed.

C. Dynamic Graph CNNs

At the heart of the proposed system lies a Dynamic Graph Convolutional Neural Network (DGCNN) [32]. DGCNN is trained to extract task-relevant shape features of an object. An object is given as a partial point cloud $P = \{p_0, \dots, p_l\}$ wherein $p_i \in \mathbb{R}^3$ are the coordinates of a point in the camera frame. Each point is connected to its k-nearest neighbors in P. The points in P and their connections form a graph structure, which is given as input to the DGCNN. The first layer of the network performs an edge convolution on the object and returns a set of feature vectors $X = \{x_0, \dots, x_l\}$, one for each point $p_i \in P$. A feature vector $x_i \in X$ is computed as $x_i = \max_{p_j \in k\text{NN}(p_i)} h_\Theta(p_i, p_j)$ where h_Θ is a function parameterized by $\Theta = (\theta, \phi)$ and defined as $h_\Theta(p_i, p_j) = \text{ReLU}\big(\theta.(p_i - p_j) - \phi.p_i\big)$. Set X of feature vectors returned from the first layer is used to form a new graph that is given to the second layer, wherein the same operations are repeated using different values for weights Θ. And so on, this process is repeated for a number of layers, followed by fully connected layers that return a feature descriptor of the entire object. In each layer of the network, the structure of the graph is dynamically re-defined by connecting each point $x \in X$ to its k-nearest neighbors from X. This is different from standard graph CNNs, where the graph structure is defined in the input and kept fixed inside the network. The resulting architecture is more expressive than regular GCNNs, as points from distant parts of an object can become neighbors in later layers, depending on the task.

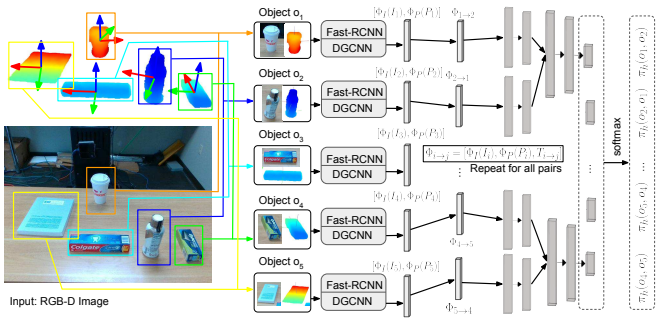

Fig. 2: High-level policy

D. Architecture

High-level policy. The high-level policy receives as input a state s_t and a candidate pair of objects (i, j) and returns $\pi_h(i, j)$, the probability that (i, j) is indeed the pair of tool and target objects needed for performing the manipulation task that the system was trained on. This operation is repeated on all pairs of objects in the scene, and the pair that receives the highest probability $\pi_h(i, j)$ is selected and forwarded to the intermediate policy. To compute π_h, we start by extracting features of each pair (i, j) of the objects present in the scene. For object j paired with object i, we extract a descriptor vector $\Phi_{j\to i}$ of size 1548 from its point cloud P_j and corresponding cropped RGB image I_j. The descriptor vector is defined as $\Phi_{j\to i} = [\Phi_I(I_j), \Phi_P(P_j), T_{j\to i}]$. The first component $\Phi_I(I_j)$ is a vector of size 1024 obtained from Fast-RCNN [33] for the RGB features. The second component $\Phi_P(P_j)$ is a vector of size 512 returned by the DGCNN module, as explained in Sec. III-C, for the shape features. The last component $T_{j\to i}$ is a vector of size 12 that represents the transformation (translation and rotation) of X_j to X_i, wherein X_j and X_i are respectively the intrinsic frames of reference of objects j and i, computed after performing a PCA on each of their point clouds (Sec. III-A). Instead of keeping the camera's coordinates system, we take advantage of our pairing input structure, and express an object's PCA pose in the coordinates system of its counterpart, object i. With this input, the network can avoid irrelevant influences from different camera poses and focus on motions between the target and the tool objects.

The backbone of the high-level policy is a class-agnostic network that takes as inputs $\Phi_{j\to i}$ and $\Phi_{i\to j}$ and returns a score $F_\theta(\Phi_{j\to i}, \Phi_{i\to j}) \in \mathbb{R}$ for every candidate pair (i, j), wherein θ are the network's parameters. From these outputs, we define $\pi_h(i, j) = \text{softmax}_{(i',j')} F_\theta(\Phi_{j'\to i'}, \Phi_{i'\to j'})$ by normalizing over all pairs in the scene, and we set $(o_{\text{tool}}, o_{\text{target}}) = \arg\max_{(i,j)} \pi_h(i, j)$.

Because the policy's probabilities $\pi_h(i, j)$ are obtained by normalizing the network's outputs over all pairs in the scene, the size of the network is decoupled from the number of objects and their order. In contrast, an ordinary classifier requires a fixed number and ordering of output classes. So our model is more scalable and compact. The backbone F_θ

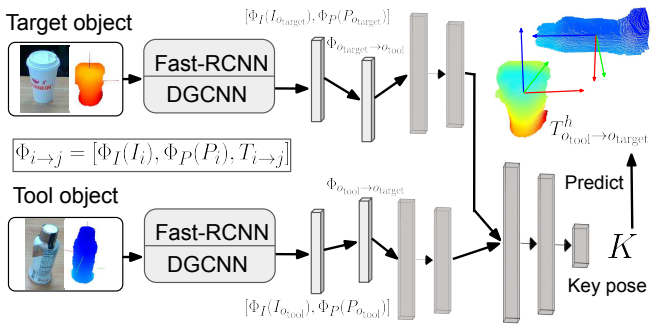

Fig. 3: Intermediate-level policy

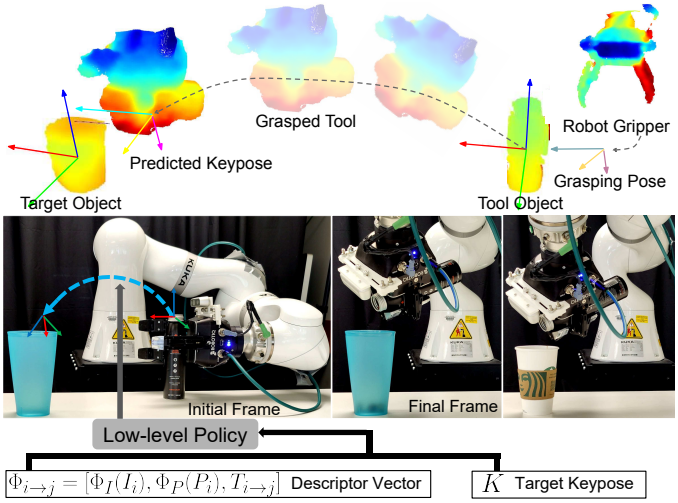

Fig. 4: Learned low-level policy. Note how the generated 6D keyposes and trajectories depend on the size of the target cup.

is composed of encoders consisting of fully connected layers for tool candidate descriptor, $\Phi_{j\to i}$, and target candidate descriptor, $\Phi_{i\to j}$, separately. The two branches return two encoding vectors, one for the tool and another for the target. The two encoding vectors are concatenated and provided as input to hidden fully connected layers, which finally output a scalar. In this design, F_θ receives only features for target and tool candidates and has no class-specific architecture or parameters, so it does not require predefined categories and it can generalize to objects with similar shape and appearance without manual labeling. It can also handle scenes with an arbitrary number of objects.

Intermediate-level policy. The pair of objects o_{target} and o_{tool}, received from high-level policy π_h, is provided as input to an intermediate-level policy π_m, which is a second neural network with a structure identical to F_θ, except for the last layer that returns a 6D keypose K instead of a scalar. Returned keypose K is the desired pose of the tool in the target object's coordinates system. For stacking, for example, desired keypose puts the tool object right on the surface of the target object. For painting, the keypose of a brush places the tip of the brush on the surface of another object that needs painting, and keeps the brush orthogonal to the target's surface. Keyposes are defined here as relative placements of objects with respect to each other, which are more consistent than the ones expressed in the camera coordinates system. Additionally, as only relative poses between objects are used here, one can easily use a different calibrated camera during robot execution (i.e. testing) without the need for an alignment with the camera pose used during the demonstrations.

We use rotation matrices in input features Φ and a quaternion for the orientation in output K, because transformations with homogeneous matrices can be easily represented in linear operations inside neural networks, while the quaternion is a better output format as it requires less constraints. Similar to high-level policy π_h, π_m is also class-agnostic, and keypose computation can be shared among similar-shaped objects, which facilitates the training.

Low-level policy. The last component of our decomposed policy is the low-level policy π_l that moves object o_{tool} to its next desired keypose K in the intrinsic frame of reference of object o_{target}, after receiving o_{tool}, o_{target} and K from the previous components. To be specific, $\pi_l = P(\Delta T_{\text{tool}\to\text{target}} | \Phi_{\text{tool}\to\text{target}}, K; \eta)$ where η are the low-level neural network's parameters. To facilitate the training, we also include $(K - T_{\text{tool}\to\text{target}})$ in the input of this low-level policy network, which has an inner structure that resembles the intermediate-level policy (Fig. 3). $\Delta T_{\text{tool}\to\text{target}}$ describes the change of tool's pose in each time step. After applying this change, a new state is sent to the policy, and the policy returns the next pose change. This process repeats until the end of the episode or reaching the predicted final keypose K. The robot's motion is performed by computing a change Δc_t from $\Delta T_{\text{tool}\to\text{target}}$ to apply on the robot's configuration c_t, using an inverse kinematics model of the robot, and the pose of the tool in the robot's frame. If o_{tool} selected by the high-level policy happens to be the robot's hand, then the low-level policy automatically closes the hand once o_{tool} is placed in keypose K, which results in grasping the object.

E. Learning from Demonstrations

During training, the system receives as inputs sequences of RGB-D images showing a human demonstrating a grasping-and-manipulation task on unknown objects. The system first automatically segments the images into individual point clouds, one per object, by removing the background. The objects include the human hand, for learning task-appropriate grasp poses. After segmenting the frames, an RGB-based tracker [34] is applied to match segments across consecutive frames. The segment with the most significant motion is labeled as o_{tool} unless that object is the hand of the demonstrator, in which case it is considered as the tool only if no object is being grasped. The object closest to the tool object in the end of the demonstration is labeled as o_{target}, and the transformation $T_{o_{\text{tool}}\to o_{\text{target}}}$ in the end of the demonstration is considered as the final keypose K. This entire process is fully automated and does not require any human input other than performing the demonstration. Given these labels, we train the high-level policy network to maximize the likelihood of $\pi_h(o_{\text{tool}}, o_{\text{target}})$. Intermediate-level and low-level policies are trained to minimize the mean squared errors.

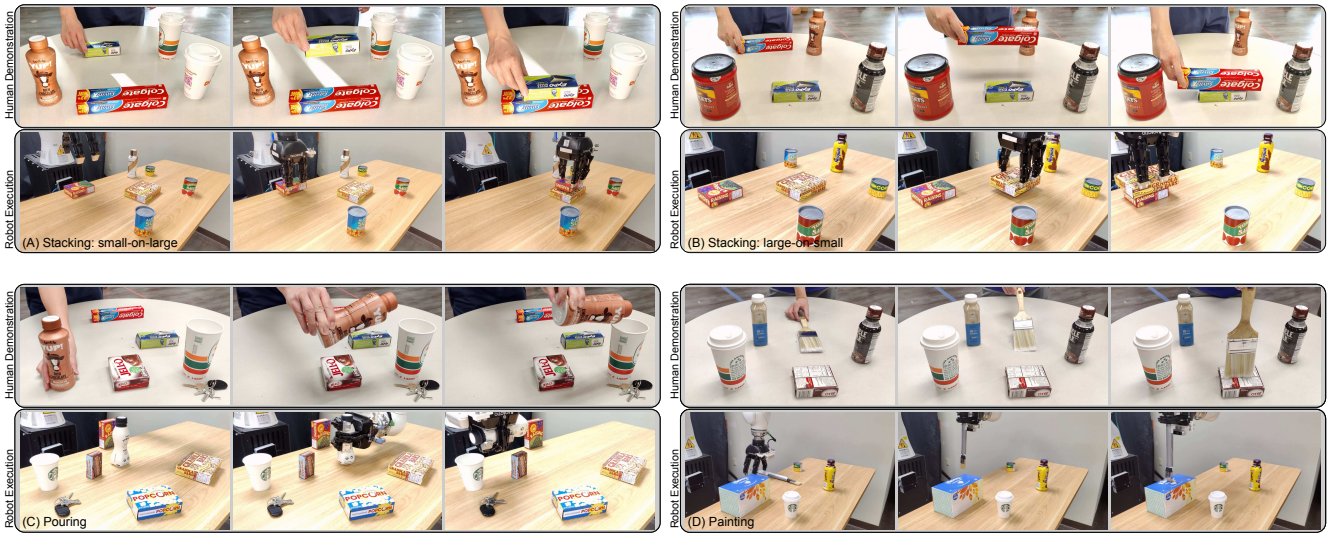

Fig. 5: Examples of demonstration videos used for training (top rows) in each of the four manipulation tasks, and examples of the resulting robot executions on novel scenes (bottom rows).

IV. Experiments

We evaluate the proposed method on four manipulation tasks, listed in Fig. 5, and compare it with several alternative architectures on the same tasks. For each task, we record 80 demonstration videos for training. The trained system is then tested on 20 different scenes per task. More details about these experiments are included in the long version of this paper available at https://tinyurl.com/ynavw363 along with videos of the robot experiments.

We compare the proposed approach with several network architectures and imitation learning techniques. The first three are **average pooling**, **max pooling**, and **attention**, which aggregate shape (from DGCNN) and appearance (from Fast-RCNN) descriptors of all the objects that are present in the scene into one large vector, and use that as an input to a neural network that returns a predicted target object, tool object and keypose. In the next two methods, we compare the proposed feature extractor with **ResNet** [35] and **Dense Object Nets** [11], which was specifically proposed for learning from demonstrations. Additionally, we compare against an **End-to-End** architecture based on ResNet [35], where the input is identical to our method's input, but the output is a low-level action that corresponds to moving the end-effector or grasping. We also compare with the Generative Adversarial Imitation Learning **(GAIL)** algorithm [36], with an architecture similar to ours, except that it directly predicts grasps and changes of hand poses in lieu of the proposed hierarchical decomposition.

The following tasks are used to evaluate the different learning methods. In **Small-on-large box stacking (A)**, the robot is tasked with finding the smallest box in the scene and stacking it on top of the largest one. A predicted keypose is considered accurate if it is within $3cm$ from the ground-truth. The order of the boxes in the stack is reversed in the **large-on-small stacking task (B)**. In the **pouring task (C)**, the robot is trained to find a bottle, grasp it from the side, move it close to a cup, and rotate it to point toward the cup in order to transfer its content to the cup without spilling. In the **painting task (C)**, the robot is trained to locate a brush, grasp it from the side, and use it to draw a short straight line on a canvas. In all four tasks, we evaluate the methods based on their: (1) accuracy in predicting target-tool pairs, (2) accuracy in predicting keyposes, and (3) overall task success rate, which combines together the two previous criteria.

Success / Trial	Task A	Task B	Task C	Task D
Proposed	**5/5**	**5/5**	**4/5**	**4/5**
Average Pooling	0/5	0/5	0/5	0/5
Max Pooling	2/5	1/5	1/5	0/5
Attention	1/5	1/5	1/5	1/5
ResNet [35]	0/5	1/5	1/5	0/5
End-to-End ResNet [35]	0/5	1/5	0/5	0/5
GAIL [36]	0/5	0/5	0/5	0/5
Dense Object Nets [11]	1/5	0/5	0/5	0/5

TABLE I: Real robot test success rates

Small-on-large box stacking (A), and large-on-small box stacking (B). Results in the top two rows of Fig.6 show the advantage of the proposed method in not only overall success rate but also in selecting tool-target pairs and generating keyposes. Furthermore, ResNet and Dense Object Nets baselines reach higher accuracy on target and tool object prediction than other baselines with other aggregation mechanisms. The advantage of DGCNN features is clearly shown through the decline in keypose prediction accuracy when the proposed method is used with ResNet or Dense Object Nets.

(C) Pouring. We can see from the third row of Fig. 6 a more pronounced advantage of the proposed method over other methods in keypose predictions, due to the more complex nature of the intermediate and low level policies in this task. The ground-truth keyposes depend on the initial relative pose of the bottle with respect to the cup because the demonstrator used the same (left or right) side for grasping and for pouring. The learned intermediate-policy extracts this information from the pairs of 6D poses of objects relative to each other.

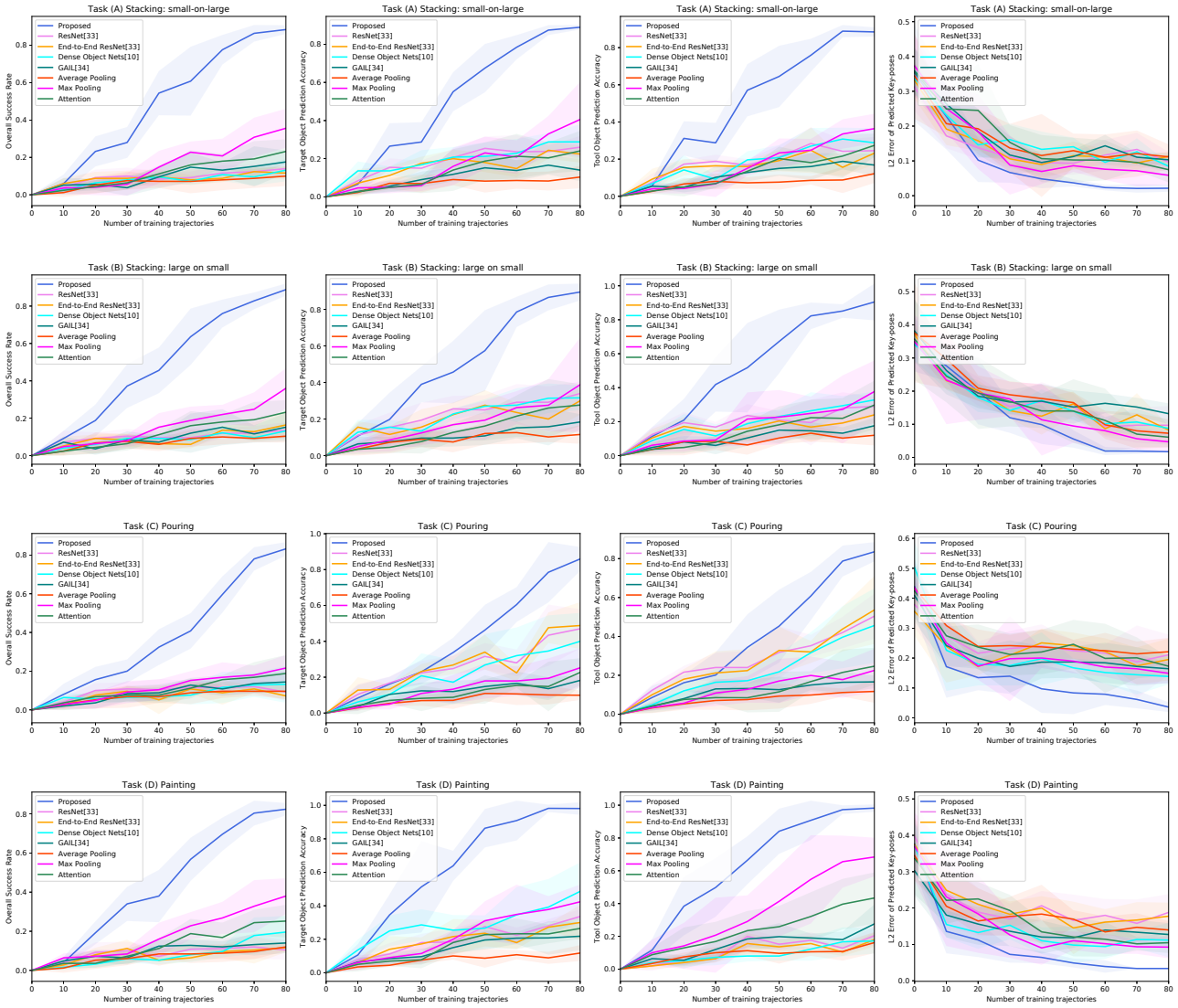

Fig. 6: Results of experiments on predicting tool-target pairs and keyposes in real novel static scenes, as a function of the number of videos used for training. The results are averaged over five independent trials.

(D) Painting. Results in the last row of Fig. 6 clearly show the advantage of the proposed method. But we can see that in this case, max pooling and attention achieve good accuracy in target object prediction. We hypothesize that the reason is that brushes have much different shapes than other objects present in the scene, and hence, the tool object is easier to be distinguished when DGCNN features are provided.

IoU	Task (A)	Task (B)	Task (C)
Proposed	84.1%	82.6%	74.6%
Without DGCNN	77.3%	78.1%	56.8%

TABLE II: IoU between generated and ground-truth trajectories

To assess DGCNN features on the low-level policy alone (without the tool/target and keypose selection), we also evaluate the Intersection over Union (IoU) between trajectories generated by the policy and ground-truth trajectories. Two poses from different trajectories are considered as an intersection if (1) they are within $2cm$ (2) the cosine between their rotation axes is higher than 0.9, and (3) the difference between their rotation angles is less than $20°$. The higher IoU from Table II suggests that DGCNN features help low-level motion generation. The improvement is more significant in pouring than in stacking because the size of the tool/target objects plays a more important role in pouring trajectories.

V. Conclusion

We presented a novel robot imitation learning framework for performing manipulation tasks in scenes that contain multiple unknown objects. The proposed model takes features from images and point clouds as input, and predicts a pair of target and tool objects, and a desired keypose for placing the tool relative to the target. The proposed system does not require any predefined class-specific priors, and can generalize to new objects of different shapes within the same category. Extensive experiments using real demonstration videos and a real robot show that the proposed model significantly outperforms state-of-the-art methods.

REFERENCES

[1] A. Boularias, O. Kroemer, and J. Peters, "Learning robot grasping from 3-d images with markov random fields," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*, 2011, pp. 1548–1553. [Online]. Available: http://dx.doi.org/10.1109/IROS.2011.6094888

[2] A. Boularias, J. A. Bagnell, and A. Stentz, "Learning to manipulate unknown objects in clutter by reinforcement," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, 2015, pp. 1336–1342. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9360

[3] R. Detry, C. H. Ek, M. Madry, and D. Kragic, "Learning a dictionary of prototypical grasp-predicting parts from grasping experience," 05 2013.

[4] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *International Journal of Robotics Research*, vol. 34, 01 2013.

[5] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4304–4311.

[6] X. Yan, J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson, and H. Lee, "Learning 6-dof grasping interaction via deep 3d geometry-aware representations," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2018)*, May 2018.

[7] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 2901–2910. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00299

[8] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2308–2315.

[9] D. Seita, N. Jamali, M. Laskey, A. K. Tanwani, R. Berenstein, P. Baskaran, S. Iba, J. Canny, and K. Goldberg, "Deep transfer learning of pick points on fabric for robot bed-making," 2019.

[10] L. Manuelli, W. Gao, P. R. Florence, and R. Tedrake, "kpam: Keypoint affordances for category-level robotic manipulation," *ArXiv*, vol. abs/1903.06684, 2019.

[11] P. R. Florence, L. Manuelli, and R. Tedrake, "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation," in *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, ser. Proceedings of Machine Learning Research, vol. 87. PMLR, 2018, pp. 373–385. [Online]. Available: http://proceedings.mlr.press/v87/florence18a.html

[12] S. Yang, W. Zhang, R. Song, J. Cheng, and Y. Li, "Learning multi-object dense descriptor for autonomous goal-conditioned grasping," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4109–4116, 2021.

[13] C. Chai, K. Hsu, and S.-L. Tsao, "Multi-step pick-and-place tasks using object-centric dense correspondences," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019*, ser. IEEE International Conference on Intelligent Robots and Systems. United States: Institute of Electrical and Electronics Engineers Inc., Nov. 2019, pp. 4004–4011, null ; Conference date: 03-11-2019 Through 08-11-2019.

[14] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese, "KETO: learning keypoint representations for tool manipulation," *CoRR*, vol. abs/1910.11977, 2019. [Online]. Available: http://arxiv.org/abs/1910.11977

[15] M. Vecerik, J.-B. Regli, O. Sushkov, D. Barker, R. Pevceviciute, T. Rothörl, C. Schuster, R. Hadsell, L. Agapito, and J. Scholz, "S3k: Self-supervised semantic keypoints for robotic manipulation via multi-view consistency," 2020.

[16] K. E. B. Junchi Liang, Bowen Wen and A. Boularias, "Learning sensorimotor primitives of sequential manipulation tasks from visual demonstrations," in *Proceedings of the 2022 International Conference on Robotics and Automation (ICRA)*, 2022.

[17] A. Morgan, B. Wen, J. Liang, A. Boularias, A. Dollar, and K. Bekris, "Vision-driven compliant manipulation for reliable, high-precision assembly tasks," in *Proceedings of Robotics: Science and Systems (RSS)*, 2021.

[18] J. Liang and A. Boularias, "Inferring time-delayed causal relations in pomdps from the principle of independence of cause and mechanism," in *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI), Montreal, Canada*, 2021.

[19] ——, "Self-supervised learning of long-horizon manipulation tasks with finite-state task machines," in *Proceedings of the Learning for Dynamics and Control Conference (L4DC), ETH, Zurich, Switzerland*, 2021.

[20] ——, "Learning transition models with time-delayed causal relations," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Nevada, US*, 2020.

[21] T. Schmidt, R. A. Newcombe, and D. Fox, "Self-supervised visual descriptor learning for dense correspondence," *IEEE Robotics Autom. Lett.*, vol. 2, no. 2, pp. 420–427, 2017. [Online]. Available: https://doi.org/10.1109/LRA.2016.2634089

[22] A. Boularias, J. A. Bagnell, and A. Stentz, "Efficient optimization for autonomous robotic manipulation of natural objects," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 2014, pp. 2520–2526. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8414

[23] A. T. Pas and R. Platt, "Using geometry to detect grasp poses in 3d point clouds," in *ISRR*, 2015.

[24] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, D. Kragic, A. Bicchi, and A. D. Luca, Eds. IEEE, 2016, pp. 3406–3413. [Online]. Available: https://doi.org/10.1109/ICRA.2016.7487517

[25] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 515–524.

[26] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," 2017.

[27] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," 2018.

[28] M. Laskey, J. Lee, C. Chuck, D. Gealy, W. Hsieh, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "Robot grasping in clutter: Using a hierarchy of supervisors for learning from demonstrations," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 827–834.

[29] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[30] A. ten Pas, M. Gualtieri, K. Saenko, and R. P. Jr., "Grasp pose detection in point clouds," *CoRR*, vol. abs/1706.09911, 2017. [Online]. Available: http://arxiv.org/abs/1706.09911

[31] B. Wen, W. Lian, K. E. Bekris, and S. Schaal, "Catgrasp: Learning category-level task-relevant grasping in clutter from simulation," *CoRR*, vol. abs/2109.09163, 2021. [Online]. Available: https://arxiv.org/abs/2109.09163

[32] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *CoRR*, vol. abs/1801.07829, 2018. [Online]. Available: http://arxiv.org/abs/1801.07829

[33] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[34] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[36] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, pp. 4565–4573, 2016.